MATLAB Tips & Tricks: Exploiting the comma-separated list

Vectorizing cell array and structure references

by Clay M. Thompson

Have you ever found yourself wondering if it was possible to vectorize an expression that involved a structure or cell array? It is possible to vectorize some algorithms if you can take advantage of a comma-separated list. The comma-separated list syntax described below can be used to vectorize the following common operations:

- Concatenating conforming elements of a cell array into a single array (via \[ \] or cat)
- Converting between cell arrays and structure arrays or assigning multiple elements of a structure array at once (via deal and { })
- Indexing into an N-D array with N subscripts without knowing N ahead of time (see \texttt{fftshift} for an example)
- Passing on variable arguments (in conjunction with varargin or varargout)

If your algorithm fits one of these categories, you can operate on the whole cell array or structure at once. Since the speed that an M-file takes is usually proportional to the number of lines of code executed, using the comma-separated list syntax can speed up your M-files.

The comma-separated list

The comma-separated list has been part of MATLAB from the beginning. In its simplest form, it is a list of statements separated by commas as in

\[
x = 1:10, \ x, \ x(2)
\]

\[
x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}
\]

\[
\text{ans} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}
\]
Normally these statements would be placed on separate lines but you can put them all on one line by separating the statements with commas. Note that \texttt{ans} is displayed multiple times as each statement is evaluated.

A comma-separated list can be used in five situations:

\begin{itemize}
\item to combine multiple statements -- \texttt{a,b,c,d}
\item inside \texttt{[ ]} for horizontal concatenation -- \texttt{[a,b,c,d]}
\item inside \texttt{( )} to create a cell array -- \texttt{\{a,b,c,d\}}
\item inside \texttt{( )} for indexing and function input arguments -- \texttt{test(a,b)}
\item inside \texttt{[ ]} for function output arguments -- \texttt{[v,d] = eig(a)}
\end{itemize}

All of these situations work with the comma-separated list syntax for cell arrays and structures.

\textbf{The comma-separated list syntax}

The comma-separated list syntax is an extension of the syntax used to index into a cell array or structure and retrieve its contents. When you ask for more than one element, a comma-separated list is produced. For instance, suppose

\begin{verbatim}
strs = {'This' 'is' 'an' example'};
then

strs{4}
\end{verbatim}

\begin{verbatim}
ans =

   example
\end{verbatim}

and by indexing into more than one element at a time

\begin{verbatim}
strs{1:4}
\end{verbatim}

\begin{verbatim}
ans =

   This
\end{verbatim}

\begin{verbatim}
ans =

   is
\end{verbatim}

\begin{verbatim}
ans =

   an
\end{verbatim}

\begin{verbatim}
ans =

   example
\end{verbatim}
you get a comma-separated list. It is as if you typed

```matlab
>> strs{1},strs{2},strs{3},strs{4}
```

Note that MATLAB displays the value of `ans` multiple times, just like it did in the original comma-separated list expression. Any number of subscripts can be used. For example, `C(:,1)` is a comma-separated list if `C` has more than one row.

A comma-separated list is produced for a structure array when you access one field from multiple structure elements at a time. For instance if `S` is a 5-by-1 structure array then `S.name` is a five-element comma-separated list of the contents of the name field.

**Where to learn more**

In MATLAB, see the `help for lists, deal, varargin, vavargout`. Also see Chapter 13 of *Using MATLAB.*