

EGR 106 – Week 4 – Math on Arrays

- Linear algebraic operations:
 - Multiplication
 - Division
- Row/column based operations
- Rest of chapter 3

Array Multiplication (Linear Algebra)

- In linear algebra, the matrix expression $F = A * B$ means

$$F(r,c) = \sum_k A(r,k) * B(k,c)$$

- Entries are **dot** products of rows of the first matrix with columns of the second

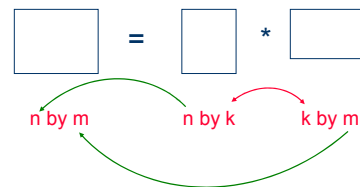


- For example:

$$\begin{bmatrix} 2 & 3 \\ 1 & 5 \end{bmatrix} * \begin{bmatrix} 1 & 4 \\ 2 & 6 \end{bmatrix} = \begin{bmatrix} 2*1+3*2 & 2*4+3*6 \\ 1*1+5*2 & 1*4+5*6 \end{bmatrix} = \begin{bmatrix} 8 & 26 \\ 11 & 34 \end{bmatrix}$$

- Notes:

- The operation is generally **not** commutative
 $A*B \neq B*A$
- The number of columns of the 1st **must** match the number of rows of the 2nd



- For example, here multiplication works both ways, but is not commutative:

```
A =      B =
  1  2      7  9  11
  3  4      8  10 12
  5  6

>> F = A * B      >> F = B * A
F =              F =
  23  29  35      89  116
  53  67  81      98  128
  83  105 127
```

← quite different!

- And here it doesn't work at all:

```
A =      B =
  1  2      7  8
  3  4      9  10
  5  6      11 12

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

Application of Multiplication

- Application of matrix multiplication: n simultaneous equations in m unknowns (the x 's)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1m}x_m &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2m}x_m &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nm}x_m &= b_n \end{aligned}$$

- For example: $2x_1 + 3x_2 + 3x_3 = 7$
 $4x_1 + 2x_2 + 9x_3 = 5$
 $6x_1 - 7x_2 + 2x_3 = 1$

- In matrix form this is $A * x = b$ with

$$A = \begin{bmatrix} 2 & 3 & 3 \\ 4 & 2 & 9 \\ 6 & -7 & 2 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} 7 \\ 5 \\ 1 \end{bmatrix}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1m}x_m &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2m}x_m &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nm}x_m &= b_n \end{aligned}$$

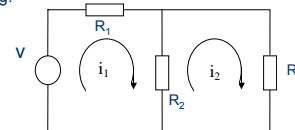
- In general: $A * x = b$

- A is n by m
- x is m by 1
- b is n by 1

column vectors
(lower case)

- Usages – finding:

- cable tensions in statics
- fluid flow in piping
- heat flow in thermodynamics
- currents in circuits
- traffic flow
- economics
- e.g.



$$\begin{bmatrix} R_1 + R_2 & -R_2 \\ R_2 & -(R_2 + R_3) \end{bmatrix} * \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} v \\ 0 \end{bmatrix}$$

Array Division

- Recall the command `eye(n)`

- This result is the array multiplication identity matrix I
- For any array A

$$A * I = I * A = A$$

must be properly sized!

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

- Imagine that for square arrays A and B we have

$$A * B = B * A = I$$

then we call them inverses

$$A = B^{-1} \quad B = A^{-1}$$

- In Matlab: `A ^ -1` or `inv(A)`

- When does A^{-1} exist?
 - A is square
 - A has a non-zero determinant ($\det(A)$)

```

• For example:
A =
    0     9     8
    7     4     5
    4     4     2
>> det(A)
ans =
    150
>> A*B
ans =
    1.0000     0     0
    0.0000     1.0000    -0.0000
    0.0000    -0.0000     1.0000
>> B = A^-1
B =
   -0.0800    0.0933    0.0867
    0.0400   -0.2133    0.3733
    0.0800    0.2400   -0.4200
>> B*A
ans =
    1.0000    0.0000    0.0000
     0     1.0000   -0.0000
     0     0.0000    1.0000

```

- Solving $A * x = b$
 - Assume that A is square and $\det(A) \neq 0$
 - Multiply both sides by A^{-1} on the left

$$A^{-1} * A * x = A^{-1} * b$$

$$= I$$

$$= x$$

SO $x = A^{-1} * b$

- In Matlab, $x = A \backslash b$ or $x = \text{inv}(A) * b$

← backwards slash

```

• For example:
A =
    0     9     8
    7     4     5
    4     4     2
b =
     6
     8
     0
>> x = A\b
x =
    0.2667
   -1.4667
    2.4000
• Check your work:
>> A*x
ans =
     6.0000
     8.0000
    -0.0000

```

General Linear Equation Solving (not in the book)

- Problem types:
 - overdetermined
 - underdetermined
- Solution methods:
 - Cramer's method
 - Gaussian elimination
 - inverse matrix
 - others
- Solution situations:
 - non-singular:
 - one unique solution
 - singular:
 - no solution
 - many solutions
- MatLab does them all

Vector Based Operations

- Some operations **analyze** a vector to yield a single value. For example:

```

A =
     6     3     5     1
>> sum(A)
ans =
    15

```

← sums the elements

- Other operations for a vector A:
 - Minimum: $\min(A)$
 - Maximum: $\max(A)$
 - Median: $\text{median}(A)$
 - Mean or average: $\text{mean}(A)$
 - Standard deviation: $\text{std}(A)$
 - Product of the elements: $\text{prod}(A)$

- Some operators yield two results:

- min and max can yield both the value and its location
- default is the first result

```
A =
     6     3     5     1
>> [value,location] = min(A)
value =
     1
location =
     4
```

- Some operators yield vector results

- size(A) we've already seen
- sort

```
A =
     6     3     5     1
>> sort(A)
ans =
     1     3     5     6
```

- Or multiple vectors:

```
A =
     6     3     5     1
>> [vals,locs] = sort(A)
vals =
     1     3     5     6
locs =
     4     2     3     1
```

- Finally, when applied to an array, these operators perform their action on columns

```
A =
     2     7     5     5
     3     3     3     4
     5     8     7     6
>> sum(A)
ans =
    10    18    15    15
```

- Unless you instruct it to work on rows!

```
A =
     2     7     5     5
     3     3     3     4
     5     8     7     6
>> sum(A,2)
ans =
    19
    13
    26
```

the 2 means "use the 2nd dimension" i.e. spanning the columns

- Use help to discover how to use these work

```
>> help max
MAX Largest component.
For vectors, MAX(X) is the largest element in X. For matrices,
MAX(X) is a row vector containing the maximum element from each
column. For N-D arrays, MAX(X) operates along the first
non-singleton dimension.
[Y,I] = MAX(X) returns the indices of the maximum values in vector I.
If the values along the first non-singleton dimension contain more
than one maximal element, the index of the first one is returned.
MAX(X,Y) returns an array the same size as X and Y with the
largest elements taken from X or Y. Either one can be a scalar.
[Y,I] = MAX(X,[],DIM) operates along the dimension DIM.
```