Introduction to MATLAB®

Lecture 2: Vectors and Matrices

Vectors and Matrices

- **Vectors** and **matrices** are used to store values of the same type
- A **vector** can be either column vector or a row vector.
- **Matrices** can be visualized as a table of values with dimensions $r \times c$ (r is the number of rows and c is the number of columns).

Creating row vectors

Place the values that you want in the vector in square brackets separated by either spaces or commas. e.g

```
>> row_vec=[1 2 3 4 5]
row_vec =
     1     2     3     4     5

>> row_vec=[1,2,3,4,5]
row_vec=
     1     2     3     4     5
```

Creating row vectors - colon operator

If the values of the vectors are regularly spaced, the colon operator can be used to iterate through these values.

(first:last) produces a vector with all integer entries from first to last e.g.

```
>> row_vec = 1:5
row_vec =
     1     2     3     4     5
```

Creating row vectors - colon operator

A **step value** can also be specified with another colon in the form (`first:step:last`)

```
>>odd_vec = 1:2:9
odd_vec =
     1     3     5     7     9
```

Exercise

In using (`first:step:last`), what happens if adding the step value would go beyond the range specified by last? e.g:

```
1          >>v = 1:2:6
```

Exercise

Use (`first:step:last`) to generate the vector `v1 = [9 7 5 3 1 ]`?

Creating row vectors - `linspace` function

linspace (Linearly spaced vector)

```
>>linspace(x,y,n)
```

`linspace` creates a row vector with n values in the inclusive range from x to y.

Example

```
>>v2 = linspace(3,15,5)
  v2 =
       3     6     9    12    15
```

Vector concatenation

- We can use existing vectors to create new ones

```
>> v1= 9:-2:1
v1 =
     9     7     5     3     1
>> v2 = linspace(3,15,5)
v2 =
     3     6     9    12    15
>> new_vec = [v1 v2]
new_vec =
     9     7     5     3     1     3     6     9    12    15
```

Accessing elements of a vector

new_vec

```
new_vec =
    9    7    5    3    1    3    6    9    12   15
```

- 5^{th} element

```
>> new_vec(5)
ans =
     1
```

- Elements 4 through 6

```
>> new_vec(4:6)
ans =
     3     1     3
```

Accessing elements of a vector

new_vec

```
new_vec =
    9    7    5    3    1    3    6    9    12   15
```

- Elements 2, 3, 7

```
>>new_vec([2 3 7])
ans =
     7     5     6
```

- To set the first entry to 10

```
>> new_vec(1)=10
new_vec =
10    7    5    3    1    3    6    9    12   5
```

Exercise

1. Use the help function to look up how the `zeros` function works.
2. Use the `zeros` function to create a row vector of 10 zeros, name it `zero_vec`
3. Set entries 2 through 4 of `zero_vec` to 1
4. Set entries 7 though 10 of `zero_vec` to values 7 through 10.

What about column vectors?

- Explicitly

```
>> col_vec = [2;4;6;8]
col_vec =
     2
     4
     6
     8
```

- No direct way using `colon` operator or `linspace` function...BUT we can fix this!

What about column vectors?

- Simply take the transpose of your row vector!

```
>> row_vec =2:2:8
row_vec =
     2     4     6     8
>> col_vec = row_vec'
col_vec =
     2
     4
     6
     8
```

Matrix variables

- We generalize the creation of row and column vectors, e.g.

```
>> mat1 = [3 4 5; 2 5 6]
mat1 =
     3     4     5
     2     5     6
```

- Or use the `colon` operator within rows

```
>> mat2=[1:3; 4:6]
mat2 =
     1     2     3
     4     5     6
```

Caution

- NOTE: There must always be the same number of values in each row
- If you attempt to create a matrix with different number of values in the rows MATLAB will complain:

```
>> mat=[1 2 4; 3 4]
Error using vertcat
Dimensions of matrices being concatenated
```

Matrices of random numbers

- `rand(n)` - returns an $n \times n$ matrix of random values on $(0, 1)$.
- `rand(m,n)` - returns an $m \times n$ matrix.
- A 3×3 matrix of random entries on $(0, 1)$.

```
>> rand(3)
ans =
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
```

- A (2×3) matrix of random entries on $(0, 1)$.

```
>> rand(2,3)
ans =
    0.7922    0.6557    0.8491
    0.9595    0.0357    0.9340
```

Referring to and modifying matrix elements

- Given the following 3×4 matrix

```
>> mat=[1:4;5:8;9:12]
mat =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

- We can extract element $mat_{2,3}$ as follows:

```
>> mat(2,3)
ans =
     7
```

- The submatrix consisting of rows $1, 2$ and columns $2, 3$:

```
>> mat(1:2,2:3)
 ans =
      2     3
      6     7
```

Referring to and modifying matrix elements

- Given

```
>> mat=[1:4;5:8;9:12]
mat =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

- Extract the first row of `mat`

```
>> mat(1,:)
ans =
     1     2     3     4
```

- Extract the first column of `mat`

```
>> mat(:,1)
ans =
     1
     5
     9
```

Referring to and modifying matrix elements

- Given

```
>> mat=[1:4;5:8;9:12]
mat =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

- Set the first column of `mat` to `[5;6;7]`

```
>> mat(:,1) = 5:7
mat =
     5     2     3     4
     6     6     7     8
     7    10    11    12
```

Deleting vector entries

- Given a vector with 8 random integers between 1 and 10

```
>> vec=round((rand(1,8))*10)
vec =
     7     2     1     5    10     3     6     2
```

- We can delete the 3*rd* entry using the *empty vectors* `[]` as

```
>> vec(3)=[]
vec =
     7     2     5    10     3     6     2
```

- To delete multiple entries, simply pass in an `index array`, e.g. to delete all even entries:

```
>> vec(2:2:length(vec)) =[]
vec =
     7     1    10     6
```

- Note: Here, we can also use `end` as `>>vec(2:2:end) = []`

Deleting matrix entries

- Given a 5×5 matrix

```
>> M=magic(5) %look up the magic function!
M =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

- Delete the 3*rd* column

```
>> M(:,3)=[]
M =
    17    24     8    15
    23     5    14    16
     4     6    20    22
    10    12    21     3
    11    18     2     9
```

Adding entries

- Given a 5×5 matrix

```
>> M=magic(5)
M =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

- Add a 6th column

```
>> [M,[1:5]']
ans =
    17    24     1     8    15     1
    23     5     7    14    16     2
     4     6    13    20    22     3
    10    12    19    21     3     4
    11    18    25     2     9     5
```

Exercise

1. Add a 6th row to the matrix M
2. Look up the `diag` function and extract the diagonal of the matrix M

Some useful array commands

Find out what each command does before starting **Homework 1**.

1. `length`
2. `size`
3. `numel`
4. `max`
5. `min`
6. `sort`
7. `zeros`
8. `ones`
9. `reshape`
10. `eye`
11. `spy`