KU LEUVEN

MATLAB

Array creation

Topics

- Creating Arrays
  - manually entering the elements
  - special functions, special matrices

Creating Arrays

- To create a matrix that has multiple rows, separate the rows with **semicolons**.
  - always refer to rows first and columns second. 4-by-3

```
>> arr_1 = [1 3 5 ; 2 4 1 ; 3 3 3 ; 2 1 9]

arr_1 = 4×3

1 3 5
2 4 1
3 3 3
2 1 9
```

```
arr_2 = [1, 3, 5, 4; 2, 4, 1, 4 ; 3, 3, 3, 4 ; 2, 1, 9, 4]

arr_2 = 4×4

1 3 5 4
2 4 1 4
3 3 3 4
2 1 9 4
```

- create an empty matrix.

```
arr_3 = [] % empty array

arr_3 =

[]
```

KU LEUVEN

Creating Arrays

- The elements of an array can be
  - Numbers,
  - mathematical expressions,
  - functions.
- All the rows must have the same number of elements.
  - MATLAB displays an error message if an incomplete matrix is entered

Creating Arrays

MATLAB provides functions for creating standard arrays.

```
>> help elmat
  Elementary matrices and matrix manipulation.

  Elementary matrices.
    zeros       - Zeros array.
    ones        - Ones array.
    eye         - Identity matrix.
    repmat      - Replicate and tile array.
    rand        - Uniformly distributed random numbers.
    randn       - Normally distributed random numbers.
    linspace    - Linearly spaced vector.
    logspace    - Logarithmically spaced vector.
    freqspace   - Frequency spacing for frequency response.
    meshgrid    - X and Y arrays for 3-D plots.
    accumarray  - Construct an array with accumulation.
    :           - Regularly spaced vector and index into matrix.
```


Examples

```
>> ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
>> zeros(2,5)
ans =
     0     0     0     0     0
     0     0     0     0     0
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
>> rand(1,5)
ans =
    0.9501    0.2311    0.6068    0.4860    0.8913
```

```
>> randn(3,2)
ans =
   -0.4326    0.2877
   -1.6656   -1.1465
    0.1253    1.1909
>> a=1:4
a =
     1     2     3     4
>> diag(a)
ans =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
>> diag(a,-2)
ans =
     0     0     0     0     0     0
     0     0     0     0     0     0
     1     0     0     0     0     0
     0     2     0     0     0     0
     0     0     3     0     0     0
     0     0     0     4     0     0
```

Test matrices

- Use the `gallery` function to create test matrices
- `doc gallery`

Concatenating Arrays

- Arrays can be concatenated by enclosing them inside of square brackets [ ]
  - Use **space or comma** to glue horizontally
  - Use **semicolon** to specify to glue vertically
  - Dimensions must fit!

```
>> A = [[1 2 3],rand(1,3)]
A =
    1.0000    2.0000    3.0000    0.9649    0.1576    0.9706
>> B = [A;A]
B =
    1.0000    2.0000    3.0000    0.9649    0.1576    0.9706
    1.0000    2.0000    3.0000    0.9649    0.1576    0.9706
```

Concatenate

cat

- Concatenate arrays
- Syntax
  - `C = cat(dim,A,B)`
  - `C = cat(dim,A1,A2,A3,A4...)`
- `C = cat(dim,A,B)` concatenates the arrays A and B along dim.
- `C = cat(dim,A1,A2,A3,A4,...)` concatenates all the input arrays (A1, A2, A3, A4, and so on) along dim.
- `cat(2,A,B)` is the same as [A,B]
- `cat(1,A,B)` is the same as [A;B]

Repeating Arrays

- **repmat**
  syntax `B = repmat(A,m,n)` creates a large matrix B consisting of an m-by-n tiling of copies of A.
  `repmat(A,n)` creates an n-by-n tiling.

```
>> A = [1 2; 3 4]
A =
     1     2
     3     4
>> arr = repmat(A,3,2)
arr =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
```

Reshaping

- **reshape**
  `B = reshape(A,m,n)`
  returns the m-by-n matrix B whose elements are taken columnwise from A. An error results if A does not have m*n elements.

```
>> A = 1:9

x =

     1     2     3     4     5     6     7     8     9

>> arr = reshape(A,3,3)

arr =

     1     4     7
     2     5     8
     3     6     9
```

KU LEUVEN

Some utilities

- determine the size of an array by using the `size()` command

```
>> A = rand(3,4)

A = 3×4

0.3674 0.8852 0.0987 0.6797
0.9880 0.9133 0.2619 0.1366
0.0377 0.7962 0.3354 0.7212

size(A)

ans = 1×2

3 4

[nrows,ncols] = size(A)

nrows = 3

ncols = 4

ncols = size(A,2)

ncols = 4
```

KU LEUVEN

Some utilities

- `numel()` command.

```
>> A = rand(3,4)

A = 3×4

0.0835 0.3909 0.0605 0.4168
0.1332 0.8314 0.3993 0.6569
0.1734 0.8034 0.5269 0.6280
numel(A)

ans = 12
```

- `end:` indicate last array index

```
>> A = rand(3,4)

A = 3×4

0.2920 0.9841 0.3724 0.3395
0.4317 0.1672 0.1981 0.9516
0.0155 0.1062 0.4897 0.9203
A(end)

ans = 0.9203

B = A(end,1:end)

B = 1×4

0.0155 0.1062 0.4897 0.9203
```


Transposing an Array

- A m-by-n array can be transposed into a n-by-m array by using the transpose operator .' Check the documentation!

```
>> A = [1 2 3 4 ; 5 6 7 9]
A =
     1     2     3     4
     5     6     7     9
>> B= A'
B =
     1     5
     2     6
     3     7
     4     9
```

Multidimensional Arrays

- arrays with more than 2 subscripts
- Examples:
  - 3D physical data
  - sequence of matrices
  - samples of a time-dependent 2D or 3D data
- To make multidimensional array:

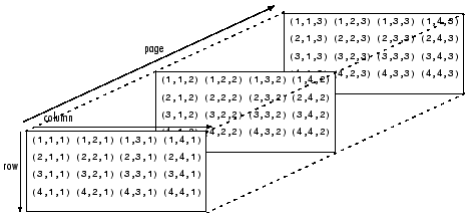
```
>> a = [2 4 6; 7 8 9; 1 2 3]
>> a(:,:,2)= [1 11 12; 0 1 2; 4 5 6]
```

When you add elements and expand the size
Unspecified elements are set to zero

```
>> a(:,:,4) = [ 1 1 1; 2 2 2; 3 3 3]
```


Multidimensional Arrays

- create a multidimensional array by creating a 2-D matrix first, and then extending it.




```
A = magic(3)
A = 3×3
8 1 6
3 5 7
4 9 2

A(:,:,2) = ones(3)
A =
A(:,:,1) =

8 1 6
3 5 7
4 9 2

A(:,:,2) =

1 1 1
1 1 1
1 1 1

A(:,:,4) = rand(3)
A =
A(:,:,1) =

8 1 6
3 5 7
4 9 2

A(:,:,2) =

1 1 1
1 1 1
1 1 1

A(:,:,3) =

0 0 0
0 0 0
0 0 0

A(:,:,4) =

0.1781 0.1711 0.8819
0.1280 0.0326 0.6692
0.9991 0.5612 0.1904
```

Demo / recap

- *File: create_arrays.mlx used in screencast matlab_array_create_arrays*