Armstrong State University
Engineering Studies
MATLAB Marina – Cell Arrays Primer

Prerequisites
The Cell Arrays Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, arrays, logic expressions, conditional structures, iteration, functions, debugging, and characters and strings. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, MATLAB Marina Arrays module, MATLAB Marina Logic Expressions module, MATLAB Marina Conditional Structures module, MATLAB Marina Iteration module, MATLAB Marina Functions module, MATLAB Marina Debugging module, and MATLAB Marina Character and Strings module.

Learning Objectives
1. Be able to write MATLAB functions and programs that create cell arrays.
2. Be able to extract the containers and the contents of the containers (data) from cell arrays for use by MATLAB functions and programs.
3. Be able to store data generated by MATLAB functions and programs in cell arrays.

Terms
cell, cell array, container

MATLAB Functions, Keywords, and Operators
cell, ( ), { }

Cells and Cell Arrays
A MATLAB cell is a container that can hold data of any type. Cell arrays are arrays of containers. Cell arrays differ from "regular" arrays in that they can hold data of different types (arrays and vectors hold data of the same data type typically numbers). Cell arrays can be thought of as a collection of containers: each element of the cell array is a container that holds data and the data in each container does not have to be of the same data type or the same size.

MATLAB cell arrays are primarily used for file input/output. Cell arrays can also be used to organize collections of data of different types but structure arrays (covered in a later module) provide a better way of organizing collections of data of different types than cell arrays.

Creating Cell Arrays
Cell arrays can be created by:
- Entering the values directly with row elements separated by commas and elements in columns separated by semicolons all enclosed by braces
- Assigning an individual value to a variable indexed using braces
- Assigning a cell container to a variable indexed using parentheses

- Concatenating cell containers.
- Empty cell arrays can be created using the MATLAB function `cell`.

Examples of creating a one by four cell array containing a number, a character, a string, and an array of numbers are shown in Figures 1a – 1c.

```
>> ca1 = {1, 'a', 'abcde', [1 2 3]}
ca1 =     [1]    'a'    'abcde'    [1x3 double]
```

Figure 1a, Cell Array Creation using Direct Entry

```
>> ca2{1} = 5;
>> ca2{2} = 3;
>> ca2{3} = [1 2 3 4];
>> ca2
ca2 =   [5]    [3]    [1x4 double]
```

Figure 1b, Cell Array Creation from Values

```
>> ca3(1) = {'abc'}
ca3 =     'abc'
>> ca3(3) = {[1 2]}
ca3 =     'abc'    []    [1x2 double]
>> ca3(2) = {5}
ca3 =     'abc'    [5]    [1x2 double]
```

Figure 1c, Cell Array Creation from Containers

Figure 1d shows examples of creating empty cell arrays using the `cell` function. The MATLAB function `cell` takes one to three arguments and creates a cell array of the size specified with empty matrices for each cell. If one argument is given, a square 2-D cell array with that many rows and columns is created.

```
>> rowCellArray = cell(1,7)
rowCellArray =     []    []    []    []    []    []    []
>> twoDCellArray = cell(4,3)
twoDCellArray =
    []    []    []
    []    []    []
    []    []    []
    []    []    []
```

Figure 1d, Empty Cell Arrays

The elements of a cell arrays can be cells or cell arrays (cell array of cell arrays). Figure 1e shows an example of a cell array containing two cells and a cell array. Generally, one should try to keep the cell depth at one or two, otherwise the organization becomes confusing and extracting the contents of the buried cells can be difficult.

```
>> ca4{1} = {'a'};
>> ca4{2} = {'abc', 5, [2, 2, 2]};
>> ca4{3} = {5};
>> ca4
ca4 = {1x1 cell}    {1x3 cell}    {1x1 cell}
```

Figure 1e, Cell Array of Cell Arrays

Organizing Data using Cell Arrays

The MATLAB program of Figure 2a and display of the resulting cell arrays in Figure 2b illustrate the flexibility of data organization achievable with cell arrays. The same data is stored in both cell arrays but with difference organization.

```
% create cell array of grocery items and quantities
groceryData1 = {'apple',4, 'orange',3, 'banana',2, 'pear',8};
groceryData2 = {{'apple',4},{'orange',3},{'banana',2},{'pear', 8}};
```

Figure 2a, MATLAB Program to Create `groceryData` Cell Arrays

```
>> disp(groceryData1)
    'apple'   [4]   'orange'   [3]   'banana'   [2]   'pear'   [8]

>> disp(groceryData2)
    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}
```

Figure 2b, `groceryData` Cell Arrays

In the second cell array, `groceryData2`, each container is another cell array of two items. To extract the data in the second cell array, first the 1 by 2 container must be extracted and then the data in the 1 by 2 container can be extracted.

The MATLAB program of Figure 3a shows how similar cell arrays as those created in the program of Figure 2a could be created from data read from a user. Figure 3b shows a sample run of the program of Figure 3a and the contents of each cell array.

```
% create cell array of grocery items and quantities
numberItems = input('Enter number of items: ');
% initialize cell arrays
groceryData1 = cell(1,2*numberItems);
groceryData2 = cell(1,numberItems);
% read in items and quantities and store in cell array
for k = 1:numberItems
    item = input('Enter item: ', 's');
    message = sprintf('Enter quantity of %s: ', item);
    quantity = input(message);

    groceryData1{2*k-1} = item;
    groceryData1{2*k} = quantity;

    container = {item, quantity};
    groceryData2{k} = container;
end
```

Figure 3a, MATLAB Program to Create `groceryData` Cell Array

```
Enter number of items: 3
Enter item: apple
Enter quantity of apple: 4
Enter item: tuna
Enter quantity of tuna: 2
Enter item: pear
Enter quantity of pear: 8
>> disp(groceryData1)
    'apple'    [4]    'tuna'    [2]    'pear'    [8]
>> disp(groceryData2)
    {1x2 cell}    {1x2 cell}    {1x2 cell}
```

Figure 3b, `groceryData` Cell Arrays

Accessing/Extracting Data from Cell Arrays

To extract data from a cell arrays so that it can be operated on, the cell array is indexed (sliced) similar to arrays. The indexing can be done using parenthesis ( ) which will return the containers or using braces { } which will return the contents of the container (the data the container holds). Figure 4 shows an example of indexing to extract the containers and the contents of the containers. Since most operations involving cell arrays are concerned with processing the data in the cell array rather than the containers holding the data, indexing should generally be done using braces.

```
>> ca = {1, 'a', 'abcde', [1 2 3]};
>> ca(1)
ans =    [1]
>> ca(2)
ans =    'a'
>> ca(3)
ans =    'abcde'
>> ca(4)
ans =     [1x3 double]

>> ca{1}
ans =     1
>> ca{2}
ans =     a
>> ca{3}
ans =     abcde
>> ca{4}
ans =     1     2     3
```

Figure 4, Indexing Cell Arrays

When assigning cell arrays to variables, either when creating or accessing them, the number of variables on the left hand side of the assignment must equal the number of cells on the right hand side. Using parenthesis for indexing is regular indexing which returns a single item (scalar or array) whereas using braces for indexing works like a function call and can return multiple results (multiple containers).

In Figure 5a, the indexing operation returns a one by three cell array which is assigned to the variable `v1` (`v1` becomes a one by three cell array).

```
>> ca = {1, 'a', 'abcde', [1 2 3]};
>> v1 = ca(1:1:3)
v1 =    [1]    'a'    'abcde'
```

Figure 5a, Assignment of Result of Indexing Cell Array

In Figure 5b, the indexing operation returns a single cell array of size one by three. Assigning the single cell array to three variables, results in a syntax error.

```
>> [v1 v2 v3] = ca(1:1:3)
??? Indexing cannot yield multiple results.
```

Figure 5b, Improper Assignment of Result of Indexing Cell Array

In Figure 5c, the indexing operation returns three results (a number, a character, and a string) which are assigned to the corresponding variable in the list of three variables on the left side of the assignment.

```
>> [v1 v2 v3] = ca{1:1:3}
v1 =    1
v2 =    a
v3 =   abcde
```

Figure 5c, Proper Assignment of Result of Indexing Cell Array

In Figure 5d, the indexing operation returns three results as the previous example, but there is only one variable on the left side of the assignment. This is not a syntax error, but only the first of the extracted elements is saved in the variable, the other two are discarded as there is no variable to store them in.

```
>> v4 = ca{1:1:3}
v4 =    1
```

Figure 5d, Improper Assignment of Result of Indexing Cell Array, but Syntactically Legal

The general rules for indexing cell arrays are:

- When indexing to extract the contents of the cell array (using braces), have as many variables on the left side of the assignment as the number of elements extracted.
- When indexing to extract the containers of the cell array (using parentheses), have a single variable on the left side of the assignment as the result of the indexing results in a single cell array not multiple separate containers.

Operating on Data in Cell Arrays

Most MATLAB operations cannot be done on data in cell arrays since the type of data in the cell array may vary. To operate on data in cell arrays: the data in the cell array must be extracted, operated on, and moved back in the cell array if desired. Indexing, slicing, and iteration can be done on cell arrays.

The MATLAB program of Figure 6a illustrates the process of extracting data from a cell array and operating on it. Note that for the second cell array, `groceryData2`, first the 1 by 2 container is extracted and then the second element of the 1 by 2 container is extracted to get the quantity of that item. This could be done in one step although the syntax is a bit confusing

Note in Figure 6b, the `groceryData2` cell array contains four 1 by 2 cell arrays. An inner cell array can be extracted by indexing with braces, `groceryData2{1}`, returning the 1 by 2 cell array container. This container can then be indexed with braces again to extract one of the data elements in the container, `groceryData2{1}{2}`. The first indexing extracts one of the cell arrays and the second indexing extracts a data element from that cell array.

```
% create cell array of grocery items and quantities
groceryData1 = {'apple',4,'orange',3,'banana',2,'pear',8};
groceryData2 = {{'apple',4},{'orange',3},{'banana',2}, {'pear',8}};

% extract quantities of groceryData1 into separate array
numberItems1 = length(groceryData1)/2;
quantities1 = zeros(1,numberItems1);
for k = 1:numberItems1
    quantities1(k) = groceryData1{2*k};
end

% extract quantities of groceryData2 into separate array
numberItems2 = length(groceryData2);
quantities2 = zeros(1,numberItems2);
for k = 1:numberItems2
    container = groceryData2{k};
    quantities2(k) = container{2};
end
```

Figure 6A, MATLAB Program to Extract `groceryData` Cell Array Data

```
>> disp(groceryData2)
    {1x2 cell}    {1x2 cell}    {1x2 cell}    {1x2 cell}
>> disp(groceryData2{1})
    'apple'    [4]
>> disp(groceryData2{1}{2})
     4
```

Figure 6B, Extracting Data from Cell Array Containers in a Cell Arrays

Last modified Thursday, November 13, 2014