Using the **kbordermatrix** package

K. Border
kcb@caltech.edu

January 26, 2003
Rev. July 13, 2012

The **kbordermatrix** package provides the `\kbordermatrix` command, which can be used to make displays such as this, taken from my working paper, *Reduced form auctions revisited* [1],

$$\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{indices} & (1{\cdot}11) & (1{\cdot}12) & (1{\cdot}22) & (2{\cdot}11) & (2{\cdot}12) & (2{\cdot}22) & & & \\
(1) & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & 0 & 0 & 0 & p_{1\cdot11} & = & P_1 \\
(2) & 0 & 0 & 0 & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & p_{1\cdot12} & = & P_2 \\
\hline
(111) & 3 & 0 & 0 & 0 & 0 & 0 & p_{1\cdot22} & \le & 1 \\
(112) & 0 & 2 & 0 & 1 & 0 & 0 & p_{2\cdot11} & \le & 1 \\
(122) & 0 & 0 & 1 & 0 & 2 & 0 & p_{2\cdot12} & \le & 1 \\
(222) & 0 & 0 & 0 & 0 & 0 & 3 & p_{2\cdot22} & \le & 1
\end{array}$$

Note that the array on the left has a matrix surrounded by brackets with an additional row on top and column on the left. This is what Knuth [2] calls a bordered matrix. He even provides a `\bordermatrix` command, which is still available in LaTeX, but is not documented. I tweaked his command to work better with LaTeX, and renamed it to avoid confusion.

Note that the border row and column are typeset in `\scriptstyle`, but the most important thing about this display is the the rows of the left-hand matrix align with the rows of the right-hand matrix, and the top row is seemingly ignored. To convince yourself that this is so, let's "open up" the display with the LaTeX command
`\renewcommand{\arraystretch}{2}`

$$\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{indices} & (1{\cdot}11) & (1{\cdot}12) & (1{\cdot}22) & (2{\cdot}11) & (2{\cdot}12) & (2{\cdot}22) & & & \\
(1) & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & 0 & 0 & 0 & p_{1\cdot11} & = & P_1 \\
(2) & 0 & 0 & 0 & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & p_{1\cdot12} & = & P_2 \\
\hline
(111) & 3 & 0 & 0 & 0 & 0 & 0 & p_{1\cdot22} & \le & 1 \\
(112) & 0 & 2 & 0 & 1 & 0 & 0 & p_{2\cdot11} & \le & 1 \\
(122) & 0 & 0 & 1 & 0 & 2 & 0 & p_{2\cdot12} & \le & 1 \\
(222) & 0 & 0 & 0 & 0 & 0 & 3 & p_{2\cdot22} & \le & 1
\end{array}$$

I also have an experimental version that will put the labels on the right and bottom. If you ask, I will make it available. I might even try to make a version that lets you specify the layout, but I am content with top and left.

Known issues

The command works by first typesetting the array without delimiters and then disassembling the resulting box to make some measurements, and finally overlaying the delimiters on the typeset box. Since the first row and first column are typeset differently, it is not surprising that if there are not at least two rows and two columns that the command may behave strangely, but I wouldn't call this a bug. For instance, an empty array, `\kbordermatrix{}`, reverses the delimiters:

$$] \, . \qquad [$$

(Don't ask me why.) A one-element array, e.g., `\kbordermatrix{x}`, yields:

$$\begin{array}{c} x \\ [] \end{array},$$

which is what it should do, if you think about it.

If you have really tall entries, such as $\sum_{n=1}^{\infty} \frac{1}{2^n}$, you will probably need to open up the array in order to make enough room for the rows to line up. That is what the `\arraystretch` parameter is for.

Using it

First you must put

```
\usepackage{kbordermatrix}
```

in your document preamble, but you knew that. Second, `\kbordermatrix` *only works in equations.*

Here are some simple examples of use. Start by replacing `\left[\begin{array}{...}` by `\kbordermatrix{`. Note that there is no column specifier. You can make as many columns as you like, but they will all be centered. To finish, instead of `\end{array}\right]`, just type `}`. Like this:

```
\kbordermatrix{\mbox{indices}&1&2&3&4\\
1&M_{1,1}&M_{1,2}&M_{1,3}&M_{1,4}\\
2&M_{2,1}&M_{2,2}&M_{2,3}&M_{2,4}
}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \kbordermatrix{\mbox{indices}&1&2&3&4\\ 1&M_{1,1}&M_{1,2}&M_{1,3}&M_{1,4}\\ 2&M_{2,1}&M_{2,2}&M_{2,3}&M_{2,4}}$$

Without a column specifier, how do we put vertical rules in? Unfortunately, you have to put a `\vrule` in its own column in each row.

```
\kbordermatrix{\mbox{indices}&
1&2&\vrule&3&4\\
1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&M_{1,4}\\
2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&M_{2,4}
}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \kbordermatrix{\mbox{indices}&1&2&\vrule&3&4\\ 1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&M_{1,4}\\ 2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&M_{2,4}}$$

Note the gap in the bar below the first row. This emphasizes that the first row is different. (It also means that I couldn't figure out a nice way to get rid of it and still add extra space below the first row.)

If you don't like all the horizontal space around the vertical rule, you can get rid of it using plain TEX's `\omit` command.

```
\kbordermatrix{\mbox{indices}&
1&2&\omit\vrule&3&4\\
1&M_{1,1}&M_{1,2}&\omit\vrule&
M_{1,3}&M_{1,4}\\
2&M_{2,1}&M_{2,2}&\omit\vrule&
M_{2,3}&M_{2,4}}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \kbordermatrix{\mbox{indices}&1&2&\omit\vrule&3&4\\ 1&M_{1,1}&M_{1,2}&\omit\vrule&M_{1,3}&M_{1,4}\\ 2&M_{2,1}&M_{2,2}&\omit\vrule&M_{2,3}&M_{2,4}}$$

If you think the vertical rule in the first row is too tall, you can shorten it using plain TEX's `height` command.

```
\kbordermatrix{\mbox{indices}&
1&2&\omit\vrule height 1ex&3&4\\
1&M_{1,1}&M_{1,2}&\omit\vrule&
M_{1,3}&M_{1,4}\\
2&M_{2,1}&M_{2,2}&\omit\vrule&
M_{2,3}&M_{2,4}}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \kbordermatrix{\mbox{indices}&1&2&\omit\vrule height 1ex&3&4\\ 1&M_{1,1}&M_{1,2}&\omit\vrule&M_{1,3}&M_{1,4}\\ 2&M_{2,1}&M_{2,2}&\omit\vrule&M_{2,3}&M_{2,4}}$$

Horizontal lines works as you might expect.

```
\kbordermatrix{\mbox{indices}&
1&2&\vrule&3&4\\
1&M_{1,1}&M_{1,2}&\vrule&
M_{1,3}&M_{1,4}\\\hline
2&M_{2,1}&M_{2,2}&\vrule&
M_{2,3}&M_{2,4}
}
```

$$\kbordermatrix{\mbox{indices}&1&2&\vrule&3&4\\ 1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&M_{1,4}\\\hline 2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&M_{2,4}}$$

Notice how the `\hline` cuts across all the columns, but it doesn't connect to the closing bracket. I am not sure I like this behavior, but you can use `\cline`.

```
\kbordermatrix{\mbox{indices}&1&2&
\vrule&3&4\\
1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&
M_{1,4}\\\cline{1-1} \cline{2-6}
2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&
M_{2,4}}
```

$$\kbordermatrix{\mbox{indices}&1&2&\vrule&3&4\\ 1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&M_{1,4}\\\cline{1-1} \cline{2-6} 2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&M_{2,4}}$$

```
\kbordermatrix{\mbox{indices}&1&2&
\vrule&3&4\\
1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&
M_{1,4}\\\cline{2-6}
2&M_{2,1}&M_{2,2}&\vrule&
M_{2,3}&M_{2,4}
}
```

$$\kbordermatrix{\mbox{indices}&1&2&\vrule&3&4\\ 1&M_{1,1}&M_{1,2}&\vrule&M_{1,3}&M_{1,4}\\\cline{2-6} 2&M_{2,1}&M_{2,2}&\vrule&M_{2,3}&M_{2,4}}$$

Tall items

Here is an example of how a tall item can destroy the row alignment, and how it can be fixed with `\renewcommand{\arraystretch}{2.2}`.

$$\kbordermatrix{\mbox{indices}&1&2\\ 1&\frac{M_{1,1}}{2}&\frac{M_{1,2}}{2}\\ 2&\frac{M_{2,1}}{2}&\frac{M_{2,2}}{2}\\ 3&\frac{M_{3,1}}{2}&\frac{M_{3,2}}{2}}\kbordermatrix{&\\ &a_{1,1}&a_{1,2}\\ &a_{2,1}&a_{2,2}\\ &a_{3,1}&a_{3,2}}$$

$$\renewcommand{\arraystretch}{2.2}\kbordermatrix{\mbox{indices}&1&2\\ 1&\frac{M_{1,1}}{2}&\frac{M_{1,2}}{2}\\ 2&\frac{M_{2,1}}{2}&\frac{M_{2,2}}{2}\\ 3&\frac{M_{3,1}}{2}&\frac{M_{3,2}}{2}}\kbordermatrix{&\\ &a_{1,1}&a_{1,2}\\ &a_{2,1}&a_{2,2}\\ &a_{3,1}&a_{3,2}}$$

The first example

By the way, the listing for the first display is (in part):

```
\kbordermatrix{\text{indices}&
(1\cdot11)&(1\cdot12)&(1\cdot22)&
\vrule&
(2\cdot11)&(2\cdot12)&(2\cdot22)\\
%
(1)&\lambda(1)^2&2\lambda(1)\lambda(2)&\lambda(2)^2&\vrule&0&0&0\\
```

```
(2)&0&0&0&\vrule&\lambda(1)^2&2\lambda(1)\lambda(2)&\lambda(2)^2\\
\hline
(111)&3&0&0&\vrule&0&0&0\\
(112)&0&2&0&\vrule&1&0&0\\
(122)&0&0&1&\vrule&0&2&0\\
(222)&0&0&0&\vrule&0&0&3}
```

where `\text{}` is defined in the **amstext** package.

Changing delimiters

The **kbordermatrix** package defines four style parameters that can be used to change the appearance of the array. The first two are `\kbldelim` and `\kbrdelim`, the left and right delimiters. By default they are `[` and `]` but you can change them like this:

```
\renewcommand{\kbldelim}{\{}
\renewcommand{\kbrdelim}{.}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \left\{ \begin{array}{cccc} 1 & 2 & 3 & 4 \\ M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \end{array} \right.$$

```
\renewcommand{\kbldelim}{\langle}
\renewcommand{\kbrdelim}{|}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \left\langle \begin{array}{cccc} 1 & 2 & 3 & 4 \\ M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \end{array} \right|$$

Or if you really like the `\bordermatrix` look, try

```
\renewcommand{\kbldelim}{(}
\renewcommand{\kbrdelim}{)}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \left(\begin{array}{cccc} 1 & 2 & 3 & 4 \\ M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \end{array} \right)$$

Changing the border row and column style

You can change the style of the border row and column column entries by redefining `\kbrowstyle` and `\kbcolstyle`, which are by default set to `\scriptstyle`. You could, for instance, say `\renewcommand{\kbrowstyle}{\relax}` to typeset the first row as usual. By the way, the upper left corner is governed by the column style. You can always use an `\mbox{}` to change the style of any particular entry.

Changing spacing

Besides changing the delimiters, you can also change the space inserted after the first row and after the first column. These are governed by the lengths `\kbrowsep` and `\kbcolsep`. By default they are `.2ex` and `.5\arraycolsep`, respectively.

```
\setlength{\kbrowsep}{0pt}
\setlength{\kbcolsep}{26pt}
```

$$\begin{array}{c} \text{indices} \\ 1 \\ 2 \end{array} \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \end{array} \right]$$

Note how `\setlength{\kbrowsep}{0pt}` connects the vertical rule.

Comparison with the array environment

There are some differences from the familiar `\left\[\begin{array}` ... `\end{array}\right\]` construction.

1. It takes the array as an argument. It does not use `\begin` ... `\end`. Is this a feature or a bug?
2. Consequently, you cannot use a column specifier (e.g., `{l|cr}`).
3. Consequently the maximum number of columns is not specified.
4. All columns are centered.
5. Vertical rules must be put in each row in a separate column as `\vrule`.
6. The first row and column are spaced a bit further apart from the rest.
7. The way the array is centered vertically, it is almost as if there is an invisible bottom row of the same height as the segregated top row that adds to the height of the equation. You can see that here. The `\fbox{$...$}` puts a box around the array, but the `\left\{...\right\}` show its relation to the "math axis."

```
\fboxsep0pt
\left\{\fbox{$\kbordermatrix{&1&2\\
    1&A&B\\
    2&C&D\\
    3&E&F}$}\right\}
```

$$\left\{\boxed{\kbordermatrix{ & 1 & 2 \\ 1 & A & B \\ 2 & C & D \\ 3 & E & F }}\right\}$$

Comparison with Knuth

The `\kbordermatrix` command is based on Knuth's `\bordermatrix` command, which is defined on page 361 of *The TEXbook* [2]. It also borrows from his `\vrulealign` command on page 392. His command yields a display like this:

$$\begin{array}{l|ccc|ccc}
\text{indices} & (1\cdot 11) & (1\cdot 12) & (1\cdot 22) & (2\cdot 11) & (2\cdot 12) & (2\cdot 22) \\
(1) & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & 0 & 0 & 0 \\
(2) & 0 & 0 & 0 & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 \\
\hline
(111) & 3 & 0 & 0 & 0 & 0 & 0 \\
(112) & 0 & 2 & 0 & 1 & 0 & 0 \\
(122) & 0 & 0 & 1 & 0 & 2 & 0 \\
(222) & 0 & 0 & 0 & 0 & 0 & 3
\end{array}
\left[\begin{array}{c} p_{1\cdot 11} \\ p_{1\cdot 12} \\ \hline p_{1\cdot 22} \\ p_{2\cdot 11} \\ p_{2\cdot 12} \\ p_{2\cdot 22} \end{array}\right]
\begin{array}{c} = \\ = \\ \hline \leq \\ \leq \\ \leq \\ \leq \end{array}
\left[\begin{array}{c} P_1 \\ P_2 \\ \hline 1 \\ 1 \\ 1 \\ 1 \end{array}\right]$$

This is not bad, but not surprisingly, I like mine better.

To illustrate a difference, let's "open up" the arrays with the LaTeX command `\renewcommand{\arraystretch}{1.5}`

$$\begin{array}{l|ccc|ccc}
\text{indices} & (1\cdot 11) & (1\cdot 12) & (1\cdot 22) & (2\cdot 11) & (2\cdot 12) & (2\cdot 22) \\
(1) & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & 0 & 0 & 0 \\
(2) & 0 & 0 & 0 & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 \\
\hline
(111) & 3 & 0 & 0 & 0 & 0 & 0 \\
(112) & 0 & 2 & 0 & 1 & 0 & 0 \\
(122) & 0 & 0 & 1 & 0 & 2 & 0 \\
(222) & 0 & 0 & 0 & 0 & 0 & 3
\end{array}
\left[\begin{array}{c} p_{1\cdot 11} \\ p_{1\cdot 12} \\ \hline p_{1\cdot 22} \\ p_{2\cdot 11} \\ p_{2\cdot 12} \\ p_{2\cdot 22} \end{array}\right]
\begin{array}{c} = \\ = \\ \hline \leq \\ \leq \\ \leq \\ \leq \end{array}
\left[\begin{array}{c} P_1 \\ P_2 \\ \hline 1 \\ 1 \\ 1 \\ 1 \end{array}\right]$$

$$\begin{array}{c|ccc|ccc}
\text{indices} & (1\cdot 11) & (1\cdot 12) & (1\cdot 22) & (2\cdot 11) & (2\cdot 12) & (2\cdot 22) \\
(1) & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 & 0 & 0 & 0 \\
(2) & 0 & 0 & 0 & \lambda(1)^2 & 2\lambda(1)\lambda(2) & \lambda(2)^2 \\
\hline
(111) & 3 & 0 & 0 & 0 & 0 & 0 \\
(112) & 0 & 2 & 0 & 1 & 0 & 0 \\
(122) & 0 & 0 & 1 & 0 & 2 & 0 \\
(222) & 0 & 0 & 0 & 0 & 0 & 3
\end{array}
\begin{array}{c|c|c}
p_{1\cdot 11} & = & P_1 \\
p_{1\cdot 12} & = & P_2 \\
\hline
p_{1\cdot 22} & \leq & 1 \\
\hline
p_{2\cdot 11} & \leq & 1 \\
p_{2\cdot 12} & \leq & 1 \\
p_{2\cdot 22} & \leq & 1
\end{array}$$

Notice that the rows of the `\kbordermatrix`, which was designed with LaTeX in mind, continue to line up with the other arrays, while `\bordermatrix`, which was written long before LaTeX, does not.

The `\kbordermatrix`, being derived from `\bordermatrix`, has a number of points in common with it.

1. It takes the array as an argument. It does not use `\begin{}..\end{}`. Is this a feature or a bug?
2. The lower $(n-1)\times(n-1)$ block is set off by delimiters.
3. There is an invisible bottom row of the same height as the segregated top row that adds to the height of the equation.

But there are some differences from `\bordermatrix`:

1. Square brackets are used in place of parentheses. (But you can change that.)
2. By default the first row and column are set with `\scriptstyle`. (But you can change that.)
3. You may use `\\` instead of `\cr`.
4. The line heights agree with LaTeX's line heights for the array environment, and `\arraystretch` is respected. This means the bottom $(n-1)$ rows align with the rows of an $(n-1)$-rowed `\begin{array}` ... `\end{array}` (with or without delimiters).
5. You can use `\hline` and `\cline`.
6. All columns are centered. Knuth's first column is left justified.
7. There is more space following the first row and column. (But you can change that.)
8. Knuth's `\bordermatrix` handles an empty argument more gracefully.

Why you need it

You might think that making such a display is easy, but let's see how we might try it. Here are some simpler displays. A standard LaTeX matrix array, is usually made like this,

```
\left[\begin{array}{l|cr}
\mbox{indices}&(1\cdot11)&(2\cdot11)\\\hline
1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\
2&a_{1,(1\cdot11)}&?
\end{array}\right]
```

$$\left[\begin{array}{l|cr}
\text{indices} & (1\cdot 11) & (2\cdot 11) \\
\hline
1 & a_{1,(1\cdot 11)} & a_{1,(2\cdot 11)} \\
2 & a_{1,(1\cdot 11)} & ?
\end{array}\right]$$

This is not bad for a labeled bordered matrix, but let's put this next to another array.

```
\left[\begin{array}{l|cr}
\mbox{indices}&
(1\cdot11)&(2\cdot11)\\\hline
1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\
2&a_{1,(1\cdot11)}&?
\end{array}\right]
\left[\begin{array}{cr}
b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\
b_{1,(1\cdot11)}&b_{2,(2\cdot11)}
\end{array}\right]
```

$$\left[\begin{array}{l|cr}\mbox{indices}&(1\cdot11)&(2\cdot11)\\\hline 1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\2&a_{1,(1\cdot11)}&?\end{array}\right]\left[\begin{array}{cr}b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\b_{1,(1\cdot11)}&b_{2,(2\cdot11)}\end{array}\right]$$

Usually you would want the rows to line up, and they don't. You can fix this by adding an extra blank row to the second array, like this,

```
\left[\begin{array}{l|cr}
\mbox{indices}&
(1\cdot11)&(2\cdot11)\\\hline
1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\
2&a_{1,(1\cdot11)}&?
\end{array}\right]
\left[\begin{array}{cr}\\
b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\
b_{1,(1\cdot11)}&b_{2,(2\cdot11)}
\end{array}\right]
```

$$\left[\begin{array}{l|cr}\mbox{indices}&(1\cdot11)&(2\cdot11)\\\hline 1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\2&a_{1,(1\cdot11)}&?\end{array}\right]\left[\begin{array}{cr}\\b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\b_{1,(1\cdot11)}&b_{2,(2\cdot11)}\end{array}\right]$$

Now the brackets on the second don't look right. So you could put the second array inside another array, like this:

```
\left[\begin{array}{lcr}
\mbox{indices}&(1\cdot11)&(2\cdot11)\\
1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\
2&a_{1,(1\cdot11)}&?
\end{array}\right]
\begin{array}{c}\\
\left[\begin{array}{cr}
b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\
b_{1,(1\cdot11)}&b_{2,(2\cdot11)}
\end{array}\right]
\end{array}
```

$$\left[\begin{array}{lcr}\mbox{indices}&(1\cdot11)&(2\cdot11)\\1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\2&a_{1,(1\cdot11)}&?\end{array}\right]\begin{array}{c}\\\left[\begin{array}{cr}b_{1,(1\cdot11)}&b_{1,(2\cdot11)}\\b_{1,(1\cdot11)}&b_{2,(2\cdot11)}\end{array}\right]\end{array}$$

This is much better, but the brackets on the left aren't where a mathematician wants them. Let's try the array inside an array trick again:

```
\begin{array}{lc}
\mbox{indices}&\begin{array}{cr}
(1\cdot11)&(2\cdot11)\end{array}\\
\begin{array}{c}1\\2\end{array}&
\left[\begin{array}{cr}
1&a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\
2&a_{1,(1\cdot11)}&?
\end{array}\right]
\end{array}
```

$$\begin{array}{lc}\mbox{indices}&\begin{array}{cr}(1\cdot11)&(2\cdot11)\end{array}\\\begin{array}{c}1\\2\end{array}&\left[\begin{array}{cr}a_{1,(1\cdot11)}&a_{1,(2\cdot11)}\\a_{1,(1\cdot11)}&?\end{array}\right]\end{array}$$

This works well in this case, but here is a natural example where this technique fails:

```
\begin{array}{lc}
\mbox{indices}&
\begin{array}{cccc}1&2&3&4\end{array}\\
\begin{array}{c}1\\2\end{array}&
\left[\begin{array}{cccc}
M_{1,1}&M_{1,2}&M_{1,3}&M_{1,4}\\
M_{2,1}&M_{2,2}&M_{2,3}&M_{2,4}
\end{array}\right]
\end{array}
```

$$\begin{array}{lc}
\mbox{indices}&
\begin{array}{cccc}1&2&3&4\end{array}\\
\begin{array}{c}1\\2\end{array}&
\left[\begin{array}{cccc}
M_{1,1}&M_{1,2}&M_{1,3}&M_{1,4}\\
M_{2,1}&M_{2,2}&M_{2,3}&M_{2,4}
\end{array}\right]
\end{array}$$

The problem is that by having three separate arrays for the first row, first column, and the body, the columns cannot maintain their spacing. You might try to use the `\lceil`, `\lfoor`, etc., commands to construct the delimiter on your own. The problem is matching the vertical rules with these pieces. It isn't well documented in *The TEXbook*. Moreover, you need to add blank rows at the top of you other arrays to get the proper row alignment. Here is my best shot at this approach, and it works reasonably well, at least in the one or two arrays I have used it. (It's on the left, `\kbordermatrix` is on the right.)

$$\begin{array}{c|c}
\mbox{indices} & (j,t) \\
\vdots & \vdots \\
(i,\tau) & \cdots \quad \delta_{i,j}\delta_{\tau,t_j}\mu_i(t^{-j}|\tau) \quad \cdots \\
\vdots & \vdots \\
\hline
\vdots & \vdots \\
s & \cdots \qquad \delta_{s,t} \qquad \cdots \\
\vdots & \vdots
\end{array}
\qquad
\left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \hline 4 \\ 5 \\ 6 \end{array}\right]
\qquad
\left[\begin{array}{c|c} 1 & \\ 2 & \\ 3 & \\ \hline 4 & \\ 5 & \\ 6 & \end{array}\right]
\qquad
\begin{array}{c|c}
\mbox{indices} & (j,t) \\
\vdots & \vdots \\
(i,\tau) & \cdots \quad \delta_{i,j}\delta_{\tau,t_j}\mu_i(t^{-j}|\tau) \quad \cdots \\
\vdots & \vdots \\
\hline
\vdots & \vdots \\
s & \cdots \qquad \delta_{s,t} \qquad \cdots \\
\vdots & \vdots
\end{array}$$

It is made by the following sets of commands.

```
\[
\begin{array}{cccc}
\multicolumn{1}{c}{\scriptstyle\text{indices}}&\multicolumn{3}{c}{\scriptstyle(j,t)}\\
\vdots&\toplft{\strut}&\vdots&\toprt{}\\
\scriptstyle(i,\tau)&\ledge{\cdots}&
\delta_{i,j}\delta_{\tau,t_j}\mu_i(t^{-j}|\tau)&\redge{\cdots}\\
\vdots&\ledge{}&\vdots&\redge{}\\ \cline{1-1}\cline{2-4}
\vdots&\ledge{}&\vdots&\redge{}\\
\scriptstyle s&\ledge{\cdots}&\delta_{s,t}&\redge{\cdots}\\
\vdots&\botlft{\strut}&\vdots&\botrt{}
\end{array}
\]
```

where `\toplft`, `\ledge`, `\botlft`, etc. are defined as follows.

```
\makeatletter
\newcommand{\tall}[1]{{\hbox{\smash{$\left#1\vtop to \arraystretch\ht\strutbox{}\right.\n@space$}}}}
\makeatother
\newlength{\lhoff} \newlength{\rhoff}
\setlength{\lhoff}{.6\arraycolsep} \setlength{\rhoff}{\lhoff}
\addtolength{\arrayrulewidth}{.3pt}
\def\toplft#1{\multicolumn{1}{@{\kern-\lhoff\tall\lceil\kern\lhoff}c}{#1}}
\def\ledge#1{\multicolumn{1}{|c}{\strut#1}}
\def\botlft#1{\multicolumn{1}{@{\kern-\lhoff\tall\lfloor\kern\lhoff}c}{#1}}
```

```
\def\toprt#1{\multicolumn{1}{c@{\kern\rhoff\tall\rceil\kern-\rhoff}}{#1}}
\def\redge#1{\multicolumn{1}{c|}{#1}}
\def\botrt#1{\multicolumn{1}{c@{\kern\rhoff\tall\rfloor\kern-\rhoff}}{#1}}
```

But you may find a better way to do things.

References

[1] K. C. Border. 2003. Reduced form auctions revisited. Social Science Working Paper 1175, California Institute of Technology, Pasadena CA 91125. www.hss.caltech.edu/SSPapers/wp1175.pdf

[2] D. E. Knuth. 1984. *The TEXbook*. Reading, Massachusetts: Addison–Wesley.