Vectorization

In this VG101 course, we will learn about three programming languages: Matlab, C and C++. Among those three languages, Matlab is the slowest while running, especially when you use loop commands such as for and while. However, sometimes in Matlab you can make good use of the feature of Matlab that everything is stored in a matrix to save time from long loops. Such method is called vectorization.

Algebraic Operations on Matrices

The most basic operations are plus/minus and multiplication of matrices, and I believe that you have all learned about them in your math classes. And please keep in mind that these operations all have some requirements on the sizes of the matrices.

The multipliation of matrices is usually used in the calculation of linear functions. For example, if you have the following function $z = x + 2y + 1$ and you want to know the value of z at the point (1,1) and (1,2), then you can write like following:

```
Z = [1,1,1;1,1,2]*[1;1;2]
Z =
    4
    6
```

The former matrix are the values of variables(there is another 1 because of the constant term in z) and the latter matrix represents the coefficients in the linear function.

Besides the operations above, Matlab also gives you some special operation called "element-by-element" multiplication/division, for which the operators are .* and ./. The two operation requires that the two matrices should have exactly the same size and the result is worked out by multipling/dividing every pair of elements in two matrices which are on the same location. For example:

```
A = [1,2,3;4,5,6;7,8,9];
B = [9,8,7;6,5,4;3,2,1];
A.*B

ans =
  9 6 21
  24 25 24
```

```
  21 6 9

A./B

ans =
  0.1111 0.2500 0.4286
  0.6667 1.0000 1.5000
  2.3333 4.0000 9.0000
```

These two operations will be very useful because there will be many situations in which you should inplement an element-by-element operation.

We all know that for vectors, there is another operation which is also very important---inner product, which is also called a "dot product" in high school. Based on the operations mentioned before, it is easy to deduce that there are two ways to implement an inner product of two vectors(suppose that they are all column vectors):

```
inner_product = transpose(A)*B;
inner_product = sum(A.*B);
```

The function ***transpose*** is just the transpose operation of a matrix, and I believe that you have already learned that in your math classes.

Algebraic Operation on a Matrix Together with a Scalar

Addition/Subtraction/Multiplication/Division

If you do a plus/minus/multiply/divide operation on a matrix and a constant, then it will be a element-by-element operation. Same calculation will be applied on every element in the matrix and the scalar constant.

```
[1,2;3,4]+1

ans =
  2 3
  4 5
```

Power Operation

If you want to get the square/cubic/... powers of every element in the matrix, use the .^ command:

```
A = [1,2;3,4];
A.^2

ans =
  1 4
  9 16
```

Same operation can be used conversely too

```
2.^A

ans =
  2 4
  8 16
```

Math Functions

Many math functions such as ***sin***() and ***exp***() can also apply on a matrix, which will apply a element-by-element calculation.

Special Matrices

Many special matrices will also help you to save time, and here I will list some which are often used.

Matrices with 0s and 1s

```
zeros(m,n) %m*n matrix whose elements are all zero
ones(m,n) %m*n matrix whose elements are all one
eye(n) %n*n identity matrix
```

Arithmatic Sequence

```
linspace(a,b,n) %An arithmatic sequence from a to b with n elements
a:d:b %An arithmatic sequence from a to b with common difference d
a:b %An arithmatic sequence from a to b with common difference 1
```

And what mentioned above are all important vectorization methods in MATLAB. Look back to the exercise 2 in the previous chapter(one row challenge), all of the solutions use vectorizations to reduce the size of the code.

Because actually you do not have to apply vectorization to save time and size although it is highly recommended, there will be no exercise for this chapter for you.