SMS 3515: Scientific Computing
Lecture 8: Data Structures: Cell Arrays and Structures
2014

Instructor: Nurul Farahain Mohammad

- Data structures: variables that store more than one value.
- Array: data structure in which all of the values are logically related in that they are of the same type and represent ”the same thing”.
- Cell array:
  - A kind of data structure that stores values of different types.
  - Can be vectors or matrices.
  - Store pointers to the stored data.
  - Store strings of different lengths.
- Structures:
  - Data structures that group together values that are logically related, but are not the same thing and not necessarily the same type.
  - Can be used to setup a database of information.
- Field: A component that break down the structure.
- *struct*: built-in function for field in MATLAB.
- Difference between arrays and structures: Cell arrays are indexed thus can be used with loops or vectorized code. However, structures are not indexed which the values are referenced using the names of the fields thus they can be more mnemonic than indexing.

1 Cell Arrays

- Elements in cell arrays are cells that can store different types of values.

1.1 Creating Cell Arrays

- Unlike vectors and matrices, cell arrays uses curly brackets instead of square brackets [ ].
- To create column cell array, values are separated by semicolon ;
- *cell(r,c)*: preallocate a cell array of r rows and c columns.

1.2 Referring to and Displaying Cell Array Elements and Attributes

- Content indexing: subscripts of a cell.
- Row and column subscripts are used to refer to the contents of an element in a matrix by using curly brackets .
- Values can be assigned to cell array elements using assignment sign =.
- Cell indexing: using parantheses ( ) for the subscripts references the cells.
- When an element of a cell array is itself a data structure, only the type of the element is displayed when the cells are shown. Thence, parantheses ( ) can be used to refer to its elements.
- Methods for displaying cell arrays:
  - *celldisp*: display the contents of all elements of the cell array.
  - *cellplot*: put a graphical display of the cell array into a Figure Window; show the cells, not their contents.
- Many of the functions and operations on arrays that we have already seen also work with cell arrays.

1.3 Storing Strings in Cell Arrays

- Cell arrays can store different types of values, strings of different lengths can be stored in the elements.
- Strings in cell arrays do not have extra trailing blanks.
- Length of each string can be displayed using a *for* loop through the elements of the cell array.
- *cellstr*: convert a character array padded with blanks to a cell array in which the trailing blanks have been removed.
- *char*: convert a cell array to a character matrix.
- *iscellstr*: return logical true if cell array is a cell array of all strings or logical false if not.

2 Structures

- Structures are data structures that group together values that are logically related in what are called fields of the structure.
- Fields are named.
- Structure variables are not array thence not possible to loop through the values.

2.1 Creating and Modifying Structure Variables

- Creating structure variables:
  - store values in fields using assignment statements.
  - use *struct*.
  - use dot operator to refer to fields within the structure.
- Adding a field to a structure by using assignment statement.
- To print from a structure:
  - *disp*: display either the entire structure or an individual field.
  - *fprintf*: only print individual fields.
- *rmfield*: remove a field from a structure and return a new structure with the field removed, but does not modify the original structure.

2.2 Passing Structures to Functions

- If entire structure variable is passed, function must use the dot operator to refer to the fields.
- If fields are passed to the function, the dot operator is not needed in the function.

2.3 Related Structure Functions

- *isstruct(structurename)*: return logical 1 for true if the variable argument is a structure variable or 0 if not.
- *isfield(structurename,'fieldname')*: return logical true if a fieldname is a field in the structure argument or logical false if not.
- *fieldnames*: return the names of the fields that are contained in a structure variable.
- curly brackets { } are used to refer to the elements.

2.4 Vectors of Structures

- Ways to create vectors of structures:
  - Treat first structure as a vector to begin with, and later extend the vector. Use parantheses ( ).
  - Create one element with the values from one structure, and use *repmat* to replicate it to the desired size. The remaining elements can then be modified.
- Typing the name of the variable will display only the size of the structure vector and the names of the fields.
- To display one element in the vector (one structure), an index into the vector would be specified.
- To refer to a field, it is necessary to refer to the particular structure and then the dot operator to refer to a field.

- 3 levels of data structure: variable (vector of structures); structure; field.
- Use *fprintf* to refer to a particular field for all structures.
- Use dot operator to refer to all values of a field and then store the values in a vector.

```
variablename = [structurename.fieldname]
```

2.5 Nested Structures

- A nested structure is a structure in which at least one member is itself a structure.
- Methods of defining nested structures:
  - Nest calls to the *struct* function.
  - Create structure variables first for the points, and then use these for the fields in the *struct* function.
  - Build the nested structure one field at a time.
- Typing the name of the variables shows only that it is a structure consisting of two fields.
- Typing the name of one of the nested structures will display the field names and values within that structure.
- Using dot operator twice will refer to an individual coordinate.

2.6 Vectors of Nested Structures

- Combination of vectors and nested structures.
- Levels: Vector of structs; struct; field; subfield.