Indexes In Matrices and Vectors

In order to access the data stored in vectors or matrices, you need to know the index of the value you wish to access. Every number in matrices and vectors has its own index, and data can be extracted from matrices and vectors using indexes. This is where Matlab differs from certain other languages. In Matlab, the first element in any array or matrix has an index of 1 (not 0), the second element is 2, and so on. For example, if a vector is declared as below:

input:

```
>> d = 6:2:16
```

output:

```
d =
    6    8   10   12   14   16
```

The index of 6 will be 1 since it is the first element; the index of 8 will be 2 since it is the second. Indexes in matrices are similar, except that they have another dimension, which requires another index. For example:

input:

```
>> d = [1:5;6:10]
```

output:

```
d =
    1    2    3    4    5
    6    7    8    9   10
```

The index of 1 will now be 1,1 because it is in the first row, and is the first number. The index of 6 will be 2,1 because it is in the second row and is the first number. A better way to explain this will be: a vector(1D) is a collection of numbers and a matrix(2D) is a collection of vectors; in a matrix, each row is a vector. So, 1 refers to the first vector and in this first vector, another 1 refers to the first number, which gives its index of 1,1. This idea can also be applied to data that have more dimensions. For example a 3D matrix is a collection of 2D matrices, so an index of a number in a 3D matrix will have 3 parts, being (which 2D matrix),(which vector in that 2D matrix),(which number in that vector).

To access the specific value at the specific index, you just have to place the index in parentheses and right after the vectors/arrays/matrices variable. For example:

input:

```
>> x = 1:10
>> a = x(6)
```

output:

```
x =
    1    2    3    4    5    6    7    8    9   10
a =
    6
```

There is an advanced use of indexes - placing a vector instead of just one number. This action will give you a vector. As always, 2D arrays (matrices) need 2 index keys, and this will give you a matrix. The length of the vector you put will be the length of the vector you get and

the dimensions of your matrix. A shortcut could be a colon. A colon by itself means everything from the first element to the last element. For example:

input:

```
>> x = 1:10
>> a = x(2:5)
```

output:

```
x =
    1    2    3    4    5    6    7    8    9   10
a =
    2    3    4    5
```

input:

```
>> d = [1:10;11:20;21:30]
>> x = d(1, 3:5)
```

output:

```
d =
    1    2    3    4    5    6    7    8    9   10
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25   26   27   28   29   30
x =
    3    4    5
```

input:

```
>> d = [1:10;11:20;21:30]
>> x = d(2, :)
```

output:

```
d =
    1    2    3    4    5    6    7    8    9   10
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25   26   27   28   29   30
x =
   11   12   13   14   15   16   17   18   19   20
```

input:

```
>> d = [1:10;11:20;21:30]
>> x = d(1:2, 5:8)
```

output:

```
d =
    1    2    3    4    5    6    7    8    9   10
   11   12   13   14   15   16   17   18   19   20
   21   22   23   24   25   26   27   28   29   30
x =
    5    6    7    8
   15   16   17   18
```