

Volume

1
b

FOXES TEAM

Tutorial for BigMatrix.xla

Linear Algebra for Big Matrices

TUTORIAL FOR BIGMATRIX.XLA

Linear Algebra for Big Matrices

© 2003, by Foxes Team
Roma, Italy
leovlp@libero.it

BigMatrix for EXCEL

Package - ver. 1.0- Jan. 2003 - by Leonardo Volpi

BigMatrix is a tool for managing and performing heavy computation of large matrices with thousand of elements. The inversion of large structured matrices is a delicate problem which often arises in statistic, in finite element modeling applications, or in non-stationary inverse filtering problems in signal processing. Inversion or multiplication of matrices up to 200 x 200 needs some tips that are not usual present in common PC programs
 Hope this little work will be useful for someone.

Index

About BigMatrix.xla	3
How to install	4
How to unisintall	5
How to select a big matrix	6
Arithmetic accuracy	6
Elaboration time	9
Matrix manager	11
Matrix sub-selection	14
Matrix operations	17
Inversion.....	17
Transpose	22
Copy.....	22
Move	22
Addition	22
Subtraction	22
Multiplication	22
Scalar product.....	23
Linear System	24
Linear Regression	26
<i>Example: Perform the StRD Longley test</i>	26
Polynomial Regression.....	29
<i>Example: Perform the StRD Filippelli test</i>	29
<i>Summary of Certification Results for Linear Regression</i>	32
<i>Example: polynomial regression</i>	33
Mop-up	34
Round	34
Norma	34

About BigMatrix.xla

Matrix selection – the usual way for input – is not very handy when we have to select ranges of more than 10.000 cells

Standard 32 bit arithmetic in double precision is not more sufficient for ordinary accuracy.

Elaboration on-line – the usual way for Excel functions – becomes too heavy

To overcome these problems this addin has:

A matrices manager to select, copy, move, and input large range in a very simple way

Internal arithmetic of 30 significant digits; input and output are always done in 15 significant digits

Batch elaboration for the more heavy tasks like inversion and multiplication.

How to install

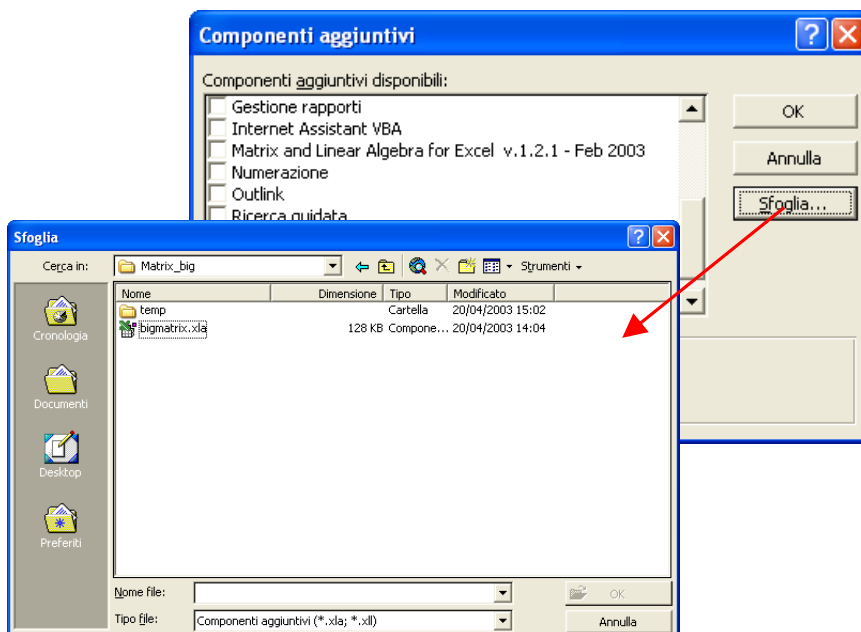
This addin for Excel 97/98/2000 comprises 6 files:

- 1) BIGMATRIX.XLA
- 2) MAT_INV.EXE
- 3) MAT_MULT.EXE
- 4) REG_LIN.EXE
- 5) SYS_LIN.EXE
- 6) BIGMATRIX.HLP

Place all the above files in a folder of your choice. The addin is contained entirely in this directory. Your system is not modified in any other way. If you want to uninstall this package, simply delete its folder - it's as simple as that!

Follow the usual procedure for installing Excel Addins:

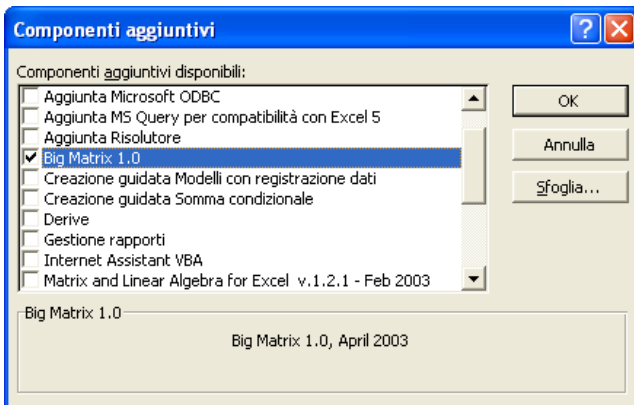
- 1) Open Excel
- 2) Select <Addins...> from the <Tools> menu
- 3) Once in the Addins Manager, search for and select BigMatrix.xla
- 4) Click OK and have fun!



Nella versione italiana di Excel, "Addin Manager" si chiama "Componenti aggiuntivi" a si trova nel menu Strumenti\ Modelli e aggiunte...

Tutorial for Bigmatrix.xla

After the first installation, BigMatrix.xla will be added to the Addins' list manager



When Excel starts, all addins checked in the Addins Manager will be automatically loaded

If you want to stop the automatic loading of matrix.xla simply deselect the check box before closing Excel

If all goes OK you should see the welcome popup of matrix.xla. This appears only when you select "on" the check box of the Addin Manager. When Matrix is automatically loaded by Excel, this popup is hidden.



How to unisintall

If you want to uninstall this package, simply delete its folder. Once you have cancelled the Matrix.xla file, to remove the corresponding entrie in the Addin Manager list, follow these steps:

- 1) Open Excel
- 2) Select <Addins...> from the <Tools> menu.
- 3) Once in the Addins Manager, click on the MATRIX.XLA
- 4) Excel will inform you that the addin is missing and ask you if you want to remove it from the list. Give "yes".

How to select a big matrix

Large matrices cannot be easily manipulated with usual methods. So this add-in offers several tools for handling wide range of thousand cells.

There are three different methods to select and input matrices for elaboration macros.

- 1) Selecting the first cell (top-left corner) and asking the manager to select automatically all adjacent cells. This method needs that the matrix must be surrounded by blank cells.
- 2) Selecting the first cell (top-left corner) and the last cell (bottom-right corner). This method is similar to the usual selection but without the dragging action of the mouse. Simply two clicks.
- 3) Selecting the first cell (top-left corner) and giving the exact dimension of rows and columns.

	A	B	C	D	E	F	G
1			First cell		autoselection		
2							
3							
4		74.5	48.5	34.9	40.4	69.8	
5		40.2	84.5	52.7	92.7	61.4	
6		28.9	36.5	64.7	43.4	13.7	
7		8.2	41.2	2.7	20.3	56	
8		68	7.4	12.2	8.5	94.2	
9		61	86.2	50.4	12.4	76.3	
10		88.4	47.1	68.7	15.3	79.6	
11							

	J	K	L	M
13			First cell	
14				
15				
16	92.1	9.4	40.9	21.6
17	97.7	85.4	45.1	76.8
18	49.9	54	36.4	9.3
19	84	19.9	88.1	92.9
20	26.4	49.8	52.5	58.8
21	66.3	11.2	75.5	67.6
22	18.1	97.4	46.4	23.5
23	3	14.3	83.9	46.1
24	90.6	99.4	3.6	20.2
25	82.5	98.3	61.1	60.7
26				
27			Last cell	
28				

All these simple tasks can be performed by the Matrix Manager, the Excel interface for large matrix operation.

Arithmetic accuracy

One recurrent question about matrix computation is: *What is the max dimension for a matrix operation, for example the determinant, or inversion?*

Well, the right answer is: it depends. Many factors, such as hardware configuration, algorithm, software coding, operating system and - of course - the matrix, contribute to limit the max dimension. One sure thing is that the limit is not fixed at all.

In the past, the main limitation was memory and elaboration speed, but nowadays there is no more a limit. We can say that, for the standard PC, the main limitation regards the 32-bit arithmetic and to the matrix itself.

Finite arithmetic limits the max dimension of matrices that can be elaborated in two fundamental aspects: overflow and round-off errors.

Overflow

Suppose you have a matrix (n x n) with its elements a_{ij} randomly distributed from -k to k. With this hypothesis the determinant grows roughly as:

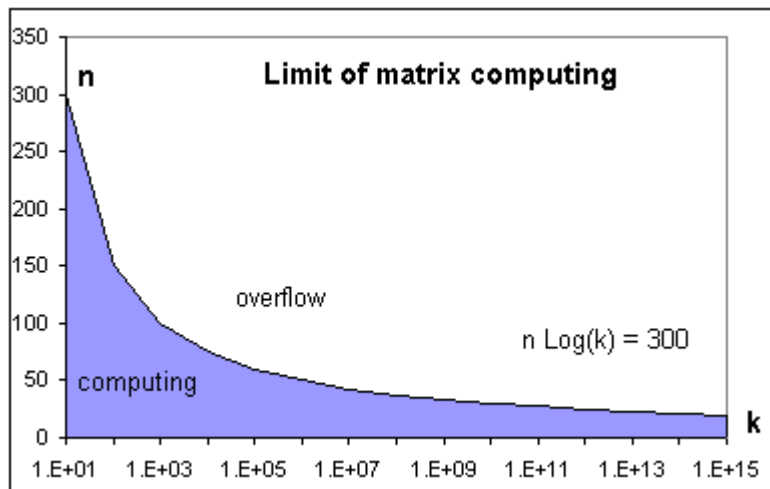
$$\text{Log}(|D|) \cong n \text{Log}(k) + 0.0027 \cdot n^2 \cong n \text{Log}(k)$$

where Log is decimal logarithm, n is the dimension of the matrix, k its max value.

In 32 bit double precision the max value allowed is about 1E+300, 1E-300. So if we want to avoid the overflow/underflow error, we must take the constrain:

$$300 \geq n \text{Log}(k) \quad (1)$$

Plotting this relation for all points (k, n) we have the area for computing (blue area in the graph below); the dangerous error area is the remaining (white) area.



How does it work?

Simple. If you have to compute the determinant of a matrix (80 x 80) having values no more than 1000, the point (1000, 80) falls into blue area; so you will be able to perform this operation. On the contrary, if you have a matrix (80 x 80) having values up to 1E+7, the point (1E+7, 80) falls into white area; so you will probably get an overflow error.

From this graph we see that matrices (25 x 25) or less, can be elaborated for all values, while matrices (100 x 100) or more can be computed only if their values are less than 1000.

Of course this result is valid only for generic matrices, not ill-conditioned. For this special case you can get an overflow/underflow error even for low/moderate values. Fortunately, there are also special kinds of matrices that can be elaborated even if the constrain (1) is false. We speak about diagonal, tridiagonal, sparse, block matrices etc.

Round-off errors

But to get a result does not mean a "good result". We have to take care, specially for large matrices, of the round-off errors. They are quite underhand and very difficult to detect also; very often the same results of large matrix inversion are taken as good even if they are wrong.

If you think that this kind of error regards only large matrices, have a look to the following example: Compute the numeric inverse of this simple (3 x 3) matrix

127	-507	245
-507	2025	-987
245	-987	553

If you use a 32-bit standard precision program on your PC, the answer looks like the following:

Tutorial for Bigmatrix.xla

-2.121E+14	-5.614E+13	-6.238E+12
-5.614E+13	-1.486E+13	-1.651E+12
-6.238E+12	-1.651E+12	-1.835E+11

And the determinant? You probably get a results near to $DET = -6.867E-10$

If you repeat the calculus with other programs you get similar results. Is there any reasons for suspecting this results? Yes, there is, because this result is **completely wrong !**. In fact, the determinant is 0, the given matrix is singular and its inverse, simply does not exist (you can easily compute it by hand with exact fractional numbers)

The matrix of above example is very hill-conditioned but what about other matrices that came across in ordinary application works? Are they so difficult? Fortunately not, but the round-off error limits the max dimension that we can successfully elaborate

We can observe this phenomena with **Tartaglia's matrices** with several dimension n . These matrices are very easy to generate and - this is very important - both matrix and its inverse have always integer values. It is very useful for testing the round-off error.

They are defined as following:

The matrix can be built starting from the first row - all 1 - and the first column - all 1.

After that, each cells is the sum of all left cells of the upper row.

$$a(1, j) = 1 \quad \text{for } j = 1 \dots n$$

$$a(i, 1) = 1 \quad \text{for } i = 1 \dots n$$

$$a(i, j) = \sum a(i-1, j) \quad \text{for } j = 2 \dots n$$

For example, here is a 6x6 Tartaglia's matrix and its inverse

1	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126
1	6	21	56	126	252

6	-15	20	-15	6	-1
-15	55	-85	69	-29	5
20	-85	146	-127	56	-10
-15	69	-127	117	-54	10
6	-29	56	-54	26	-5
-1	5	-10	10	-5	1

As we can see both matrices are integer. Any decimal of the inverse matrix must be regard as a round-off error and immediately detected.

This trick can be used to evaluate the round-off errors introduced by an algorithm for inverting matrices without to know exactly the result. For example, assume that in the inversion of a Tartaglia's matrix you find an element like this

$$A(1,1) = 5.99999999999985$$

Clearly, the exact value is 6 and the error will easily compute in the following way:

$$Err(1,1) = |A(1,1) - Round(A(1,1),0)| = 0.000000003456 \cong 1.518E-13$$

If we take the average of errors of all matrix elements, repeating for different dimension of Tartaglia's matrices, we have the following result:

Dim	Err	Err.rel
5	5.00E-16	6.00E-17
6	4.96E-12	6.27E-15
8	7.80E-10	9.62E-12
9	9.06E-09	1.94E-10
10	1.11E-05	2.85E-09
11	0.000286	3.23E-08
13	0.14	2.29E-06

As we can see the round-off errors became evident also for moderate dimension. You can try to obtain the inverse of a Tartaglia's matrix of 25 x 25 with standard 32-bit arithmetic ; the round-off errors will completely mask the true values.

This is the main reason because the max limit for PC programs is usually set around 25 – 50 BIGMATRIX performs all internal computation with 30 significant digits floating points arithmetic; input and output are always performed in standard 15 digits.

Elaboration time

In the past, elaboration for large matrices was performed on big mainframe (IBM 370, Vax, PDP11, etc.) and required several minutes or even hours. Typical job for this kind of elaboration was the batch submission: a file – input file - , containing the data to elaborate, was given in input to a program and, at the end of the process, the user got a second file – output file – containing the data elaborated (or the errors list!).

This similar job is replied in BIGMATRIX package for heavy duty task:

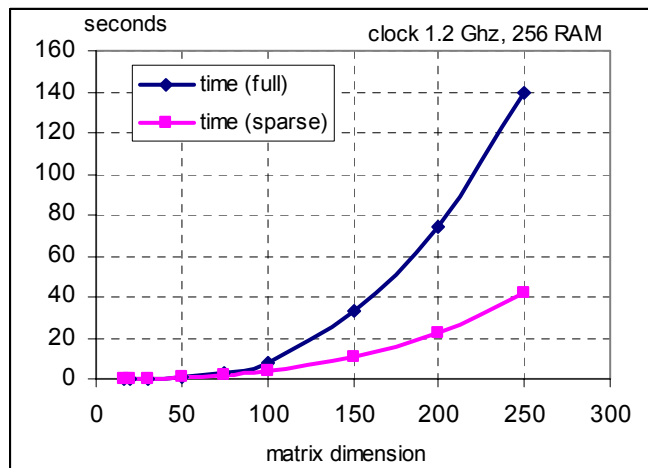
- Matrix inversion
- Matrices multiply
- Linear System resolution
- Linear Regression
- Polynomial Regression

Of course the elaboration time depend strongly by the matrices dimension and by the PC. Here same result obtained for matrix inversion: the time of the first column are measured with dense matrices while the second one is obtained with sparse (50%) matrices.

Tutorial for Bigmatrix.xla

Note: elaboration time obtained with 1.2 GHz Clock Pentium, 256 MB RAM

dim	time (full)	time (sparse)
16	0.04	-
20	0.08	-
30	0.23	0.07
50	1	0.5
75	3.3	1.5
100	8	3.5
150	33	10.4
200	74	22.8
250	140	42



As we can see the operative range of this package for matrix dimension is about 10 – 150; in this range the elaboration time is under 30 seconds. For larger dimension the time increase sharply and you have to wait for minutes.

Matrix manager

In order to provide a better way to manage large matrices we have developed a tool for helping with common matrices operations. Typical operations involves one, two or three matrices. For example the multiplication needs information for three matrices:

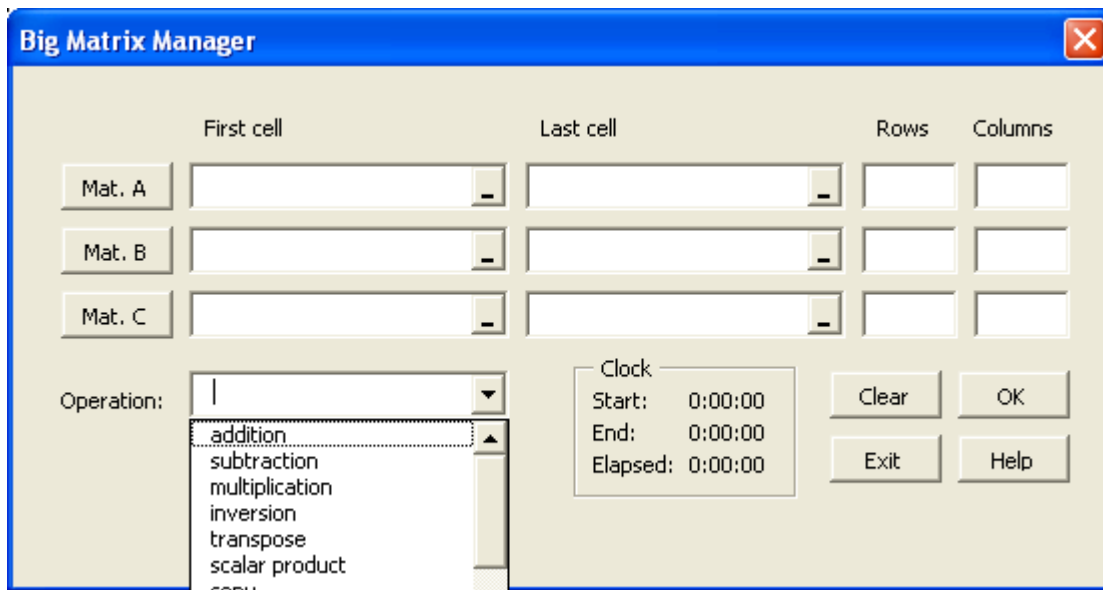
$$C = A \times B$$

The inversion requires only two matrices

$$B = \text{INV}(A)$$

And so on...

You can call the Matrix manager from the Excel menu **Tools\Big matrices...**



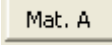
All operation are performed with this simple interface. Let's see how it works.

Example: Matrix Inversion

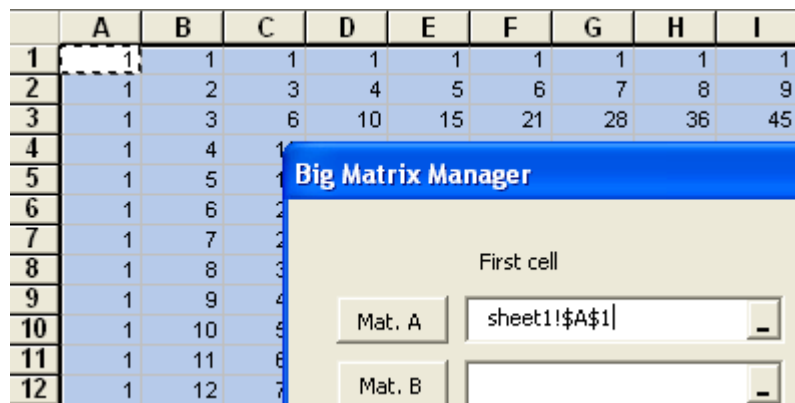
Assume you need to invert a matrix 20 x 20 that you have put in sheet1 in range A1:T20. And you want to obtain its inverse in sheet2 into the same range (but as we can see, you put it in any part of the same workbook as you want).

Open the Matrix manager and select the **inversion** from the combo box operation; move the cursor on the **first cell box** of **Mat. A** and after that with the mouse select the first cell of the input matrix: A1 in this example.

After click the button Mat.A

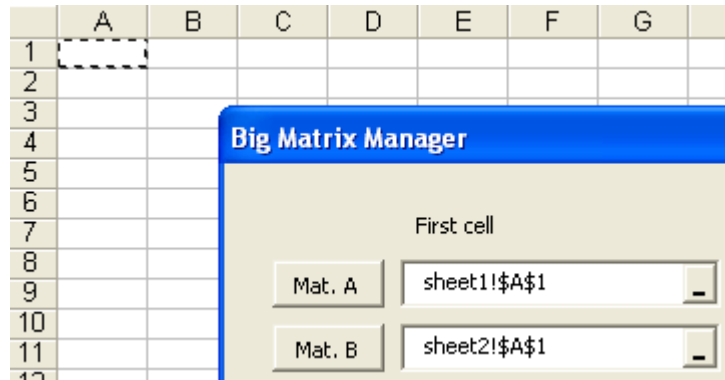
. This automatically select all adjacent cells of the whole matrix. You see that the exact row and column are calculated and show into the correspondents field.

Now you only indicate when the inverse matrix will be put. For that simply move the cursor into the First



Tutorial for Bigmatrix.xla

cell box of Mat . B and select the first cell of the output matrix: in the example the cell A1 of the sheet2. For output matrix you do not provide any other information because the machine will reserve all dimension that needs.



After that press OK and... wait. If all goes right you see, after any seconds, the following screen

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	20	-190	1140	-4845	15504	-38760	77520	-125970	167960	-184756	167960	-125970	77520	-387
2	-190	2470	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209
3	1140	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387
4	-4845	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504
5	15504	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816
6	-38760	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920
7	77520	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06
8	-125970	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06	2E+06
9	167960	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06	2E+06	-125970
10	-184756	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06	2E+06	-125970	167960
11	167960	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06	2E+06	-125970	167960	-184756
12	-125970	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-1E+06	167960
13	77520	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	658920	-125970
14	-38760	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-255816	77520
15	15504	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504	-38760
16	-4845	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-387	15504
17	1140	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	7209	-4845
18	-190	2470	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-1E+06	1140
19	20	-190	2470	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	2E+06	-190
20	-1	20	-190	2470	-16815	76551	-255816	658920	-1E+06	2E+06	-3E+06	3E+06	-3E+06	20
21														-1

The clock show the elaboration start time, stop time and elapsed time: 1.24 sec in that case, with a 1.2 GHz CPU Pentium machine.

You note behind the Matrix manager the 20x20 inverse matrix computed. It is exactly what you wanted !. Close the Matrix manager and use your result as you like.

Example: matrix Multiplication

Assume you need to multiply two matrices: the first one 10 x 10 that you have put in sheet1 in range A1:J10 and the second one 10 x 2 that is, in the same sheet, into the range K1:L10. And you want to obtain the matrix product in the same sheet, into the range N1:O10

Because the input matrices are not bounded by blank cells we indicated the first and the last cells for input. Let' see.

Tutorial for Bigmatrix.xla

Open the Matrix manager form and select the **multiplication** from the combo box operation; move the cursor on the **first cell box** of **Mat. A** and after that with the mouse select the first cell of the input matrix: A1 in this example.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1	1	1	1	1	1	1	1	1	1	100	1034			
2	1	2	3	4	5	6	7	8	9	10	250	1890			
3	0	3	6	7	9	11	18	15	17	19	400	2746			
4	-2	4	10	10	13	16	34	22	25	28	550	3602			
5	-5	5	15	13	17	21	55	29	33	37	700	4458			
6	-9														
7	-14														
8	-20														
9	-27														
10	-35														
11															
12															
13															
14															
15															
16															

Move the cursor on the **Last cell box** of **Mat. A** and after, with the mouse, select the last cell of the input matrix: J10 in this example. Press the button Mat.A to see the selected matrix. This is not necessary; but it is a useful tip to check a possible wrong selection

Repeat the same action for the second matrix; only choose the Mat.B fields and select the first cell K1 and the last cell L10

Tip: When you try to select a cell in the worksheet, the Matrix manager form could obstruct you.

You can easily reduce it by the little button  at the right of each cell-field.



Clicking on the button  the Matrix manager re-appears.

At least, you only tell when the product matrix will be put. For that simply move the cursor into the First cell box of Mat. C and select the first cell of the output matrix: in the example the cell N1. For output matrix you never provide any other information because the machine will reserve all dimension that needs.

Choose OK to start the multiplication job. For high dimension, the elaboration time will be relevant. So be patient!

Note: During the elaboration, the clock shows the elapsed time. Do not close the Matrix manager because, it will not be able to load automatically the result matrix (for v.1.0). If it happens, you can do that manually. In fact, the result is never missed. It is saved in the file #MAT003.TXT of the temp directory under the main directory that you have chosen for BIGMATRIX. Results in #MAT003.TXT are in ASCII tab delimited with 16 digits exponential format.

Matrix sub-selection

Several time we have to select and extract part of matrices. Typical sub-parts are: diagonals, triangular lower or upper side, etc.

Excel can do that with the multi-range selection (selection with CTRL key keep down) but, it cannot permit the copy of a multi-range.

To overcome this limitation the BigMatrix addin has a few macros to perform the selection and the copy of this kind of range. They are useful overall for large matrices. Let's see how they work

You can call these macros from the menu

Tools\Big matrices...\Select scraps ([Tool for selection of matrix parts](#))

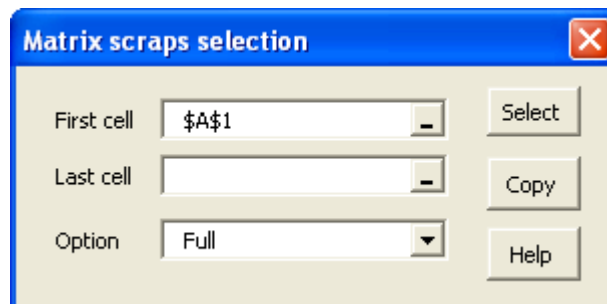
Tools\Big matrices...\Copy scraps ([Tool for copy matrix parts selected](#))

Note: The copy tool is accessible also from the Select form

Note: The macro **Tools\Big matrices...\Select** is suitable to auto select all matrix' element simply by selecting the left-upper corner of the matrix and the click on the menu **select** i Just straight and simple. The selection ended to the first empty column to right and to the first empty row to bottom.

Tool for selection of matrix parts

From menu **Tools\Big matrices...\Select scraps**



First Cell: select the first cell of the sub part that you want to copy: matrix, diagonal, sub-matrix

Last Cell: select the last cell of the sub part that you want to copy: matrix, diagonal, sub-matrix. If omitted, the selection extend itself till the matrix borders.

Option: Tell the program which sub part you want to select. You can choose one of the following option:

- = "Full" (the same as the **select** command of the menu)
- = "Diagonal left"
- = "Diagonal right"
- = "Triang. lower"
- = "Triang. upper"
- = "Subtriang. lower"
- = "Subtriang. upper"

Here is how it works. Select the first upper corner (the element a_{11}) of a square matrix (for simplicity) and leave it; than choose in sequence all the options. You see the following pattern:

1	2	3	4	5	6
0	3	6	2	1	0
-1	4	9	0	3	-6
-2	5	12	-2	7	-1
-3	6	15	-4	11	-2
-4	7	18	-6	15	8

	1	2	3	4	5	6
0	3	6	2	1	0	
-1	4	9	0	3	-6	
-2	5	12	-2	7	-1	
-3	6	15	-4	11	-2	
-4	7	18	-6	15	8	

1	2	3	4	5	6
0	3	6	2	1	0
-1	4	9	0	3	-6
-2	5	12	-2	7	-1
-3	6	15	-4	11	-2
-4	7	18	-6	15	8

1	2	3	4	5	6
0	3	6	2	1	0
-1	4	9	0	3	-6
-2	5	12	-2	7	-1
-3	6	15	-4	11	-2
-4	7	18	-6	15	8

	1	2	3	4	5	6
0	3	6	2	1	0	
-1	4	9	0	3	-6	
-2	5	12	-2	7	-1	
-3	6	15	-4	11	-2	
-4	7	18	-6	15	8	

	1	2	3	4	5	6
0	3	6	2	1	0	
-1	4	9	0	3	-6	
-2	5	12	-2	7	-1	
-3	6	15	-4	11	-2	
-4	7	18	-6	15	8	

Of course it works also for any rectangular matrix.

You can select also any sub-diagonal that you like. For example if you need to select the sub diagonal lower; simply choose as the first cell the element a₁₂, the option diagonal left, and click the select button.

	1	2	3	4	5	6
0	3	6	2	1	0	
-1	4	9	0	3	-6	
-2	5	12	-2	7	-1	
-3	6	15	-4	11	-2	
-4	7	18	-6	15	8	

Note: The selection of matrix's parts is obtained by a multi range selection. Excel cannot copy it. If you try to give the usual sequence CTRL+C you will have an error. To do it, call the Copy Scraps tool

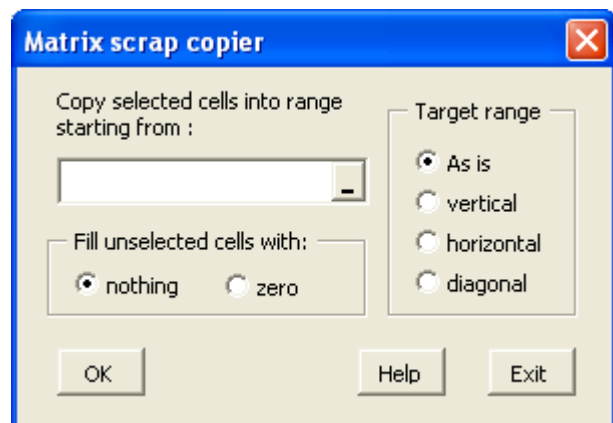
Tool for copy matrix parts selected

From menu **Tools\Big matrices...\Copy scraps**

Or, simply, by click on the button **copy** in the **Select scrap** tool

To copy the multi range previous selected simply indicate the target range (first cell) That's all.

Of course the tool have other interesting options. Let's see.



Fill unselecting cells with zero: check this if you fill all other cell of the matrix with zero. This is useful to build a new matrix with a part of the original one.

Tutorial for Bigmatrix.xla

Example: If you want to build a new lower triangular matrix with the element of a given matrix you can use this simply option and the result will be similar to the following:

1	2	3	4	5	6	1	0	0	0	0	0
0	3	6	2	1	0	0	3	0	0	0	0
-1	4	9	0	3	-6	-1	4	9	0	0	0
-2	5	12	-2	7	-1	-2	5	12	-2	0	0
-3	6	15	-4	11	-2	-3	6	15	-4	11	0
-4	7	18	-6	15	8	-4	7	18	-6	15	8

Change the target range: Normally the range is copied as is. But sometime we need to reconvert the topology of the given range. This happens, for example when we want to extract the diagonal elements from a given matrix and to convert it in a vertical vector.

In this case, after you have selected the diagonal, check the option "vertical" and the copy it

The diagonal element will be.. "verticalized". Fine, isn't?

1	2	3	4	5	6	1
0	3	6	2	1	0	3
-1	4	9	0	3	-6	9
-2	5	12	-2	7	-1	-2
-3	6	15	-4	11	-2	11
-4	7	18	-6	15	8	8

Some time we need the inverse of this transformation: from a vertical vector, we have to build the diagonal matrix having with the vector elements on its diagonal.

For that select the vector that contains the elements. Start the **copy scraps** tool and check the "zero" and "diagonal" option. Giving OK, you will generate the matrix to the left

1	1	0	0	0	0	0
3	0	3	0	0	0	0
9	0	0	9	0	0	0
-2	0	0	0	-2	0	0
11	0	0	0	0	11	0
8	0	0	0	0	0	8

Matrix operations

BigMatrix addin 1.0 can perform the following matrix operations:

One matrix operations

Operation	Input matrix	Output matrix
Inversion	A (n x n)	B (n x n)
Transpose	A (n x m)	B (m x n)
Copy	A (n x m)	B (n x m)
Move	A (n x m)	B (n x m)
Mop-up	A (n x m)	A (n x m)
Round	A (n x m)	A (n x m)
Normal	A (n x m)	B(4 x 1)

Two matrices operations

Operation	Input matrix	Input matrix	Output matrix
AdditionI	A (n x m)	B (n x m)	C (n x m)
SubtractionI	A (n x m)	B (n x m)	C (n x m)
MultiplicationI	A (n x m)	B (m x q)	C (n x q)
Scalar productI	A (n x m)	B (n x q)	C (m x q)
Linear SystemI	A (n x n)	B (n x q)	C (n x q)
Linear RegressionI	A (n x m)	B (n x 1)	C (n+1 x 1)
Polynomial RegressionI	A (n x p), degree	B (n x 1)	C (p*degree+1 x 1)

Inversion

Returns the inverse of non singular square matrix $A(n \times n) \Rightarrow B(n \times n)$

Input and Output values have max 15 significant digits

Internal computing is performed with 30 significant digits

This task is performed by the program **mat_inv.exe** of this package.

Syntax command line

```
>> mat_inv filenameA.txt filenameB.txt
```

FilenameA contains the element of matrix to invert in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of inverted matrix in ASCII tab-delimited. Numbers are always in exponential format with 15 significant digits

Algorithm used is the Gauss-Jordan with full pivot strategy

Example

Find the inverse of the 16 x 16 Tartaglia's matrix

To show the nature of this hill conditioned matrix , first of all, we compute the inverse with MINVERSE() Excel built-in function

The screenshot shows an Excel spreadsheet with two main sections. The first section, rows 2-17, contains Tartaglia's 16x16 matrix. The second section, rows 20-35, contains the inverse of this matrix, labeled A⁻¹. The spreadsheet interface includes the menu bar, toolbar, and sheet tabs.

As we can see, several decimals begin to appear in the inverse matrix:

$a_{11} = 16.0056$, $a_{12} = -120.08$, etc..

Inverse of Tartaglia's matrix is always integer, so the decimals are produced by round-off error

For proof we calculate the product

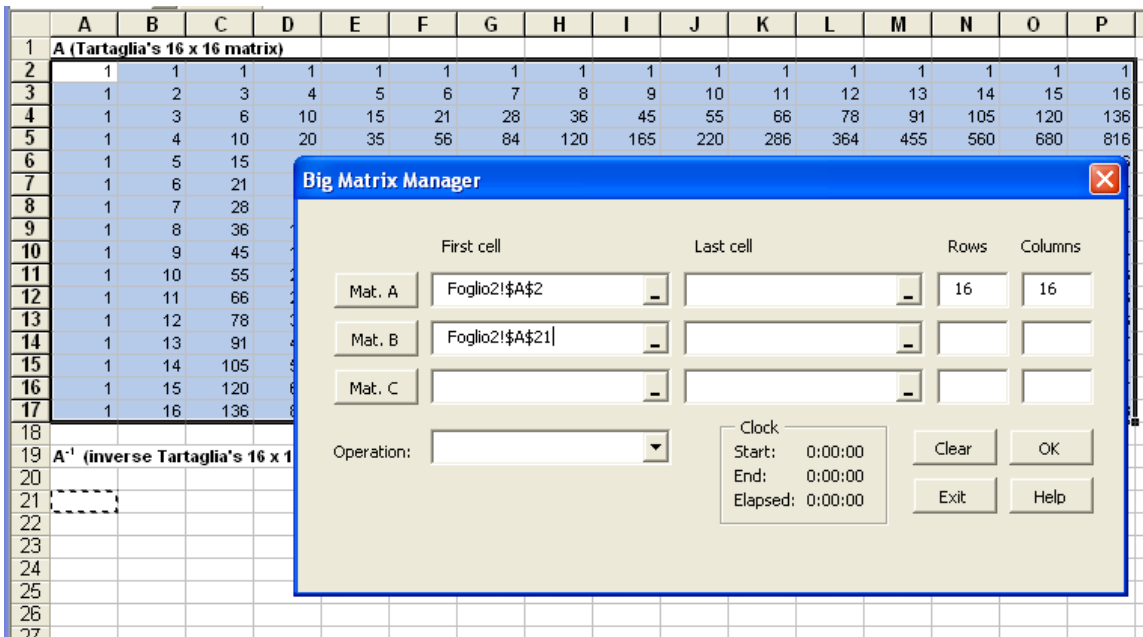
$$AA^{-1} = I$$

that should gives the identity matrix. We can use the MMULT() Excel built-in function

The screenshot shows the result of the MMULT function applied to the matrix and its inverse. The result is the identity matrix, with diagonal elements close to 1 and off-diagonal elements close to 0. The spreadsheet shows numerical values for each cell, with some cells in scientific notation.

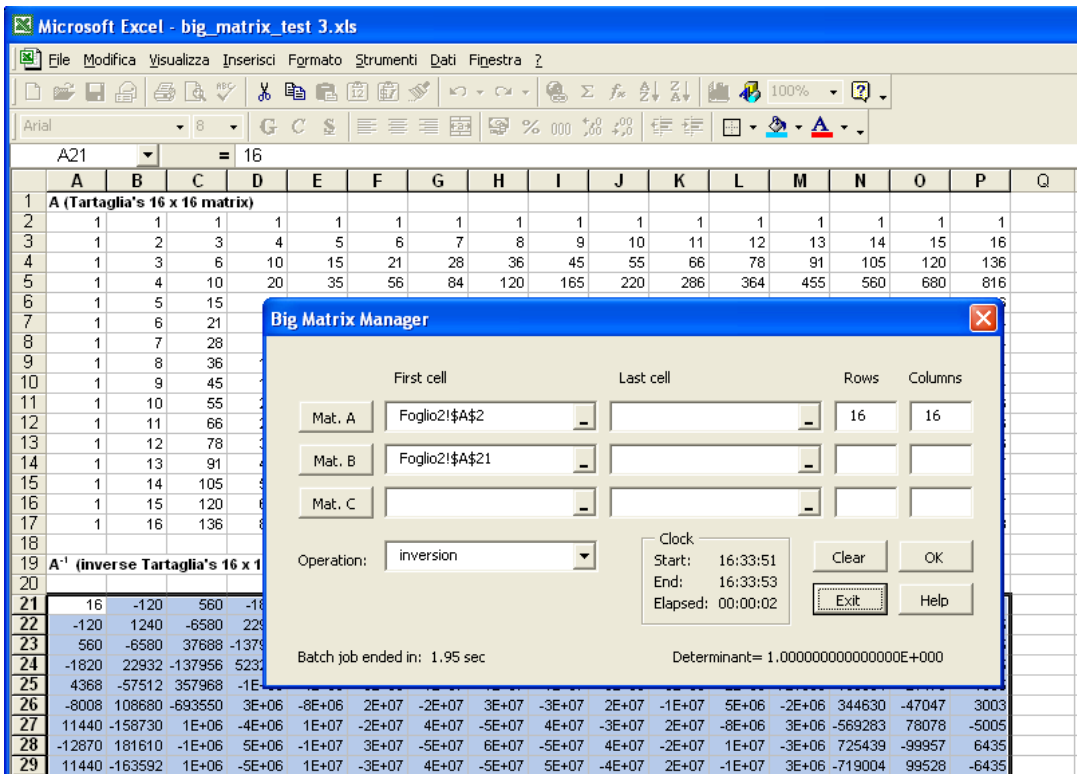
As we can see, the general accuracy is quite poor. This is due to the limit of 16 digits floating points arithmetic. The only way to gain precision is to use more digits
For example, BigMatrix use 30 digits for internal computing. Let's see.

Starting the Matrix Manager from the menu **Tools \ Big matrices... \ Manager**



The cursor is in the First cell of matrix A; move to the sheet and select the cell A2; now move to the Manager and click the **Mat. A** button

This will select all matrix A; now move the cursor to the first cell of the matrix B; move to the sheet and select the first cell A21 where you want to insert the inverse matrix. Now choose the operation "**inverse**" and gives **OK**





The batch elaboration starts and after a while (about 2 sec) The matrix B appears in the sheet. Note the exact value of determinant = 1

Tutorial for Bigmatrix.xla

Now compute the product $\mathbf{A} \mathbf{A}^{-1} = \mathbf{I}$ with BigMatrix tools

Starting the Matrix Manager from the menu **Tools \ Big matrices... \ Manager**

The cursor is in the first cell of matrix A; move to the sheet and select the cell A2; move to the Manager and click the **Mat. A** button  This will select all matrix A

move the cursor to the first cell of the matrix B; move to the sheet and select the first cell A21; move to the Manager and click the **Mat. B** button  This will select all matrix B

Now move the cursor to the first cell of the matrix C; move to the sheet and select the first cell A39.

Now choose the operation "**multiplication**" and gives **OK**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
19	A⁻¹ (inverse Tartaglia's 16 x 16 matrix)															
20																
21	16	-120	560	-1820	4368	-8008	11440	-12870	11440	-8008	4368	-1820	560	-120	16	-1
22	-120	1240	-6580	22932	-57512	108680	-158730	181610	-163592	115752	-63700	26740	-8280	1784	-239	15
23	560	-6580	37688	-137956	357968	-693550	1E+06	-1E+06	1E+06	-781508	433720	-183380	57136	-12377	1666	-105
24	-1820	22932	-137956	523720	-1086800	1587300	-1816100	1635920	-1157520	637000	-267400	82800	-17840	2390	-16660	10500
25	4368	-57512	357968	-1086800	1587300	-1816100	1635920	-1157520	637000	-267400	82800	-17840	2390	-16660	10500	-7815080
26	-8008	1086800	-6935500	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06
27	11440	-158730	1E+06	-4E+06	4E+06	-4E+06	4E+06	-4E+06	4E+06	-4E+06	4E+06	-4E+06	4E+06	-4E+06	4E+06	-4E+06
28	-12870	181610	-1E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06
29	11440	-163592	1E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06	5E+06	-5E+06
30	-8008	115752	-781508	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06	-3E+06	3E+06
31	4368	-63700	433720	-2E+06	2E+06	-2E+06	2E+06	-2E+06	2E+06	-2E+06	2E+06	-2E+06	2E+06	-2E+06	2E+06	-2E+06
32	-1820	26740	-183380	778000	-778000	778000	-778000	778000	-778000	778000	-778000	778000	-778000	778000	-778000	778000
33	560	-8280	57136	-244000	244000	-244000	244000	-244000	244000	-244000	244000	-244000	244000	-244000	244000	-244000
34	-120	1784	-12377	53000	-53000	53000	-53000	53000	-53000	53000	-53000	53000	-53000	53000	-53000	53000
35	16	-239	1666	-7000	7000	-7000	7000	-7000	7000	-7000	7000	-7000	7000	-7000	7000	-7000
36	-1	15	-105	4000	-4000	4000	-4000	4000	-4000	4000	-4000	4000	-4000	4000	-4000	4000
37																
38	A A⁻¹															
39	1	0	0													
40	0	1	0													
41	0	0	1													
42	0	0	0													
43	0	0	0													
44	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Big Matrix Manager

First cell: Mat. A: Foglio2!\$A\$2, Mat. B: Foglio2!\$A\$21, Mat. C: Foglio2!\$A\$39

Last cell: (empty)

Rows: 16, Columns: 16

Operation: multiplication

Clock: Start: 17:05:44, End: 17:05:45, Elapsed: 00:00:01

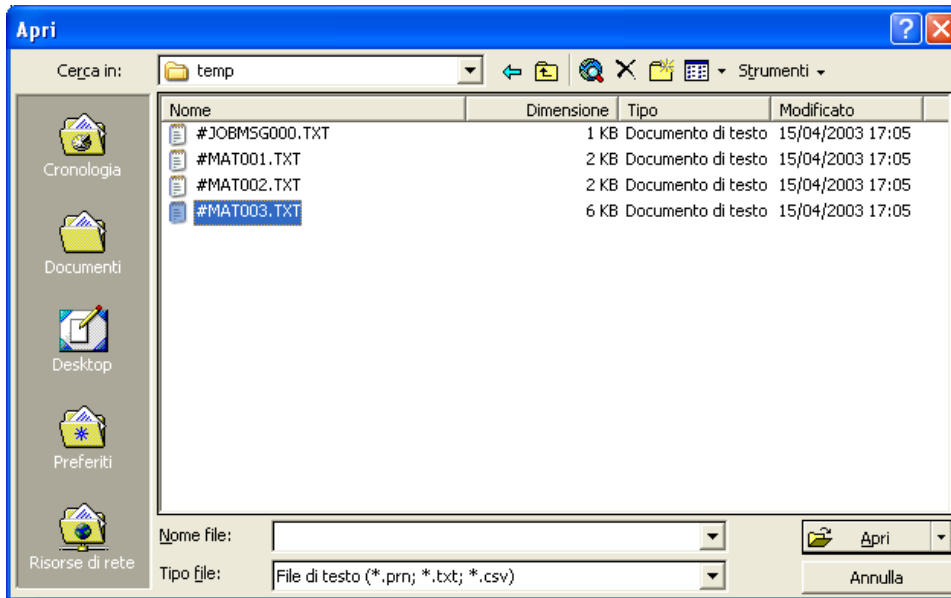
Buttons: Clear, OK, Exit, Help

Batch job ended in: 1.07 sec

We will get a perfect identity matrix! This means that round-off errors are less than 1E-16

What happens if we close the matrix manager before ending the batch elaboration ?

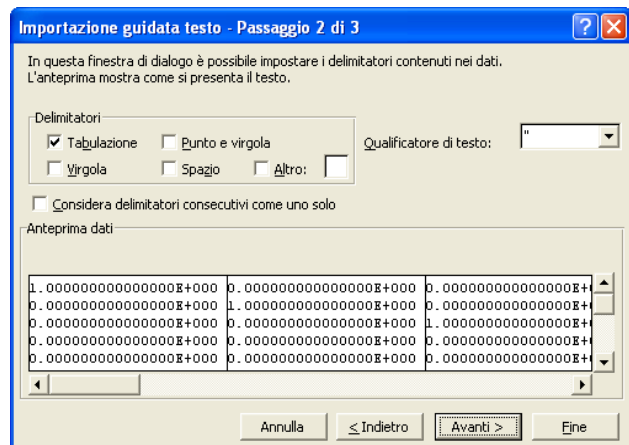
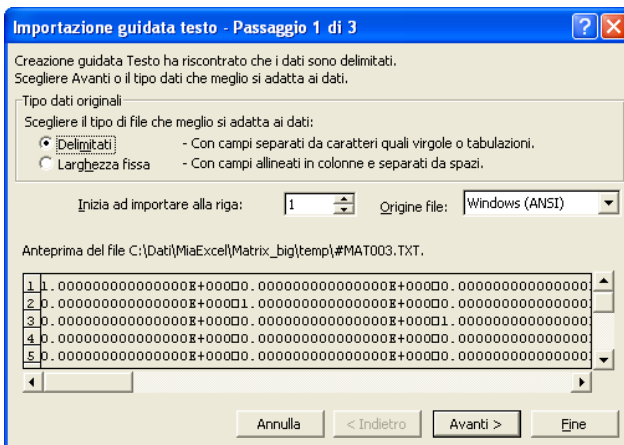
Sometime it can happen to close the window of the matrix manager before the end of an elaboration. In that case the manager will not be able to load the matrix result into the Excel sheet but the all elaboration result will be save, because it is stored in the file #MAT003.TXT in the temp directory where you have put the BigMatrix installation.



To load manually the file #MAT003.TXT simply open file from Excel menu; select text file type *.txt and search for the #MAT003.TXT in the temp sub-directory of BigMatrix directory installation

After In the next windows choose "delimited" and "tab" in the next window

Give End to import the matrix file into Excel



File imported are always in exponential format with point as decimal separator.

Transpose

Returns the transpose of a matrix $A(n \times m) \Rightarrow B(m \times n)$
No arithmetic computations are performed with this task
Result can be put in any range of any sheet of the workbook

Copy

Copy a matrix $A(n \times m)$ to another location of the workbook
No arithmetic computations are performed with this task
Result can be put in any range of any sheet of the workbook

Move

Move a matrix $A(n \times m)$ to another location of the workbook
No arithmetic computations are performed with this task
Result can be put in any range of any sheet of the workbook
The original matrix is removed

Addition

Returns the addition of two matrices, that must have the same dimensions.
Result can be put in any range of any sheet of the workbook
Arithmetic computing is performed with standard 32-bit double precision

Subtraction

Returns the subtraction of two matrices, that must have the same dimensions.
Result can be put in any range of any sheet of the workbook
Arithmetic computing is performed with standard 32-bit double precision

Multiplication

Returns the product of two matrix $A(n \times m) \times B(m \times p) \Rightarrow C(n \times p)$
Input and Output values have max 15 significant digits
Internal computing is performed with 30 significant digits
This task is performed by the program **mat_mult.exe** of this package.
Syntax command line

```
>> mat_mult filenameA.txt filenameB.txt filenameC.txt
```

FilenameA contains the element of the first matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of the second matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameC contains the element of product matrix in ascii tab-delimited. Numbers are always in exponential format with 15 significant digits

Scalar product

Returns the scalar product of two matrix $A(n \times m) * B(n \times p) \Rightarrow C(m \times p)$

If $m=1$ and $p=1$, this job returns one value, that is the scalar product of two vectors.

Input and Output values have max 15 significant digits

Internal computing is performed with 30 significant digits

This task is performed by the program **mat_mult.exe** of this package.

Syntax command line

```
>> mat_mult filenameA.txt filenameB.txt filenameC.txt
```

FilenameA contains the element of the transpose of the first matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of the second matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameC contains the element of product matrix in ascii tab-delimited. Numbers are always in exponential format with 15 significant digits

Linear System

Solve a linear system $\mathbf{A}(n \times n) \mathbf{X}(n \times m) \Rightarrow \mathbf{B}(n \times m)$

\mathbf{A} must be non singular.

Input and Output values have max 15 significant digits

Internal computing is performed with 30 significant digits

This task is performed by the program **sys_lin.exe** of this package.

Syntax command line

```
>>sys_lin filenameA.txt filenameB.txt filenameC.txt
```

FilenameA contains the element of the system matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of the B matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameC contains the element of the system solution in ascii tab-delimited. Numbers are always in exponential format with 15 significant digits

Example: solve the following 16 x16 linear system $\mathbf{A} \mathbf{x} = \mathbf{b}$

The 17th column is the \mathbf{b} constant term

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	0	0
1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	0	0
1	4	10	20	35	56	84	120	165	220	286	364	455	560	680	816	0	0
1	5	15	35	70	126	210	330	495	715	1001	1365	1820	2380	3060	3876	0	0
1	6	21	56	126	252	462	792	1287	2002	3003	4368	6188	8568	11628	15504	0	0
1	7	28	84	210	462	924	1716	3003	5005	8008	12376	18564	27132	38760	54264	0	0
1	8	36	120	330	792	1716	3432	6435	11440	19448	31824	50388	77520	116280	170544	0	0
1	9	45	165	495	1287	3003	6435	12870	24310	43758	75582	125970	203490	319770	490314	0	0
1	10	55	220	715	2002	5005	11440	24310	48620	92378	167960	293930	497420	817190	1307504	0	0
1	11	66	286	1001	3003	8008	19448	43758	92378	184756	352716	646646	1144066	1961256	3268760	0	0
1	12	78	364	1365	4368	12376	31824	75582	167960	352716	705432	1352078	2496144	4457400	7726160	0	0
1	13	91	455	1820	6188	18564	50388	125970	293930	646646	1352078	2704156	5200300	9657700	17383860	0	0
1	14	105	560	2380	8568	27132	77520	203490	497420	1144066	2496144	5200300	10400600	20058300	37442160	0	0
1	15	120	680	3060	11628	38760	116280	319770	817190	1961256	4457400	9657700	20058300	40116600	77558760	0	0
1	16	136	816	3876	15504	54264	170544	490314	1307504	3268760	7726160	17383860	37442160	77558760	155117520	0	0

Assume the matrix \mathbf{A} in the range A2:P17, an \mathbf{b} in the range Q2:Q17,

Start the matrix manager and select the first cell A2; move the cursor to the last cell box of matrix A; select the last cell P17 of the matrix; move to the Manager and click the **Mat. A** button

This will select the 16 x 16 cells of the matrix A. Now move the cursor into the first cell

box of matrix B and select the cell Q2; move to the Manager and click the **Mat. B** button

This will select all constant term \mathbf{b}

Remain to indicate where we want the solution vector; move the cursor to the first cell box of matrix C and select the cell R2.

Select the operation "**Linear System**" and give OK to start the proces

Tutorial for Bigmatrix.xla

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	A (Tartaglia's 16 x 16 matrix)																		
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	16
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	0	0	-120
4	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	0	0	560
5	1	4	10	20	35	56	84	120	165	220	286	364	455	560	680	816	0	0	-1820
6	1	5	15	35	70	126	210	330	495	715	1001	1365	1820	2380	3060	3876	0	0	4368
7	1	6	21	56	126	252	462	792	1287	2002	3003	4368	6188	8568	11628	15504	0	0	-8008
8	1	7	28	84	210	462	924	1716	3003	5005	8008	12376	18564	27132	38760	54264	0	0	11440
9	1	8	36	120	330	792	1716	3432	6435	11440	19448	31824	50388	77520	116280	170544	0	0	-12870
10	1	9	45	165	495	1287	3003	6435	12870	24310	43758	75582	125970	203490	319770	490314	0	0	11440
11	1	10	55	220	715	2002	5005	11440	24310	48620	92378	167960	293930	497420	817190	1307504	0	0	-8008
12	1	11	66	286	1001	3003	8008	19448	43758	92378	184756	352716	646646	1144066	1961256	3268760	0	0	4368
13	1	12	78	364	1365	4368	12376	31824	75582	167960	352716	705432	1352078	2496144	4457400	7726160	0	0	-1820
14	1	13	91	455	1820	6188	18564	50388	125970	293930	646646	1352078	2704156	5200300	9657700	17383860	0	0	560
15	1	14	105	560	2380	8568	27132	77520	203490	497420	1144066	2496144	5200300	10400600	20058300	37442160	0	0	-120
16	1	15	120	680	3060	11628	38760	116280	319770	817190	1961256	4457400	9657700	20058300	40116600	77558760	0	0	16
17	1	16	136	816	3876	15504	54264	170544	490314	1307504	3268760	7726160	17383860	37442160	77558760	155117520	0	0	-1

Big Matrix Manager

First cell: Mat. A: Foglio3!\$A\$2, Last cell: Foglio3!\$P\$17, Rows: 16, Columns: 16

Mat. B: Foglio3!\$Q\$2, Last cell: Foglio3!\$Q\$17, Rows: 16, Columns: 1

Mat. C: Foglio3!\$R\$2, Last cell: , Rows: , Columns:

Operation: Linear System

Clock: Start: 17:59:45, End: 17:59:46, Elapsed: 00:00:01

Clear OK

Exit Help

Batch job ended in: .87 sec Determinant= 1.000000000000000E+000

We note the exact accuracy of the result even for this hill-conditioned system. If we try to repeat the calculus with the built-in function MMULT and MINVERSE we have a different result

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	A (Tartaglia's 16x16 matrix)																		
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	16.0056
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	0	0	-120.06545
4	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	0	0	560.3525
5	1	4	10	20	35	56	84	120	165	220	286	364	455	560	680	816	0	0	-1821.1526
6	1	5	15	35	70	126	210	330	495	715	1001	1365	1820	2380	3060	3876	0	0	4370.5299
7	1	6	21	56	126	252	462	792	1287	2002	3003	4368	6188	8568	11628	15504	0	0	-8011.8716
8	1	7	28	84	210	462	924	1716	3003	5005	8008	12376	18564	27132	38760	54264	0	0	11444.095
9	1	8	36	120	330	792	1716	3432	6435	11440	19448	31824	50388	77520	116280	170544	0	0	-12872.723
10	1	9	45	165	495	1287	3003	6435	12870	24310	43758	75582	125970	203490	319770	490314	0	0	11440.579
11	1	10	55	220	715	2002	5005	11440	24310	48620	92378	167960	293930	497420	817190	1307504	0	0	-8007.04
12	1	11	66	286	1001	3003	8008	19448	43758	92378	184756	352716	646646	1144066	1961256	3268760	0	0	4366.7055
13	1	12	78	364	1365	4368	12376	31824	75582	167960	352716	705432	1352078	2496144	4457400	7726160	0	0	-1819.1335
14	1	13	91	455	1820	6188	18564	50388	125970	293930	646646	1352078	2704156	5200300	9657700	17383860	0	0	559.63205
15	1	14	105	560	2380	8568	27132	77520	203490	497420	1144066	2496144	5200300	10400600	20058300	37442160	0	0	-119.89948
16	1	15	120	680	3060	11628	38760	116280	319770	817190	1961256	4457400	9657700	20058300	40116600	77558760	0	0	15.983731
17	1	16	136	816	3876	15504	54264	170544	490314	1307504	3268760	7726160	17383860	37442160	77558760	155117520	0	0	-0.9988064

=MMULT(MINVERSE(A2:P17),Q2:Q17)

We can see the very poor accuracy of the linear system solution due to the round-off errors.

Linear Regression

Compute the linear regression for a given set of points $X(n \times m)$, $Y(n \times 1)$, where $n > m$. The solution is a vector of $1+m$ values containing the coefficients of regression; the constant term is the first value: $[a_0, a_1, a_2, \dots, a_m]$

$$\hat{y} = a_0 + \sum_{i=1}^m a_i x_i$$

Input and Output values have max 15 significant digits
Internal computing is performed with 30 significant digits
This task is performed by the program **reg_lin.exe** of this package.
Syntax command line

```
>>reg_lin filenameA.txt filenameB.txt filenameC.txt
```

FilenameA contains the element of the X matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of the Y matrix (one column) in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameC contains the coefficients of the regression in ascii tab-delimited. Numbers are always in exponential format with 15 significant digits

This program returns also the square regression coefficient R^2

Example: Perform the StRD Longley test

Standard Longley test

Model: Polynomial Class 7 Parameters (B0,B1,...,B7)

$$y = B_0 + B_1 \cdot x_1 + B_2 \cdot x_2 + B_3 \cdot x_3 + B_4 \cdot x_4 + B_5 \cdot x_5 + B_6 \cdot x_6 + e$$

Data: 1 Response Variable (y)
6 Predictor Variable (x)
16 Observations
Higher Level of Difficulty

Certified Regression Statistics

Parameter	Estimate
B0	-3482258.63459582
B1	15.0618722713733
B2	-0.358191792925910E-01
B3	-2.02022980381683
B4	-1.03322686717359
B5	-0.511041056535807E-01
B6	1829.15146461355

Tutorial for Bigmatrix.xla

Longley test data

Data:	y	x1	x2	x3	x4	x5	x6
	60323	83.0	234289	2356	1590	107608	1947
	61122	88.5	259426	2325	1456	108632	1948
	60171	88.2	258054	3682	1616	109773	1949
	61187	89.5	284599	3351	1650	110929	1950
	63221	96.2	328975	2099	3099	112075	1951
	63639	98.1	346999	1932	3594	113270	1952
	64989	99.0	365385	1870	3547	115094	1953
	63761	100.0	363112	3578	3350	116219	1954
	66019	101.2	397469	2904	3048	117388	1955
	67857	104.6	419180	2822	2857	118734	1956
	68169	108.4	442769	2936	2798	120445	1957
	66513	110.8	444546	4681	2637	121950	1958
	68655	112.6	482704	3813	2552	123366	1959
	69564	114.2	502601	3931	2514	125368	1960
	69331	115.7	518173	4806	2572	127852	1961
	70551	116.9	554894	4007	2827	130081	1962

Assume the matrix **X** in the range B5:G20 and **Y** in A5:A20

	A	B	C	D	E	F	G	H	I	J	K	L	M		
1	Linear Regression - StRD Longley test														
2	$y = B0 + B1*x1 + B2*x2 + B3*x3 + B4*x4 + B5*x5 + B6*x6 + e$														
3															
4	y	x1	x2	x3	x4	x5	x6				StRD certified	computed	Error	LRE	
5	60323	83	234289	2356	1590	107608	1947				B0	-3482258.63459582	-3482258.635	-1.02E-08	14.5
6	61122	88.5	259426	2325	1456	108632	1948				B1	15.0618722713733	15.06187227	-9.95E-14	14.2
7	60171	88.2	258054											0	15.0
8	61187	89.5	284599											-14	14.3
9	63221	96.2	328975											0	15.0
10	63639	98.1	346999											-16	14.1
11	64989	99	365385											0	15.0
12	63761	100	363112												15.0
13	66019	101.2	397469												14.6
14	67857	104.6	419180												14.1
15	68169	108.4	442769												14.1
16	66513	110.8	444546												
17	68655	112.6	482704												
18	69564	114.2	502601												
19	69331	115.7	518173												
20	70551	116.9	554894												
21															
22															
23															
24															
25															
26															

Big Matrix Manager

First cell: Longley!\$B\$5 Last cell: Longley!\$A\$20 Rows: 16 Columns: 6

Mat. A: Longley!\$B\$5 Mat. B: Longley!\$A\$5 Mat. C: Longley!\$K\$5

Operation: Linear Regression

Clock: Start: 18:50:08 End: 18:50:09 Elapsed: 00:00:01

Batch job ended in: 1.13 sec R^2 = 9.954790045772957E-001

The factor R^2 is shown in the matrix manager panel but this can be also get from the file in the temp directory of bigmatrix.xla

```
#JOBMSG000.TXT
Linear Regression solution X[7] found
determinant= 9.601927150313583E+031
R^2 coefficient= 9.954790045772957E-001
elapsed time: 0.010000 seconds
```

Certified values are put in range J5:J10 while the calculated value are in K5:K10

In report comparisons, it is common to use the Log Relative Error (LRE) that puts in evidence the decimal digits of accuracy. It cannot exceed 15 and higher values are obviously better
The absolute errors and LRE are computed ; they are shown in the last two columns

Tutorial for Bigmatrix.xla

	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	Linear Regression - StRD Longley test													
2	$y = B0 + B1 \cdot x1 + B2 \cdot x2 + B3 \cdot x3 + B4 \cdot x4 + B5 \cdot x5 + B6 \cdot x6 + e$													
3														
4	y	x1	x2	x3	x4	x5	x6			StRD certified	computed	Error	LRE	
5	60323	83	234289	2356	1590	107608	1947			B0	-3482258.63459582	-3482258.635	-1.02E-08	14.5
6	61122	88.5	259426	2325	1456	108632	1948			B1	15.0618722713733	15.06187227	-9.95E-14	14.2
7	60171	88.2	258054	3682	1616	109773	1949			B2	-0.0358191792925910	-0.035819179	0	15.0
8	61187	89.5	284599	3351	1650	110929	1950			B3	-2.02022980381683	-2.020229804	-1.02E-14	14.3
9	63221	96.2	328975	2099	3099	112075	1951			B4	-1.03322686717359	-1.033226867	0	15.0
10	63639	98.1	346999	1932	3594	113270	1952			B5	-0.0511041056535807	-0.051104106	-4.02E-16	14.1
11	64989	99	365385	1870	3547	115094	1953			B6	1829.15146461355	1829.151465	0	15.0
12	63761	100	363112	3578	3350	116219	1954							
13	66019	101.2	397469	2904	3048	117388	1955			R2	0.995479005			
14	67857	104.6	419180	2822	2857	118734	1956						max	15.0
15	68169	108.4	442769	2936	2798	120445	1957						average	14.6
16	66513	110.8	444546	4681	2637	121950	1958						min	14.1
17	68655	112.6	482704	3813	2552	123366	1959							
18	69564	114.2	502601	3931	2514	125368	1960							
19	69331	115.7	518173	4806	2572	127852	1961							
20	70551	116.9	554894	4007	2827	130081	1962							

As we can see, the average LRE for this linear regression test is 14.6

Polynomial Regression

Solve a polynomial regression for a given set of points $X(n \times 1)$, $Y(n \times 1)$

Where $n > m$

This task needs other extra parameter: degree of polynomial and (optional) intercept

If the degree of the polynomial is "d", the solution is a vector of $1+d$ values containing the coefficients of regression; the constant term (or intercept) is the first value: $[a_0, a_1, a_2, \dots, a_d]$

$$\hat{y} = a_0 + \sum_{i=1}^d a_i x^i$$

Input and Output values have max 15 significant digits

Internal computing is performed with 30 significant digits

This task is performed by the program **reg_lin.exe** of this package.

Syntax command line

```
>>reg_lin filenameA.txt filenameB.txt filenameC.txt degree intcpt
```

FilenameA contains the element of the X matrix in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameB contains the element of the Y matrix (one column) in ascii tab-delimited. Numbers can have any format (integer, decimal or exponential)

FilenameC contains the coefficients of the regression in ascii tab-delimited. Numbers are always in exponential format with 15 significant digits

Degree set the polynomial degree (default=1)

Intcpt (if present) force the intercept point $a_0 = \text{intcpt}$

This program returns also the square regression coefficient R^2

Example: Perform the StRD Filippelli test

Reference: Filippelli, A., NIST.

Data: 1 Response Variable (y)
 1 Predictor Variable (x)
 82 Observations
 Higher Level of Difficulty

Model: Polynomial Class
 11 Parameters (B0,B1,...,B10)

$$y = B_0 + B_1*x + B_2*(x^2) + \dots + B_9*(x^9) + B_{10}*(x^{10}) + e$$

Tutorial for Bigmatrix.xla

Certified Regression Statistics	
Parameter	Estimate
B0	-1467.48961422980
B1	-2772.17959193342
B2	-2316.37108160893
B3	-1127.97394098372
B4	-354.478233703349
B5	-75.1242017393757
B6	-10.8753180355343
B7	-1.06221498588947
B8	-0.670191154593408E-01
B9	-0.246781078275479E-02
B10	-0.402962525080404E-04

Filippelli test data

Data:	y	x		
	0.8116	-6.860120914	0.7678	-8.473531488
	0.9072	-4.324130045	0.7697	-8.247337057
	0.9052	-4.358625055	0.77	-7.971428747
	0.9039	-4.358426747	0.7749	-7.676129393
	0.8053	-6.955852379	0.7796	-7.352812702
	0.8377	-6.661145254	0.7897	-7.072065318
	0.8667	-6.355462942	0.8131	-6.774174009
	0.8809	-6.118102026	0.8498	-6.478861916
	0.7975	-7.115148017	0.8741	-6.159517513
	0.8162	-6.815308569	0.8061	-6.835647144
	0.8515	-6.519993057	0.846	-6.53165267
	0.8766	-6.204119983	0.8751	-6.224098421
	0.8885	-5.853871964	0.8856	-5.910094889
	0.8859	-6.109523091	0.8919	-5.598599459
	0.8959	-5.79832982	0.8934	-5.290645224
	0.8913	-5.482672118	0.894	-4.974284616
	0.8959	-5.171791386	0.8957	-4.64454848
	0.8971	-4.851705903	0.9047	-4.290560426
	0.9021	-4.517126416	0.9129	-3.885055584
	0.909	-4.143573228	0.9209	-3.408378962
	0.9139	-3.709075441	0.9219	-3.13200249
	0.9199	-3.499489089	0.7739	-8.726767166
	0.8692	-6.300769497	0.7681	-8.66695597
	0.8872	-5.953504836	0.7665	-8.511026475
	0.89	-5.642065153	0.7703	-8.165388579
	0.891	-5.031376979	0.7702	-7.886056648
	0.8977	-4.680685696	0.7761	-7.588043762
	0.9035	-4.329846955	0.7809	-7.283412422
	0.9078	-3.928486195	0.7961	-6.995678626
	0.7675	-8.56735134	0.8253	-6.691862621
	0.7705	-8.363211311	0.8602	-6.392544977
	0.7713	-8.107682739	0.8809	-6.067374056
	0.7736	-7.823908741	0.8301	-6.684029655
	0.7775	-7.522878745	0.8664	-6.378719832
	0.7841	-7.218819279	0.8834	-6.065855188
	0.7971	-6.920818754	0.8898	-5.752272167
	0.8329	-6.628932138	0.8964	-5.132414673
	0.8641	-6.323946875	0.8963	-4.811352704
	0.8804	-5.991399828	0.9074	-4.098269308
	0.7668	-8.781464495	0.9119	-3.66174277
	0.7633	-8.663140179	0.9228	-3.2644011

end of filippelli data

Filippelli test is very difficult for standard 32 bit arithmetic; for row data and without manipulation the accuracy loss is totally and the LRE is near the 0. Let's see how Bigmatrix works with this test

Assume the matrix **X** in the range B4:B86 and **Y** in A4:A86

Tutorial for Bigmatrix.xla

Before starting the job for polynomial regression you have to give two additional parameters: **Intercept** value and **Degree**.

Leave blank if you don't want to force the intercept.
Insert only 10 for the degree of Filippelli test.

	A	B	C	D	E	F	G	H	I
1	Linear Regression - StRD Filippelli test								
2	$y = B0 + B1 \cdot x + B2 \cdot (x^2) + \dots + B9 \cdot (x^9) + B10 \cdot (x^{10}) + e$								
3									
4	y	x			StRD certified	computed	Error	LRE	
5	0.8116	-6.860120914			B0	-1467.489614	-1467.489614	-1E-11	14.2
6	0.9072	-4.324130045							
7	0.9052	-4.358625055							
8	0.9039	-4.358426747							
9	0.8053	-6.955852379							
10	0.8377	-6.661145254							
11	0.8667	-6.355462942							
12	0.8809	-6.118102026							
13	0.7975	-7.115148017							
14	0.8162	-6.815308569							
15	0.8515	-6.519993057							
16	0.8766	-6.204119983							
17	0.8885	-5.853871964							
18	0.8859	-6.109523091							
19	0.8959	-5.79832982							
20	0.8913	-5.482672118							
21	0.8959	-5.171791386							
22	0.8971	-4.851705903							
23	0.9021	-4.517126416							
24	0.909	-4.143573228							
25	0.9139	-3.709075441							

BigMatrix after few seconds compute the 11 coefficients and others parameters: You can find this optional parameter in the

```
#JOBMSG000.TXT
Linear Regression solution X[11] found
determinant= 2.478562722656102E+037
R^2 coefficient= 9.967274161856201E-001
elapsed time: 0.130000 seconds
```

Certified values are put in range E5:E15 while the calculated value are in K5:K15
 In report comparisons, it is common to use the Log Relative Error (LRE) that puts in evidence the decimal digits of accuracy. It cannot exceed 15 and higher values are obviously better
 The absolute errors and LRE are computed ; they are shown in the last two columns

	A	B	C	D	E	F	G	H
1	Linear Regression - StRD Filippelli test							
2	$y = B_0 + B_1 \cdot x + B_2 \cdot (x^2) + \dots + B_9 \cdot (x^9) + B_{10} \cdot (x^{10}) + e$							
3								
4	y	x			StRD certified	computed	Error	LRE
5	0.8116	-6.860120914		B0	-1467.4896142298	-1467.4896142298	-1E-11	14.2
6	0.9072	-4.324130045		B1	-2772.17959193342	-2772.17959193342	0	15.0
7	0.9052	-4.358625055		B2	-2316.37108160893	-2316.37108160892	-1E-11	14.4
8	0.9039	-4.358426747		B3	-1127.97394098372	-1127.97394098371	-1E-11	14.1
9	0.8053	-6.955852379		B4	-354.478233703349	-354.478233703348	-1.023E-12	14.5
10	0.8377	-6.661145254		B5	-75.1242017393757	-75.1242017393756	-1.137E-13	14.8
11	0.8667	-6.355462942		B6	-10.8753180355343	-10.8753180355342	-1.013E-13	14.0
12	0.8809	-6.118102026		B7	-1.06221498588947	-1.06221498588946	-9.992E-15	14.0
13	0.7975	-7.115148017		B8	-0.0670191154593408	-0.0670191154593407	-1.11E-16	14.8
14	0.8162	-6.815308569		B9	-0.00246781078275479	-0.00246781078275478	-1.041E-17	14.4
15	0.8515	-6.519993057		B10	-4.02962525080404E-05	-4.02962525080403E-05	-1.016E-19	14.6
16	0.8766	-6.204119983						
17	0.8885	-5.853871964			R^2	0.996727416		
18	0.8859	-6.109523091					max	15.0
19	0.8959	-5.79832982					avg	14.4
20	0.8913	-5.482672118					min	14.0
21	0.8859	-5.174794388						

As we can see the test is passed with very good performance being the average LRE =14.4

Summary of Certification Results for Linear Regression

REG_LIN.EXE is a console program useful for linear and polynomial regression adapt to large matrices showing interesting performance. It is contained in the pack **BigMatrix.zip** and it is used by the Excel addin **Bigmatrix.xla**, but it can be used also alone.

Here, we report the NIST StRD test for the following linear regression functions:

LRE Results for Linear Regression (row data)			BigMatrix		
StRD Datasets	Level of difficulty	Model of class	min	avg	max
Norris	low	Linear	13.9	14.5	15
Pontius	low	Quadratic	15	15	15
NoInt1	Average	Linear	14.3	14.3	14.3
Filip	high	Polynomial	14.3	14.4	15
Longley	high	Multilinear	14.1	14.6	15
Wampler1	high	Polynomial	15	15	15
Wampler2	high	Polynomial	15	15	15
Wampler3	high	Polynomial	15	15	15
Wampler4	high	Polynomial	15	15	15
Wampler5	high	Polynomial	15	15	15

Note the general high precision REG_LIN.EXE, being the average LRE of about 14.8 for row data Not bad for only 60 Kb program, isn't it?

Example: polynomial regression

Calculate the 4th degree polynomial that approximate the f(x) of the following table

f(x)	x
85	1
83.4241	1.1
81.9136	1.2
80.4961	1.3
79.2016	1.4
78.0625	1.5
77.1136	1.6
76.3921	1.7
75.9376	1.8
75.7921	1.9
76	2
76.6081	2.1
77.6656	2.2
79.2241	2.3
81.3376	2.4
84.0625	2.5
87.4576	2.6
91.5841	2.7
96.5056	2.8

the regression model for this problem will be:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

BigMatrix compute and returns the coefficients in the same order

1. a₀
2. a₁
3. a₂
4. a₃
5. a₄

At the end, also the coefficient R² is returned

Note: The table was derived from the polynomial

$$f(x) = 100 - 12 x - 4 x^2 + x^4$$

This explains the perfect approximation (R =1)

The screenshot shows an Excel spreadsheet with columns A through N and rows 1 through 28. The data is organized as follows:

1	Polynomial regression			
2				
3	f(x)	x	term	coefficient
4	85	1	const	100
5	83.4241	1.1	x	-12
6	81.9136	1.2	x^2	-4
7	80.4961	1.3	x^3	1.02E-14
8	79.2016	1.4	x^4	1
9	78.0625	1.5		
10	77.1136	1.6	R^2	1
11	76.3921	1.7		
12	75.9376	1.8		
13	75.7921	1.9		
14	76	2		
15	76.6081	2.1		
16	77.6656	2.2		
17	79.2241	2.3		
18	81.3376	2.4		
19	84.0625	2.5		
20	87.4576	2.6		
21	91.5841	2.7		
22	96.5056	2.8		

Two dialog boxes are overlaid on the spreadsheet:

- Big Matrix Manager:** This dialog box shows the configuration for a polynomial regression. It includes fields for 'First cell' (A4), 'Last cell' (E22), 'Rows' (19), and 'Columns' (1). It lists three matrices (A, B, C) with their respective cell ranges. The 'Operation' is set to 'Polynomial Regres'. A 'Clock' section shows the start, end, and elapsed time. At the bottom, it displays 'Batch job ended in: 1.07 sec' and 'R^2 = 1.0000000000000000E+000'.
- Big Matrix manager - Poly regression:** This smaller dialog box is used to set the regression parameters. It has an 'Intercept' field (empty) and a 'Degree' field set to '4'. It includes 'OK', 'Help', and 'Exit' buttons.

Mop-up

This operation is useful to eliminate round-off error near zero.
Practically, if an element of a given matrix is absolute less than a fixed limit it is set to zero.

If $|a_{ij}| < \text{Limit} \implies a_{ij} = 0$

For large matrices this duty can be very heavy
This task operate on site: that is the input and output matrix are the same

Round

This operation is useful to round each element of a given matrix.
For large matrices this duty can be very heavy
Before starting it ask for the decimal number "dec" to round

If $\text{dec} > 0$ (zero), the element will be round at the decimal specified.
If $\text{dec} = 0$ (zero), the element will be round at the nearest integer.
If $\text{dec} < 0$ (zero), the dec integer digits of element will be rounded; the previous integer digits will set to 0 (zero).

Example

ROUND(2,15; 1) = 2,2
ROUND(2,149; 1) = 2,1
ROUND(-1,475; 2) = -1,48
ROUND(21,5; -1) = 20

This task operate on site: that is the input and output matrix are the same

Norma

This operation computes and returns the norma and a few important indices of a given matrix

Norma

The root of the sum of square of all elements

$$n = \sqrt{\sum_{i,j} a_{ij}^2}$$

Maximum absolute element

The max absolute value of all elements

$$\max(|a_{ij}|)$$

Minimum absolute element

The min absolute value; the min value is zero if at least one element is zero

Tutorial for Bigmatrix.xla

Diagonal form error

The norma without diagonal elements. It measure how distant is the given matrix from a diagonal matrix

$$n = \sqrt{\sum_{i \neq j}^n a_{ij}^2}$$

Note: The values are returns in the same sequence.

Example: The given matrix is the result of a diagonalization routine. We want to compute te average error of the elements out of the first diagonal.

The average error is $(0.00241) / 6 = 4.017E-4$

While the average relative error is:

$(\text{Diag. Err}) / (6 * \text{Norma}) = 1.093E-5$

	A	B	C	D	E
1	34	0.0023	-2E-04	Norma =	36.7287
2	5E-07	12	-4E-05	Max =	34
3	-4E-04	0.0003	-7	Min =	4.6E-07
4				Diag Err =	0.00241

Example : Stimte the Average Diagonalization Error (ADE) for the inverse of the 10x10 Tartaglia's matrix obtained by MINVERSE built-in function

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	1	1	1	1	1	1	1	1	1	1			
3	1	2	3	4	5	6	7	8	9	10			
4	1	3	6	10	15	21	28	36	45	55			
5	1	4	10	20	35	56	84	120	165	220			
6	1	5	15	35	70	126	210	330	495	715			
7	1	6	21	56	126	252	462	792	1287	2002			
8	1	7	28	84	210	462	924	1716	3003	5005			
9	1	8	36	120	330	792	1716	3432	6435	11440			
10	1	9	45	165	495	1287	3003	6435	12870	24310			
11	1	10	55	220	715	2002	5005	11440	24310	48620			
12													
13	1	3E-08	8E-08	2E-07	4E-07	7E-07	1E-06	2E-06	3E-06	4E-06		Norma	3.16227766
14	-1E-08	1	-1E-07	-3E-07	-6E-07	-1E-06	-2E-06	-4E-06	-6E-06	-9E-06		Max	1.000000081
15	6E-09	3E-08	1	2E-07	4E-07	8E-07	1E-06	2E-06	4E-06	6E-06		Min	6.66134E-15
16	-2E-09	-9E-09	-3E-08	1	-1E-07	-3E-07	-5E-07	-7E-07	-1E-06	-2E-06		Diag err	1.49899E-05
17	2E-10	1E-09	3E-09	4E-09	1	-5E-09	-2E-08	-8E-08	-2E-07	-6E-07			
18	4E-11	3E-10	1E-09	5E-09	1E-08	1	6E-08	1E-07	2E-07	5E-07			
19	-2E-11	-1E-10	-5E-10	-1E-09	-4E-09	-9E-09	1	-2E-08	-3E-08	-2E-07			
20	2E-12	1E-11	4E-11	2E-10	4E-10	1E-09	-2E-10	1	-3E-09	4E-08			
21	-5E-14	2E-13	8E-12	9E-13	2E-11	3E-11	6E-10	9E-10	1	-7E-09			
22	7E-15	-3E-14	-3E-13	-3E-13	-1E-12	2E-12	-5E-11	-1E-10	-2E-10	1			
23													
24													
25													
26													

Average Diagonalization Error = $(\text{Diag err}) / (n(n-1)) \cong 1.5E-5 / 90 \cong 1.67E-7$

References

Multiprecision arithmetic for C is derived by XNUMBERS v.2.3 by Leonardo Volpi
Compiled with the LCC-Win32 v.3.3 of Jacob Navia

LAPACK -- Linear Algebra PACKage 3.0, Updated: May 31, 2000

"Numerical Analysis" F. Sheid , McGraw-Hill Book Company, New-York, 1968

"Numerical Recipes in FORTRAN 77- The Art of Scientific Computing - 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software



© 2003, by Foxes Team
ITALY