Matrix Operations in MATLAB

Nishant Soni

Overview

- **Working with arrays (matrices and vectors) in MATLAB**
- Linear algebra
- Vectorized operations
- Array operators

Manipulation of Matrices and Vectors

- **MATLAB** is an abbreviation of "`MATrix LABoratory`".
- Matlab make it easy to perform the standard operations of linear algebra including,
  - addition and subtraction,
  - multiplication of vectors and matrices,
  - solving linear systems of equations.
- We provide a simple introduction to some operations that are necessary for linear algebra calculations.
  - Vector addition and subtraction
  - Inner and outer products
  - Vectorization
  - Array operators

Manipulation of Matrices and Vectors

- Vector addition and subtraction are element-by-element operations.

```
>> a = [3 6 8];
>> b = [7 4 2];      % a and b are row vectors
>> a + b
ans =
    10   10   10
>> a - b
ans =
    -4  2  6
```

Vector Inner and Outer Products

- The inner product combines two vectors to form a scalar

$$\sigma = u \cdot v = u \cdot v^T \iff \sigma = \sum u_i v_i \tag{1}$$

- The outer product combines two vectors to form a matrix

$$A = u^T \cdot v \iff a_{i,j} = u_i v_j \tag{2}$$

Inner and Outer Products in MATLAB

- Inner and outer products are supported in Matlab as natural extensions of the multiplication operator.

```
>> a = [3 6 8];
>> b = [7 4 2];        % a and b are row vectors
>> a * b'
ans =
     61
>> a' * b
ans =
     21 12 6
     42 24 12
     56 32 16
```

Vectorization

- **Vectorization** is the use of single, compact expressions that operate on all elements of a vector without explicitly executing a loop.
- The loop is executed by the `MATLAB` kernel, which is much more efficient at looping than interpreted `MATLAB` code.
- Vectorization allows calculations to be expressed succintly so that programmers get a high level (as opposed to detailed) view of the operations being performed.
- Vectorization is important to make `MATLAB` operate efficiently.

Vectorization of Built-in Functions

- Most built-in function support vectorized operations.
- If the input is a scalar the result is a scalar.
- If the input is a vector or matrix, the output is a vector or matrix with the same number of rows and columns as the input.

```
>> x = 0:pi/4:pi
x =
    0    0.5236    1.0472    1.5708    2.0944    2.6180    3.1416
>> y = sin(x)
y =
    0    0.5000    0.8660    1.0000    0.8660    0.5000    0.0000
```

- No explicit loop is necessary in MATLAB.

Examples on Vectorization

```
>> A = pi*[ 1  2; 3  4]
A =
    3.1416   6.2832
    9.4228  12.5664

>> C = cos(A)
C =
   -1   1
   -1   1

>> B = A/2
B =
    1.5708    3.1416
    4.7124    6.2832

>> S = sin(B)
S =
    1.0000    0.0000
   -1.0000   -0.0000
```

Array Operators

- Array operators support element-by-element operations that are not defined by the rules of linear algebra.
- Array operators are designated by a period prepended to the standard operator.

Symbol	Operation
$.*$	element-by-element multiplication
$./$	element-by-element "right" division
$.\backslash$	element-by-element "left" division
$.\wedge$	element-by-element exponentiation

- Array operators are a very important tool for writing vectorized code.

Examples on using Array Operators

- Element-by-element multiplication and division

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = a.*b          (element-by-element product)
c =
     4   10   18
>> d = a./b
d =
    0.2500   0.4000   .5000
>> x = sin(pi*a/2).*cos(pi*b/2)
x =
    1.0000   0.0000   1.0000
>> y = sin(pi*a/2)./cos(pi*b/2)   (floating-point arithmetic is not exact)
y =
    1.0000    0.4000    1.0000    (0/0 evaluates to a finite value 0.4)
```

- Beware of the floating point arithmetic calculations on computer. On my machine, the last calculation gives a finite value (0.4) while evaluating a 0./0. quantity.

Examples on using Array Operators

- Application to matrices.

```
>> A = [1 2 3 4; 5 6 7 8];
>> B = [8 7 6 5; 4 3 2 1];
>> A.*B
ans =
     8    14    18    20
    20    18    14     8

>> A*B
??? Error using ==> *
Inner matrix dimensions must agree.

>> A*B'
ans =
    60    20
   164    60

>> A.^2
ans =
     1     4     9    16
    25    36    49    64
```