Exam 1: Lectures 1 – 5, Open-note, JH 245

Sep 23rd (Tue)： 3:30 – 5:30 PM

Lecture 8

Lecture 8: Looping

- **Vectorizing**
  - In MATLAB many operations can be done and functions can be called with vectors and matrices. The term vectorizing is used in MATLAB for rewriting code using loops in a traditional programming language to matrix language in MATLAB. For example, assuming a matrix variable mat:

```
[r  c] = size(mat)
for row = 1:r
    for col = 1:c
        % do something with mat(row, col)
    end
end
```

This is the traditional way to loop through the elements

Usually in MATLAB, this is not necessary

Lecture 8: Looping

- **Vectorizing (Cont)**
  - Examples

```
>> v = [3  7  2  1];
>> v = v *3
v =
    9  21  6  3

>> v = [3  7  2  1];
>> v/2
ans =
    1.5000   3.5000  1.0000   0.5000
```

For the exponentiation operation, .^ must be used when working with vectors and matrices.

```
>> v = [3  7  2  1];
>> v .^ 2
ans =
     9   49   4  1
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - An entire vector or matrix can be passed as an argument to a function; the function will be evaluated on every element.

```
>> vec = -2:1
vec =
     -2   -1   0   1
>> sinvec = sin(vec)
sinvec =
   -0.9093   -0.8415   0   0.8415
>> signvec = sign(vec)
signvec =
   -1   -1   0   1
```

```
for i=1:4
    sinvec(i) = sin(vec(i))
end
```

```
for i=1:4
    signvec(i) = sign(vec(i))
end
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Array operation

```
>> vec1 = 1:4
vec =
     1   2   3   4
>> vec2 = 5:8
ans =
     5   6   7   8
>> vecnew = vec1 * vec2

>> vecnew = vec1 .* vec2
```

The matrix multiplication follow the rules of linear algebra. It requires the number of columns of the first matrix to be the same as the number of rows of the second matrix.

```
for i=1:4
    vecnew(i) = vec1(i) * vec2(i)
end
```

For array operations, vec 1 and vec2 must have the same dimensions

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Array operation

```
>> mat1 = [1  2; 3  4]
mat1 =
     1   2
     3   4
>> mat2 = [2  2; 2  2]
ans =
     2   2
     2   2
>> matnew1 = mat1 * mat2

>> matnew2 = mat1 .* mat2
```

matrix multiplication

$$matnew1_{ij} = \sum_{k}^{m} mat1_{ik} * mat2_{kj}$$

```
array multiplication
for i=1:4
   for j=1:4
      matnew2 (i,j) = mat1(i,j) * mat2(i,j)
   end
end
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Array operation

```
>> mat1 = [1  2; 3  4]
mat1 =
     1   2
     3   4
>> mat2 = [2  2; 2  2]
ans =
     2   2
     2   2
>> matnew1 = sin(mat1) * sin(mat2)

>> matnew2 = sin(mat1) .* sin(mat2)
```

```
array multiplication
for i=1:2
   for j=1:2
      matnew2 (i,j) = sin(mat1(i,j)) * sin(mat2(i,j))
   end
end
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Array operation

```
>> mat1 = [1  2; 3  4]
mat1 =
     1   2
     3   4
>> mat2 = [5  6; 7  8]
ans =
     5   6
     7   8
>> mat1 / mat2

>> mat1 ./ mat2
```

matrix division

$$mat1 * inv(mat2)$$

array division

```
for i=1:2
   for j=1:2
      matnew(i,j)=mat1(i, j)/mat2(i, j)
   end
end
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Vectors or matrices can be passed to user-defined function, as long as the operators used in the function are correct. For example, " .* " must be used instead of " * " when multiplying vectors term-by-term in the function.

Example:

calcarea.m

```
function area = calcarea(rad)
% This function calculates the area of a circle
area = pi * rad * rad;
```

```
>> calcarea(1:3)
Error using  *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in
the second matrix. To perform elementwise multiplication, use '.*'.

Error in calcarea (line 3)
area = pi * rad * rad;
```

Lecture 8: Looping

- **Vectors and Matrices as Function Arguments**
  - Vectors or matrices can be passed to user-defined function as well, as long as the operators used in the function are correct. For example, " .* " must be used instead of " * " when multiplying vectors term-by-term in the function.

Example:

calcareaii.m

```
function area = calcareaii(rad)
% This function calculates the area of a circle
area = pi * rad .* rad;
```

```
>> calcarea(1:3)
ans =
   3.1416   12.5664   28.2743
>> calcarean(3)
ans =
    28.2743
```

When developing functions, the array operations should be used between variables.

Lecture 8: Looping

- **Summary**
  - Vectorizing
  - Vectors and matrices as function arguments

  - Homework on Canvas (due next Friday)