

MATRIX MANUAL

LAURENCE R. TAYLOR
02/01/2012

The package **Matrix.sty** is a package designed to make typesetting simple matrices relatively simple. It can be downloaded from [here](#). Columns are justified and one can easily get at pieces of the matrix. It is somewhat Object-Oriented, or at least as much as T_EX can manage. It requires the packages **ifthen.sty** and **forloop.sty**.

1. DEFINING MATRICES

First you define a matrix and give it a name. You do this using the **NEW** command.

```
\NEW{foo}{r x c}{%  
a_11 a_12 ... a_1c  
a_21      ... a_2c  
.   
.   
.   
a_r1      ... a_rk  
}
```

The r and c entries should be positive integers, the x between them is just the lower case letter x . The row-column structure of the display is for convenience of the user. T_EX only needs to see $r \cdot c$ items separated by spaces. If you have spaces in an individual entry, enclose the entry in a brace pair but the spaces still need to be around the braces.

The **NEW** command is a bit picky about spaces in the matrix data. The opening brace must be followed by a % with the matrix starting immediately on the next line or by an entry with no space following the {. There can be more than one space between the items so you can use spaces and new lines to make your input look nice and easy to read. It is best if the final left brace is on a line by itself as well but at the very least there needs to be a space following the last entry before the closing }. T_EX code is perfectly OK inside the matrix entries.

The spaces around the $\{r x c\}$ are for your convenience, $\{rxc\}$ works just as well.

You have now defined an r by c matrix named **foo** and the amount of typing by you that it is needed is pretty minimal. It is also not difficult to get programmable matrix packages to produce output in a form that can be cut and pasted into the macro.

The **NEW** command also takes an optional argument for producing matrices whose entries can be produced by T_EX just knowing the position. The format is $\backslash\text{NEW}[\text{someMacro}]\{\text{foo}\}\{r x c\}\{\}$ where you need the last brace pair although anything inside it is ignored. You must write a *constructor macro*, say $\backslash\text{someMacro}$, as $\backslash\text{def}\backslash\text{someMacro}\#1\#2\{\dots\}$ where $\#1$ is the row position and $\#2$ is the column position of the entry being constructed. Notice that the optional argument entry is the string for the macro not $\backslash\text{someMacro}$. The actual name you choose is not important, it can be any legal T_EX macro name.

Here are some examples.

```
\def\alphaMacro#1#2{\alpha_{#1,#2}}
\NEW[\alphaMacro]{V}{4x8}{}
```

defines a matrix which can be displayed using methods from the next section as

```
 $\alpha_{1,1}$   $\alpha_{1,2}$   $\alpha_{1,3}$   $\alpha_{1,4}$   $\alpha_{1,5}$   $\alpha_{1,6}$   $\alpha_{1,7}$   $\alpha_{1,8}$ 
 $\alpha_{2,1}$   $\alpha_{2,2}$   $\alpha_{2,3}$   $\alpha_{2,4}$   $\alpha_{2,5}$   $\alpha_{2,6}$   $\alpha_{2,7}$   $\alpha_{2,8}$ 
 $\alpha_{3,1}$   $\alpha_{3,2}$   $\alpha_{3,3}$   $\alpha_{3,4}$   $\alpha_{3,5}$   $\alpha_{3,6}$   $\alpha_{3,7}$   $\alpha_{3,8}$ 
 $\alpha_{4,1}$   $\alpha_{4,2}$   $\alpha_{4,3}$   $\alpha_{4,4}$   $\alpha_{4,5}$   $\alpha_{4,6}$   $\alpha_{4,7}$   $\alpha_{4,8}$ 
```

and

```
\def\idMacro#1#2{\ifthenelse{\number#1=\number#2}{1}{0}}
\NEW[idMacro]{V}{5x5}{}
```

which can be used to display a 5x 5 identity matrix:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

2. METHODS

As an Object-Oriented imitator you might expect to see something like `\foo.display` to get the matrix. This doesn't quite work because periods are not permitted in T_EX names. To display a matrix named `foo` you use `$$\fooDisplay$`. If you want brackets, use `$$\left[\fooDisplay\right]$$`.

Here are the methods which can be used on a defined matrix `foo`

- (10) `\fooRows`
which returns the number of rows in the matrix `foo`.
- (11) `\fooCols`
which returns the number of columns in the matrix `foo`.
- (12) `\fooEntry(i,j)`
which returns the i j^{th} entry of the matrix `foo`.

The nemumbered labels are so we may refer to these methods later.

There are methods which output a `\begin{matrix} ... \end{matrix}`

- (21) `\fooDisplay`
which displays the entire matrix `foo`.
- (22) `\fooRowDisplay{i}`
which displays the i^{th} row of the matrix `foo`.
- (23) `\fooColDisplay{j}`
which displays the j^{th} column of the matrix `foo`.
- (24) `\fooSubmatrixDisplay}{i}{j}{m}{n}$`
which displays the sub-matrix of `foo` starting in position (i,j) and going down to position (m,n) .

- (25) `\fooTransposeDisplay`
which displays the transpose of `foo`.
- (26) `\fooCofactorDisplay{r}{c}`
which displays the cofactor of `foo` with row `r` and column `c` deleted.

There are also two commands which display matrices:

- (31) `\NEWDisplay{foo}{r x c}{ ... }`
which defines and displays the entire matrix `foo`.
- (32) `\DISPLAY{r x c}{ ... }` which just displays its data immediately and stores nothing.

All methods which set up a `\begin{matrix} ... \end{matrix}` take an optional variable (with the usual L^AT_EX format). This variable is the name of a macro, defined by the user. This *formatting macro* is applied to every element in the display.

You are free to name your macro any legal T_EX macro name and the definition is `\def\mymacro#1#2#3{ ... }` where `#1` is the row number, `#2` is the column number and `#3` is the entry. The default is just `\hfill #3` which right justifies every entry to which it is applied. If you need right justification in your own macros, `\RJ` can be used.

With matrix `A` as defined above,

`\def\LJ#1#2#3{#3\hfill}` produces
$$\left[\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ -11 & 6 & 7 & 8 & 9 \\ 2 & 5 & -7 & 8 & 19 \end{array} \right]$$

Not very pretty, but the entries are left justified.

More useful is something like

`\def\mymacro#1#2#3{\hbox to 20pt{\hfill $#3$}}`
which keeps all columns 20 pt's in width and also right justified.

The commands `\NEWDisplay` and `\DISPLAY` also take an optional variable which has names of macros. There is an issue as to whether the optional macro is like a `\NEW` optional macro and constructs entries or is like a `Display` optional macro and formats the display.

The solution is to allow all options: neither, either or both type of macros. If there is no optional variable then there must be matrix data and the commands display the matrix (and `\NEWDisplay` stores the matrix). If there is an optional variable, there may be one or two: if there are two they should be separated by a comma. First, the number of arguments in each macro is determined. Anything with three arguments is considered to be a formatting macro and anything with two arguments is considered to be a constructor macro. This has the bonus features that if there is one macro it can be of either sort and if there are two macros they can be in either order.

If there is a constructor macro, it is used to construct the entries and hence any data in the last argument is ignored. If there is a formatting macro, it is used to format the display.

Here are some examples. Recall the constructor macro

`\def\alphaMacro#1#2{\alpha_{#1,#2}}`

and the formatting macro

`\def\green#1#2#3{\hfill \textcolor{green}{#3}}`

from above.

Then `\left[\NEWDisplay[green,alphaMacro]{A}{3x5}{}\right]` produces

$$\left[\begin{array}{ccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array} \right]$$

as does `\left[\NEWDisplay[alphaMacro,green]{A}{3x5}{}\right]`

The matrix is saved as `A` so `\left[\ASubmatrixDisplay{2}{2}{3}{4}\right]` produces

$$\begin{bmatrix} \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \end{bmatrix}$$

Notice the formatting from `\NEWDisplay` did not come with `A`. To make the display green use `\left[\ASubmatrixDisplay[green]{2}{2}{3}{4}\right]`

$$\begin{bmatrix} \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} \end{bmatrix}$$

`\left[\NEWDisplay[alphaMacro]{A}{3x5}{}\right]` produces

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

`\left[\NEWDisplay[green]{A}{3x5}{%`

1 2 3 4 5

a b c d e

9 8 7 6 5

}\right]

produces

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ a & b & c & d & e \\ 9 & 8 & 7 & 6 & 5 \end{bmatrix}$$

And of course with no optional variable

`\left[\NEWDisplay{A}{3x5}{%`

1 2 3 4 5

a b c d e

9 8 7 6 5

}\right]

produces

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ a & b & c & d & e \\ 9 & 8 & 7 & 6 & 5 \end{bmatrix}$$

Finally `\left[\NEWDisplay[alphaMacro]{A}{3x5}{%`

1 2 3 4 5

a b c d e

9 8 7 6 5

}\right]

ignores the matrix data and produces

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

All of the `\NEWDisplay`'s above can be replaced by `\DISPLAY` with the `{A}` deleted and the outputs will look the same, but nothing will be stored for future use like the `\ASubmatrixDisplay` above.

3. COPY METHODS

In addition to `NEW` and `\NEWDisplay` there are methods which make new matrices from old. The list here is as follows.

(41) `\fooCopy{newName}`

which copies the matrix `foo` into a new matrix `newName`.

- (42) `\fooRowCopy{newName}{i}`
which copies the i^{th} row of the matrix `foo` into a new matrix `newName`.
- (43) `\fooColCopy{newName}{j}`
which copies the j^{th} column of the matrix `foo` into a new matrix `newName`.
- (44) `\fooSubmatrixCopy{newName}{i}{j}{m}{n}`
which copies the sub-matrix of `foo` starting in position (i, j) and going down to position (m, n) into a new matrix `newName`.
- (45) `\fooTransposeCopy{newName}`
which copies the transpose of `foo` into a new matrix `newName`.
- (46) `\fooCofactorCopy{newName}{r}{c}`
which copies the cofactor of `foo` with row `r` and column `c` deleted into the matrix `newName`.

As a general rule, `newName` should be different than `foo`. Most of the time the copy will work as expected except for `TransposeCopy` and `CofactorCopy` which may have problems.

Copy methods do not take an optional variable.

The method `\fooMethodDisplay ...` is much the same as `\fooMethodCopy{newName}... \newNameDisplay` except that in the latter case, the result is stored in a new matrix.

Here are some examples using the last matrix `A` from above. The same matrix is produced two ways, once with the `Display` method and once with the `Copy` method and displaying the copy. The codes are displayed side by side and the results are displayed under the code.

The Display methods: (21) & (41)

$\left[\begin{array}{ccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array} \right]$	$\backslash\text{ACopy}\{B\} \left[\begin{array}{ccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array} \right]$
$\left[\begin{array}{ccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array} \right]$	$\backslash\text{ACopy}\{B\} \left[\begin{array}{ccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array} \right]$

The Row methods: (22) & (42)

$\left[\begin{array}{ccccc} \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \end{array} \right]$	$\backslash\text{ARowCopy}\{B\}2 \left[\begin{array}{ccccc} \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \end{array} \right]$
$\left[\begin{array}{ccccc} \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \end{array} \right]$	$\backslash\text{ARowCopy}\{B\}2 \left[\begin{array}{ccccc} \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \end{array} \right]$

The Col methods: (23) & (43)

$\left[\begin{array}{c} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{array} \right]$	$\backslash\text{AColCopy}\{B\}3 \left[\begin{array}{c} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{array} \right]$
$\left[\begin{array}{c} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{array} \right]$	$\backslash\text{AColCopy}\{B\}3 \left[\begin{array}{c} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{array} \right]$

$$\begin{bmatrix} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{1,3} \\ \alpha_{2,3} \\ \alpha_{3,3} \end{bmatrix}$$

The Submatrix methods: (24) & (44)

`\left[\backslash\text{ASubmatrixDisplay}`
`{1}{2}{3}{5}\right]`\$

$$\begin{bmatrix} \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

`\SubmatrixCopy{B}{1}{2}{3}{5}`
`\left[\backslash\text{BDisplay}\right]`\$

$$\begin{bmatrix} \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

`\left[\backslash\text{ASubmatrixDisplay}`
`{1}{2}{3}{5}\right]``[green]`\$

$$\begin{bmatrix} \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

`\SubmatrixCopy{B}{1}{2}{3}{5}`
`\left[\backslash\text{BDisplay}`
`[green]\right]`\$

$$\begin{bmatrix} \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{bmatrix}$$

The Transpose methods: (25) & (45)

`\left[\backslash\text{ATransposeDisplay}\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\ \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} \end{bmatrix}$$

`\ATransposeCopy{B}` `\left[\backslash\text{BDisplay}\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\ \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} \end{bmatrix}$$

`\left[\backslash\text{ATransposeDisplay}`
`[green]\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\ \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} \end{bmatrix}$$

`\ATransposeCopy{B}`
`\left[\backslash\text{BDisplay}`
`[green]\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\ \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\ \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\ \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} \end{bmatrix}$$

The Cofactor methods: (26) & (46)

`\left[\backslash\text{ACofactorDisplay}`
`{2}{4}\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,5} \end{bmatrix}$$

`\ACofactorCopy{B}{2}{4}`
`\left[\backslash\text{BDisplay}\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,5} \end{bmatrix}$$

`\left[\backslash\text{ACofactorDisplay}`
`[green]{2}{4}\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,5} \end{bmatrix}$$

`\ACofactorCopy{B}{2}{4}`
`\left[\backslash\text{BDisplay}`
`[green]\right]`\$

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,5} \end{bmatrix}$$

4. STACKS

One of the more annoying situations in \TeX ing matrices occurs when you have entered two matrices and now you would like to display them side by side or one over the other. This involves

careful cutting and pasting or trying to line up boxes. This packages has a method for making a new matrix by stacking together previously defined matrices.

```
The method is \NEWStack{foo}{r x c}{%
  name_{11} name_{12} . . . name_{1c}
  .
  .
  name_{r1} name_{r2} . . . name_{rc}
}
```

where the `name_{ij}` are the names of previously defined matrices.

Some examples should make the use and potentials of the method clear. Suppose given three matrices

$$\begin{array}{cc}
 \begin{array}{ccccc}
 \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\
 \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\
 \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \\
 1 & 2 & 3 & 4 & 5 \\
 19 & 7 & 14 & 32 & 6 \\
 -1 & 9 & 8 & 7 & 6
 \end{array} &
 \begin{array}{ccccc}
 a & b & c & d & e & f \\
 g & h & i & j & k & l \\
 m & n & o & p & q & r
 \end{array}
 \end{array}$$

Horizontal stack of A and B: Code \rightarrow `\NEWStack{V}{1x2}{A B}`
 $\left[\right]$

$$\left[\begin{array}{cccccc}
 \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & a & b & c & d & e & f \\
 \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & g & h & i & j & k & l \\
 \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & m & n & o & p & q & r
 \end{array} \right]$$

Vertical stack of A and C: Code \rightarrow `\NEWStack{W}{2x1}{A B}`
 $\left[\right]$

$$\left[\begin{array}{ccccc}
 \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\
 \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\
 \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \\
 1 & 2 & 3 & 4 & 5 \\
 19 & 7 & 14 & 32 & 6 \\
 -1 & 9 & 8 & 7 & 6
 \end{array} \right]$$

Vertical stack of A and C with a 20pt fixed width:

```
Code  $\rightarrow$  \def\fixedwidth#1#2#3{
  \hbox to 20pt{\hfill$#3$}
  $\left[\WDisplay[fixedwidth]\right]$\}
```

$$\left[\begin{array}{ccccc}
 \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\
 \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\
 \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \\
 1 & 2 & 3 & 4 & 5 \\
 19 & 7 & 14 & 32 & 6 \\
 -1 & 9 & 8 & 7 & 6
 \end{array} \right]$$

Vertical stack of A and C with a 20pt fixed width and entries centered:

```
Code  $\rightarrow$  \def\centeredfixedwidth#1#2#3{
  \hbox to 20pt{\hfill$#3$\hfill}
  $\left[\WDisplay[centeredfixedwidth]\right]$\}
```

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \\ 1 & 2 & 3 & 4 & 5 \\ 19 & 7 & 14 & 32 & 6 \\ -1 & 9 & 8 & 7 & 6 \end{bmatrix}$$

Next is an example of a 2x2 stack:

```
\NEWStack{X}{2 x 2}{%
```

```
A C
```

```
C A
```

```
}
```

produces

$$\begin{array}{cccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & 1 & 2 & 3 & 4 & 5 \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & 19 & 7 & 14 & 32 & 6 \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & -1 & 9 & 8 & 7 & 6 \\ 1 & 2 & 3 & 4 & 5 & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ 19 & 7 & 14 & 32 & 6 & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ -1 & 9 & 8 & 7 & 6 & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array}$$

The matrices in a given row need not be the same width as those in a previous row as long as they have the same total column length. All matrices on a given row in the stack should have the same number of rows.

An example of a 2x2 stack with columns of different width is given by

```
\NEWStack{Y}{2 x 2}{%
```

```
A B
```

```
B A
```

```
}
```

which produces

$$\begin{array}{cccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & a & b & c & d & e & f \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & g & h & i & j & k & l \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & m & n & o & p & q & r \\ a & b & c & d & e & f & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ g & h & i & j & k & l & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ m & n & o & p & q & r & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array}$$

Here is a coloring of the same matrix making the difference in column width clearer.

$$\begin{array}{cccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & a & b & c & d & e & f \\ \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & g & h & i & j & k & l \\ \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & m & n & o & p & q & r \\ a & b & c & d & e & f & \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} \\ g & h & i & j & k & l & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} \\ m & n & o & p & q & r & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} \end{array}$$

Here is the code that produced the above effect.

```
\gdef\green#1#2#3{\hbox to 20pt{\hfill$\textcolor{green}{#3}$\hfill}}
```

```
\def\red#1#2#3{\hbox to 20pt{\hfill$\textcolor{red}{#3}$\hfill}}
```

```
\def\greenArea#1#2#3{%
```

```
\inSubmatrix{#1}{#2}{1}{1}{\ARows}{\ACols}{green}{greenAreaA}{#3}
```

```
}
```

```
\def\greenAreaA#1#2#3{%
```

```

\inSubmatrix{#1}{#2}{4}{7}{6}{11}{green}{redArea}{#3}
}

\def\redArea#1#2#3{%
\inSubmatrix{#1}{#2}{4}{1}{7}{\BCols}{red}{redAreaA}{#3}
}

\def\redAreaA#1#2#3{%
\inSubmatrix{#1}{#2}{1}{6}{\BRows}{11}{red}{RJ}{#3}
}

```

$\$ \backslash YDisplay[greenArea] \$$

The output of a `Stack` method is just a matrix and has all the methods any other matrix has. You can display the transpose using

$\backslash \left[\backslash YTransposeDisplay \right] \backslash$

$$\begin{bmatrix}
 \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & a & g & m \\
 \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & b & h & n \\
 \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & c & i & o \\
 \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & d & j & p \\
 \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} & e & k & q \\
 a & g & m & f & l & r \\
 b & h & n & \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\
 c & i & o & \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\
 d & j & p & \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\
 e & k & q & \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\
 f & l & r & \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5}
 \end{bmatrix}$$

Or you can `TransposeCopy` and then `Display`.

$\backslash YTransposeCopy\{Z\}$
 $\backslash \left[\backslash ZDisplay \right] \backslash$

$$\begin{bmatrix}
 \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} & a & g & m \\
 \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} & b & h & n \\
 \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} & c & i & o \\
 \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} & d & j & p \\
 \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5} & e & k & q \\
 a & g & m & f & l & r \\
 b & h & n & \alpha_{1,1} & \alpha_{2,1} & \alpha_{3,1} \\
 c & i & o & \alpha_{1,2} & \alpha_{2,2} & \alpha_{3,2} \\
 d & j & p & \alpha_{1,3} & \alpha_{2,3} & \alpha_{3,3} \\
 e & k & q & \alpha_{1,4} & \alpha_{2,4} & \alpha_{3,4} \\
 f & l & r & \alpha_{1,5} & \alpha_{2,5} & \alpha_{3,5}
 \end{bmatrix}$$

These two approaches produce the same output but the second has also created and stored the matrix `Z` for future use.

5. OTHER STUFF

There is another macro, unrelated to matrix manipulations but often useful for displays, `\vStrut` which takes one optional variable and two others. The required two variables are non-negative integers. The first is a depth and the second a height. The command constructs a \TeX -rule with a depth of the first required variable in pts and a height of the second. The optional variable sets

the width of the rule and if set to be non-zero, you can see the rule and adjust it accordingly. It can be used to make matrix entries appear taller or deeper than they really are.

Here is a big matrix defined and displayed using `\NEWDisplay{Big}{26 x 16}{...}` where the space separated entries were generated by computer. It is larger than most matrices you are ever likely to want to display, but it does show that L^AT_EX and this macro package are up to the job if required.

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.46	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.80	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.21	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.10	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.76	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.10	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.46	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.09	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.71	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.32	0.76	0.93
0.16	0.45	0.00	0.67	0.85	0.63	0.14	0.82	0.88	0.37	0.39	0.55	0.73	0.82	0.85	0.77
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.45	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.97	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.12	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.72	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.51	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.62	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.14	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.35	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.46	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.94	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.83	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.56	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.25	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.37	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.44	0.09	0.40

By adding a few struts the brackets go a bit higher and a bit lower.

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.46	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.80	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.21	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.10	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.76	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.10	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.46	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.09	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.71	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.32	0.76	0.93
0.16	0.45	0.00	0.67	0.85	0.63	0.14	0.82	0.88	0.37	0.39	0.55	0.73	0.82	0.85	0.77
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.45	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.97	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.12	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.72	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.51	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.62	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.14	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.35	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.46	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.94	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.83	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.56	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.25	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.37	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.44	0.09	0.40

The code is

```
\def\Bequal#1#2#3{\hfill\vStrut{0}{10}#3}
\def\Cequal#1#2#3{\hfill\vStrut{5}{0}#3}
\def\Dequal#1#2#3{\atPoint{#1}{#2}{\BigRows}{\BigCols}{Cequal}{RJ}{#3}}
```

```

\def\struts#1#2#3{%
\atPoint{#1}{#2}{1}{1}{Bequal}{Dequal}{#3}%
}
{\tiny\[\left[\BigDisplay[struts] \right]\]}

```

`\vStrut` can also be used instead of `\noalign{ \vskip ... }` since you can not access the `\cr`'s in the matrix display to add the `\noalign`'s.

The macro `\atPoint#1#2#3#4#5#6#7` used above regards the first two arguments as a location in a matrix and it also regards the third and fourth arguments as a location in a matrix. If they are the same location, the macro named in argument 5 is done, otherwise it is the macro named in argument 6 that is done. Both macros are formatting macros and so have three arguments. Arguments 1 and 2 of `\atPoint` are arguments 1 and 2 of the formatting macro and argument 3 of the formatting macro is argument 7 of `\atPoint`.

There is a similar command, `\inSubmatrix#1#2#3#4#5#6#7#8#9` with a similar structure. Arguments 1 and 2 are a location, arguments (3,4) and arguments (5,6) are the upper left and lower right point in the matrix. If the point is in this submatrix, do the macro named by 7, otherwise do the one named by 8. These are formatting macros and the first two arguments of the formatting macros are the first two arguments here. The third argument of the formatting macros is the ninth argument here.

For example to color a submatrix of the big matrix `Big`, use `\left[\BigDisplay[blueRegion]\right]`.

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.46	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.80	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.21	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.10	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.76	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.10	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.46	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.09	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.71	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.32	0.76	0.93
0.16	0.45	0.00	0.67	0.85	0.63	0.14	0.82	0.88	0.37	0.39	0.55	0.73	0.82	0.85	0.77
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.45	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.97	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.12	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.72	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.51	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.62	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.14	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.35	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.46	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.94	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.83	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.56	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.25	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.37	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.44	0.09	0.40

where `\blueRegion` is defined by

```

\def\blueRegion#1#2#3{\inSubmatrix{#1}{#2}{5}{8}{22}{13}{blue}{RJ}{#3}}

```

and the definition of `\blue#1#2#3` can be safely left to the reader. Note the blue colored entries start in row 5, column 8 and continue until row 22, column 13.

There is no need for an `inRow` or `inCol` macro since these are examples of `inSubmatrix`.

Just for fun, here are row 11 and column 14 of `Big` colored red.

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.46	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.80	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.21	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.10	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.76	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.10	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.46	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.09	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.71	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.32	0.76	0.93
0.16	0.45	0.00	0.67	0.85	0.63	0.14	0.82	0.88	0.37	0.39	0.55	0.73	0.82	0.85	0.77
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.45	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.97	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.12	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.72	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.51	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.62	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.14	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.35	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.46	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.94	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.83	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.56	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.25	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.37	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.44	0.09	0.40

Here is the cofactor:

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.76	0.93
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.09	0.40

As a more challenging exercise, color the row red, the column green, except for the common point which is to be blue. Make the other entries gray.

0.82	0.94	0.71	0.15	0.80	0.46	0.70	0.98	0.49	0.67	0.41	0.99	0.83	0.46	0.27	0.16
0.73	0.33	0.83	0.87	0.37	0.22	0.65	0.65	0.32	0.50	0.48	0.14	0.86	0.80	0.34	0.79
0.39	0.45	0.66	0.94	0.92	0.43	0.77	0.98	0.70	0.07	0.85	0.84	0.50	0.21	0.64	0.99
0.30	0.00	0.11	0.13	0.37	0.84	0.64	0.28	0.61	0.21	0.08	0.26	0.29	0.10	0.99	0.32
0.03	0.56	0.32	0.44	0.18	0.20	0.12	0.91	0.67	0.06	0.43	0.07	0.58	0.76	0.83	0.11
0.47	0.97	0.85	0.84	0.63	0.53	0.42	0.67	0.38	0.21	0.64	0.34	0.70	0.10	0.61	0.43
0.91	0.56	0.76	0.06	0.51	0.49	0.60	0.47	0.19	0.33	0.48	0.00	0.93	0.46	0.11	0.51
0.42	0.41	0.06	0.91	0.66	0.68	0.29	0.29	0.36	0.48	0.01	0.70	0.12	0.09	0.06	0.73
0.63	0.12	0.26	0.49	0.17	0.78	0.70	0.52	0.50	0.71	0.09	0.32	0.82	0.71	0.82	0.42
0.82	0.71	0.84	0.40	0.03	0.75	0.06	0.06	0.13	0.67	0.43	0.60	0.81	0.32	0.76	0.93
0.16	0.45	0.00	0.67	0.85	0.63	0.14	0.82	0.88	0.37	0.39	0.55	0.73	0.82	0.85	0.77
0.28	0.04	0.07	0.11	0.64	0.54	0.56	0.48	0.02	0.20	0.21	0.48	0.10	0.45	0.46	0.76
0.56	0.27	0.33	0.16	0.88	0.97	0.95	0.66	0.49	0.98	0.31	0.07	0.81	0.97	0.35	0.57
0.92	0.99	0.13	0.70	0.44	0.56	0.09	0.85	0.63	0.62	0.10	0.71	0.13	0.12	0.51	0.99
0.90	0.27	0.56	0.15	0.98	0.75	0.80	0.81	0.22	0.49	0.15	0.91	0.54	0.72	0.15	0.81
0.37	0.85	0.17	0.89	0.92	0.18	0.11	0.14	0.57	0.83	0.84	0.68	0.11	0.51	0.10	0.28
0.06	0.38	0.35	0.90	0.13	0.65	0.86	0.39	0.85	0.74	0.74	0.09	0.18	0.62	0.42	0.34
0.88	0.02	0.41	0.11	0.94	0.96	0.94	0.87	0.75	0.40	0.40	0.44	0.51	0.14	0.96	0.22
0.69	0.41	0.53	0.84	0.02	0.13	0.78	0.97	0.20	0.75	0.36	0.74	0.30	0.35	0.12	0.15
0.87	0.06	0.04	0.30	0.39	0.60	0.71	0.69	0.87	0.76	0.35	0.48	0.63	0.46	0.15	0.05
0.02	0.64	0.70	0.08	0.52	0.50	0.34	0.65	0.53	0.91	0.21	0.74	0.15	0.94	0.64	0.71
0.54	0.86	0.09	0.12	0.21	0.15	0.20	0.95	0.54	0.62	0.94	0.28	0.28	0.83	0.18	0.66
0.90	0.70	0.07	0.55	0.85	0.33	0.11	0.37	0.01	0.70	0.76	0.18	0.12	0.56	0.17	0.75
0.97	0.41	0.97	0.30	0.20	0.90	0.09	0.97	0.55	0.08	0.35	0.92	0.79	0.25	0.29	0.27
0.60	0.38	0.17	0.59	0.72	0.93	0.84	0.13	0.46	0.38	0.48	0.30	0.00	0.37	0.34	0.98
0.31	0.79	0.93	0.49	0.19	0.44	0.02	0.34	0.81	0.08	0.63	0.37	0.02	0.44	0.09	0.40