Topic 17: Structures

Overview

- Structures
- Arrays of structures

Structures (Text: Chapter 7.3)

- In an array, each element is only known (or accessed) by specifying the name of the array followed by the row and column number of the element.

  For example:
  
  ```plaintext
  >> a(3, 2) = 6;
  ``

- In the case of normal arrays, each element must be of the same type.

- Cell arrays are accessed similarly, but may hold data of different types:

  ```plaintext
  >> a{3,2} = 6;
  >> a{3,3} = 'Hill'
  ```

What is a structure?

- A structure is a data type in which individual data elements are accessed by specifying a field name of the data element to access.

- The collection of fields in a structure is like a 1D cell array except that we use field names rather than indices to access data.

- Each field of a structure can refer to data of any type.

- We can access individual fields of a structure by specifying the structure name followed by a period and then the field name of interest.

- There are two ways of generating the structure.

  1. Adding one field at a time using assignment statements
  2. All at once using the `struct` function
Adding a field to a structure

- We can add a field to the structure at any time.
- Matlab automatically accommodates this in the same way that arrays and cell arrays can be extended at any time.
- For example, imagine creating a structure called `beam` with the fields `length`, `sectionarea`, `material` and `manufacturer`.

```
>> beam.length = 10.4;
>> beam.sectionarea = 0.03;
>> beam.material = 'Carbon Fibre'
>> beam.manufacturer = 'GLC'
beam =
    length: 10.4000
    sectionarea: 0.0300
    material: 'Carbon Fibre'
    manufacturer: 'GLC'
```

A diagram for the structure

Using the `struct` function

- The syntax of the `struct` function is:

  ```matlab
  struct('field1', value1, 'field2', value2, ...)
  ```

- Each `valueX` parameter used can be a single value or a cell array of values.

```
>> beam = struct('length', 1.2, 'sectionarea', 0.05, ...
    'material', 'steel', 'manufacturers', ...
    'unknown')
beam =
    length: 1.2000
    sectionarea: 0.0500
    material: 'steel'
    manufacturer: 'unknown'
```

Manipulate the fields

- The Matlab function `fieldnames` - recovers all the field names associated with a structure.
- returns a cell array of strings.
- For example:

```
>> fieldnames(beam)
ans =
    'length'
    'sectionarea'
    'material'
    'manufacturer'
```

- Matlab function `rmfield` - removes fields from a structure.
Arrays of structures

- Often we will want multiple instances of a single structure.
- We can build up an array of structures. We call this array a **structure array**.
- Matlab allows a structure array to grow automatically.

```matlab
>> beam(2).length = 5 % Beam is now a structure array.
> % Set the length field of the second
> % element of the structure array to 5.
beam =
1x2 struct array with fields:
  length
  sectionarea
  material
  manufacturer
```

Access the elements of a structure array

- The name `beam` is now used to refer to the entire array of structures.
- We can access elements of a structure array as if it was a normal array.
- For example:

```matlab
>> beam(1)
ans =
  length: 10.4000
  sectionarea: 0.0300
  material: 'Carbon Fibre'
  manufacturer: 'GLC'
```

- Since we only specified the `length` field of `beam(2)`, Matlab initialises the remaining fields to empty arrays:

```matlab
>> beam(2)
ans =
  length: 5
  sectionarea: []
  material: []
  manufacturer: []
```
- We can then fill in extra details as required.
- For example:

```matlab
>> beam(2).material = 'Steel';
```
Accessing elements of an individual structure

- We access elements of an individual structure in the array using
  - an array index to find the array element and then
  - a field name to access the field of the structure.
- For example:
  ```
  >> beam(2).length
  ans =
  5
  ```

Object Oriented Programming

- The beam structure gathers together and encapsulates all the information relating to a beam.
- Object oriented programming takes this encapsulation one step further.
- In object oriented programming:
  - In addition to the data associated with a beam, a "Beam Object" would also have a collection of functions (called methods) for doing calculations on beams.
  - For example, we can imagine methods to calculate deflection, overall weight, buckling strength, etc.

Structure arrays as data tables

- A structure array can also be considered to take the role of a database table:

<table>
<thead>
<tr>
<th>Id</th>
<th>Length</th>
<th>Section Type</th>
<th>Material</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.4</td>
<td>0.03</td>
<td>'Carbon Fibre'</td>
<td>'GLC'</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>0.05</td>
<td>'steel'</td>
<td>'unknown'</td>
</tr>
</tbody>
</table>

Creating a structure array with the `struct` function

- Recall the syntax of the `struct` function is:
  ```
  struct('field1', value1, 'field2', value2, ...)
  ```

- Each `valueX` parameter used can be a single value or a `cell array` of values.
  ```
  >> s = struct('type', {'big', 'little'},
                'colour', 'red', 'x', {3, 4})
  s =
  1x2 struct array with fields:
    type
colour
    x
Preallocate memory for a structure array

- The main use of the `struct` function is to preallocate memory for a structure array.
- In this case, each `valueX` parameter would be a call to the cell function to create empty cell arrays of the desired size.

```matlab
>> NBeams = 20;
>> beam = struct('length', cell(1, NBeams), ...
    'sectionarea', cell(1, NBeams), ...
    'material', cell(1, NBeams), ...
    'manufacturer', cell(1, NBeams))
beam =
   1x20 struct array with fields:
       length
       sectionarea
       material
       manufacturer
```
- This produces an empty array of structures, ready for data to be attached.