MATLAB

array operations

Topics

- Mathematical operations
- Manipulation

Arithmetic operations

- 2 different types of arithmetic operations:
  - array operations
  - matrix operations.
- Arithmetic operations
  - adding two numbers,
  - raising the elements of an array to a given power,
  - multiplying two matrices
  - Etc.
- Matrix operations follow the rules of linear algebra.
- Array operations execute element by element operations and support multidimensional arrays.
  - The period character (.) distinguishes the array operations from the matrix operations.

operator

http://www.mathworks.nl/help/techdoc/ref/arithmeticoperators.html#f75-87292		function
Binary addition	A+B	plus(A,B)
Unary plus	+A	uplus(A)
Binary subtraction	A-B	minus(A,B)
Unary minus	-A	uminus(A)
Matrix multiplication	A*B	mtimes(A,B)
Arraywise multiplication	A.*B	times(A,B)
Matrix right division *Divide by*	A/B	mrdivide(A,B)
Arraywise right division	A./B	rdivide(A,B)
Matrix left division *Divide into*	A\B	mldivide(A,B)
Arraywise left division	A.\B	ldivide(A,B)
Matrix power	A^B	mpower(A,B)
Arraywise power	A.^B	power(A,B)
Complex transpose	A'	ctranspose(A)
Matrix transpose	A.'	transpose(A)

Array operations

- Array operations
  - execute element by element operations on corresponding elements of vectors, matrices, and multidimensional arrays.
  - If the operands have the same size, then each element in the first operand gets matched up with the element in the same location in the second operand.
  - If the operands have compatible sizes, then each input is implicitly expanded as needed to match the size of the other.

Scalar-Array arithmetic

- Addition, subtraction, multiplication, and division of an array by a scalar applies the operation to all elements of the array.
- Implies scalar expansion for addition and subtraction to have the mathematics correct

```
>> A = [1 2 3 4; 5 6 7 8]
A =
     1     2     3     4
     5     6     7     8

>> A-2
ans =
    -1     0     1     2
     3     4     5     6

>> 2*A-1
ans =
     1     3     5     7
     9    11    13    15

>> 3*A/5+4
ans =
    4.6000    5.2000    5.8000    6.4000
    7.0000    7.6000    8.2000    8.8000
```

Matrix operations

- Dimensions must agree!

```
arr_1 = [2 1;5 7]
arr_1 = 2×2
2 1
5 7

arr_2 = [1 2;0 1]
arr_2 = 2×2
1 2
0 1
```

```
arr_sum = arr_1 + arr_2
arr_sum = 2×2
3 3
5 8

arr_subtract = arr_1 - arr_2
arr_subtract = 2×2
1 -1
5 6

arr_mult = arr_1 * arr_2
arr_mult = 2×2
2 5
5 17

arr_power = arr_1 ^2
arr_power = 2×2
9 9
45 54
```


Element-by-element operations: dot (.) operator

- Arithmetic operations on arrays are just like the same operations for scalars but they are carried out on an element-by-element basis.
- The dot(.) before the operator indicates an array operator; it is needed only if the meaning cannot be automatically inferred.
- applies to vectors, matrices, multi-dimensional arrays

Operation	Meaning
`C = a./A`	$C_{ij} = a/A_{ij}$
`C = A.\a`	$C_{ij} = a/A_{ij}$
`C = A.^a`	$C_{ij} = A_{ij}^{a}$
`C = a.^A`	$C_{ij} = a^{A_{ij}}$
`C = A.*B`	$C_{ij} = A_{ij}B_{ij}$
`C = A./B`	$C_{ij} = A_{ij}/B_{ij}$
`C = A.\B`	$C_{ij} = B_{ij}/A_{ij}$
`C = A.^B`	$C_{ij} = A_{ij}^{B_{ij}}$

dot (.) operator

- The dot operator, used with multiplication, division, and exponentiation, creates element-wise operations.
- The one exception to that is the use of the dot operator in creating matrix transposes. The ‘regular’ matrix transpose (**'**) creates the complex-conjugate transpose of a complex vector or matrix. Using the (**.'**) creates the transpose without doing the complex-conjugate operation.

Element-by-element operations

```
arr_1 = [2 1;5 7]
arr_1 = 2×2
2 1
5 7

arr_2 = [1 2;0 1]
arr_2 = 2×2
1 2
0 1

arr_elem_mult = arr_1 .* arr_2
arr_elem_mult = 2×2
2 2
0 7

arr_elem_power = arr_1 .^ 2
arr_elem_power = 2×2
4 1
25 49
```

```
scalar_elem_right = 1 ./ arr_1
scalar_elem_right = 2×2
0.5000 1.0000
0.2000 0.1429

scalar_elem_power = 2 .^arr_1
scalar_elem_power = 2×2
4 2
32 128

arr_arr_power = arr_1 .^arr_2
arr_arr_power = 2×2
2 1
1 7

arr_elem_div = arr_1 ./ arr_2
arr_elem_div = 2×2
2.0000 0.5000
Inf 7.0000

arr_elem_backslash = arr_1 .\ arr_2
arr_elem_backslash = 2×2
0.5000 2.0000
0 0.1429
```

Implicit expansion

- MATLAB R2016b, contains a feature called implicit expansion, which is an extension of the scalar expansion.
- MATLAB now treats "matrix plus vector" as a legal operation. This is a controversial change, as it means that MATLAB now allows computations that are undefined in linear algebra.

```
A = ones(2), B = A + [1 5]

A = 2×2

1 1
1 1


B = 2×2

2 6
2 6


A = ones(2) + [1 5]'

A = 2×2

2 2
6 6
```


KU LEUVEN

More on Array Operations

- Most MATLAB functions will work equally well on both scalars and arrays (of any dimension)

```
>> A=[1 2 3 4 5];
>> sin(A)
ans =
   0.8415    0.9093
 0.1411   -0.7568
 -0.9589
>> sqrt(A)
ans =
  1.0000    1.4142    1.7321
 2.0000    2.2361
```

KU LEUVEN

Columns first!

- Most common functions operate on columns by default

```
>> A = [1:3;4:6;7:9]
A =
     1     2     3
     4     5     6
     7     8     9

>> mean(A)
ans =
     4     5     6

>> sum(A)
ans =
    12    15    18
```

Built-in functions

`help elmat`: Matrix manipulation.

- `fliplr`: Flip matrix in left/right direction.
- `flipud`: Flip matrix in up/down direction.
- `rot90`: Rotate matrix 90 degrees. rot90(a,n): Rotate n-times
- `circshift(A,shiftsize)` circularly shifts the values in the array, A, by shiftsize elements.
  - shiftsize is a vector of integer scalars where the n-th element specifies the shift amount for the n-th dimension of array A.
  - positive shiftsize: shift down (or to the right).
  - negative shiftsize: shift up (or to the left).

```
a =
     1     2     3
     4     5     6
     7     8     9
>> flipud(a)
ans =
     7     8     9
     4     5     6
     1     2     3
>> fliplr(a)
ans =
     3     2     1
     6     5     4
     9     8     7
>> rot90(a)
ans =
     3     6     9
     2     5     8
     1     4     7
>> rot90(a,2)
ans =
     9     8     7
     6     5     4
     3     2     1
>> circshift(a,[-1 1])
ans =
     6     4     5
     9     7     8
     3     1     2
```

Built-in functions

- `diag` - Diagonal matrices and diagonals of matrix.
- `tril` - Extract lower triangular part.
- `triu` - Extract upper triangular part.

```
a =
      1      2      3
      4      5      6
      7      8      9
>> diag(a)
ans =
      1
      5
      9
>> diag(ans)
ans =
      1      0      0
      0      5      0
      0      0      9
>> triu(a)
ans =
      1      2      3
      0      5      6
      0      0      9
>> tril(a)
ans =
      1      0      0
      4      5      0
      7      8      9
```


Matrix analysis functions

```
help matfun
```

cond	Condition number with respect to inversion
condeig	Condition number with respect to eigenvalues
det	Matrix determinant
norm	Vector and matrix norms
normest	2-norm estimate
null	Null space
orth	Range space of matrix
rank	Rank of matrix
rcond	Matrix reciprocal condition number estimate
rref	Reduced row echelon form
subspace	Angle between two subspaces
trace	Sum of diagonal elements

KU LEUVEN

Linear equations

`help matfun`

chol	Cholesky factorization
cholinc	Sparse incomplete Cholesky and Cholesky-Infinity factorizations
cond	Condition number with respect to inversion
condest	1-norm condition number estimate
funm	Evaluate general matrix function
ichol	Incomplete Cholesky factorization
ilu	Sparse incomplete LU factorization
inv	Matrix inverse
ldl	Block LDL' factorization for Hermitian indefinite matrices
linsolve	Solve linear system of equations
lscov	Least-squares solution in presence of known covariance
lsqnonneg	Solve nonnegative least-squares constraints problem
lu	LU matrix factorization
luinc	Sparse incomplete LU factorization
pinv	Moore-Penrose pseudoinverse of matrix
qr	Orthogonal-triangular decomposition
rcond	Matrix reciprocal condition number estimate

KU LEUVEN

Eigenvalues, singular values

`help matfun`

balance	Diagonal scaling to improve eigenvalue accuracy
cdf2rdf	Convert complex diagonal form to real block diagonal form
condeig	Condition number with respect to eigenvalues
eig	Eigenvalues and eigenvectors
eigs	Largest eigenvalues and eigenvectors of matrix
gsvd	Generalized singular value decomposition
hess	Hessenberg form of matrix
ordeig	Eigenvalues of quasitriangular matrices
ordqz	Reorder eigenvalues in QZ factorization
ordschur	Reorder eigenvalues in Schur factorization
poly	Polynomial with specified roots
polyeig	Polynomial eigenvalue problem
rsf2csf	Convert real Schur form to complex Schur form
schur	Schur decomposition
sqrtm	Matrix square root
ss2tf	Convert state-space filter parameters to transfer function form
svd	Singular value decomposition
svds	Find singular values and vectors

KU LEUVEN

Demo / recap

- *File: array_operations.mlx*
- Array operations vs matrix operations (linear algebra / 2D)