Array Operations

ENGR 151, Lecture 19: 12 Nov 14

Announcements

- Project 6 due tonight 11PM
- Project 7 out soon

Expressions in Array Elements

- Use arbitrary expressions in the construction of an array

```
a = [ 0  1+1  1+3*4 ];
b = [ a(2) a(1) a ];
```

is equivalent to

```
a = [ 0  2  13 ];
b = [ 2  0  0  2  13 ];
```

Character and String Literals

- Enclosed in single quotes

```
'M'
'Michigan'
'ann arbor'
'2014'    % not the number 2014
```

- String is a vector of characters

```
['a' 'n' 'n'] → 'ann'
```

What matrix is created by this MATLAB code?

```
a1 = [1; 2; 4; 8]
a1 = a1'
a2 = [a1(2) a1(3) a1(3)*2 a1(3)*4]
a4 = [6, 5, 4, 3]
A  = [a1; a2
      a4(3)-1, a4(3), a4(2), a4(1)
      a1(3), 2, a1(1), 0]
```

C

$$A = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 16 \\ 3 & 4 & 5 & 6 \\ 4 & 2 & 1 & 0 \end{bmatrix}$$

D

$$A = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 4 & 8 & 16 & 32 \\ 2 & 3 & 4 & 5 \\ 4 & 2 & 1 & 0 \end{bmatrix}$$

A

$$A = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 2 & 4 & 8 & 16 \\ 4 & 2 & 1 & 0 \\ 6 & 5 & 4 & 3 \end{bmatrix}$$

B

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 4 & 2 \\ 4 & 8 & 5 & 1 \\ 8 & 16 & 6 & 0 \end{bmatrix}$$

Which of the Following Vectors is Different?

A. `[ 10 20 30 40 50 60 ]`
B. `[ 10 : 10 : 60 ]`
C. `10:10:60`
D. `[ [10 20 30] [40 50 60] ]`
E. None of the above (all are the same)

Appending Arrays

- MATLAB *flattens* array sequences
- Examples:

```
[10 [20 30] [40 [50 60]]]
    → [10 20 30 40 50 60]
[1:3 1:3]   → [1 2 3 1 2 3]
[(1:3)'; (4:6)']  → (1:6)'
[(1:3)' (4:6)']   → [ 1 4
                      2 5
                      3 6 ]
```

Vectors with Constant Increments

- The colon operator provides a shortcut for vectors with regular spacing

  first : increment : last

  ```
  x = 1:2:8;
  ```

  is equivalent to

  ```
  x = [ 1  3  5  7 ];
  ```

- If increment is omitted, treated as 1
  - for example `1:4` is the vector `[1 2 3 4]`

Vectors with Constant Increments

- Works for chars/strings too:

```
'a':'f'      → 'abcdef'
'a':2:'k'    → 'acegik'
'A':'z'      →
```

Vectors in Index Expressions

- Suppose we assign: `a = [0 1 2 3 4 5 6 7 8 9]`
- What is `a([1 5])` ?

A. Error
B. `[0 4]`
C. `[0 1 2 3 4]`
D. `1:5`
E. None of the above

Vectors in Index Expressions

```
a = [0 1 2 3 4 5 6 7 8 9]

a(1) → 0
a(8) → 7
a(1:5) → [0 1 2 3 4]
a(4:10) → [3 4 5 6 7 8 9]
a(3:3:10) → [2 5 8]
```

Colon Operator for Matrices

```
M = [ 1:5; 3:3:15; 6:10 ]
     → [1  2  3  4  5
        3  6  9 12 15
        6  7  8  9 10 ]

M(2,4)     → 12
M(2,2:3)   → [6 9]
M(1:2,2:3) → [2 3; 6 9]
M(3,:)     → [6 7 8 9 10]
M(:,2)     → [2 6 7]'
```

Assigning to Index Ranges

```
M = [ 1:5; 3:3:15; 6:10 ]
    → [1  2  3  4  5
       3  6  9 12 15
       6  7  8  9 10 ]

M(2,2:3) = [66 99]
M(1:2,4:5) = -1
M(3,:) = 66:70
M(:,2) = [5 5 5]
```

Assigning Empty Array

- Suppose: `a = [0 1 2 3 4 5 6 7 8 9]`
- What is `a` after `a(3) = [ ]` ?

A. Error
B. `0:9`
C. `[0 1 3 4 5 6 7 8 9]`
D. `[ ]`
E. None of the above

Deleting Array Elements

- Assign empty array
- Examples:

```
v = 1:10
M = [1:5; 11:15; 21:25]

v(4) = []    % deletes fourth elt
v(3:5) = [] % deletes 3d through 5th elts
M(:,2) = [] % deletes 2d column
M(1,:) = [] % deletes 1st row
```

Built-In Matrix Creation Functions

`zeros(n)`	creates an n × n matrix of zeros
`zeros(n, m)`	creates an n × m matrix of zeros
`ones(n)`	creates an n × n matrix of ones
`ones(n, m)`	creates an n × m matrix of ones
`eye(n)`	creates an n × n identity matrix
`eye(n, m)`	creates an n × m matrix with `eye(min(n,m))` in upper left

Array Size Functions

`length(arr)`	returns the length of a vector or the longest dimension of a matrix
`size(arr)`	returns the rows and columns of an array

Using function that returns multiple values:

```
x = [1 2 3; 4 5 6];
[r c] = size(x); % 2 scalar results
s = size(x);     % result as vector
```

Create New Matrix based on Size of Another

```
x = [1 2 3; 4 5 6];
[r c] = size(x);
```

Equivalent:

```
y = zeros(r,c);
y = zeros([r c]);
y = zeros(size(x));
```

Exercise

- What is b after these assignments?

```
a = [7:2:11]'
b = [a a a]
```

A

7	9	11	7	9	11	7	9	11

B

7	7	7
9	9	9
11	11	11

C

7	9	11
7	9	11
7	9	11

D

7
9
11
7
9
11
7
9
11

E None of the above

Scalar Operations

- all binary, infix:

Addition	a + b
Subtraction	a – b
Multiplication	a * b
Division	a / b
Left Division	a \ b
Exponentiation	a ^ b

Scalar / Array Operations

- The same operations may also be applied to one array and one scalar (either order).
- In this case the operation is applied element-by-element to the array, holding the scalar fixed.
- For example:

```
B = [7 9 11; 13 15 17]
B + 6     → [13 15 17; 19 21 23]
```

Exercise

- What is b after these assignments?

```
b = 3 * eye(4);
b(1, 4) = 3;
b(4, 1) = 5;
```

A

3	0	0	3
0	3	3	0
0	5	3	0
5	0	0	3

B

3	0	0	5
0	3	5	0
0	3	3	0
3	0	0	3

C

3	0	0	5
0	3	0	0
0	0	3	0
3	0	0	3

D

3	0	0	3
0	3	0	0
0	0	3	0
5	0	0	3

E None of the above

Array vs. Matrix Operations

- Matrix operations are the standard Linear Algebra operations
- Array operations are performed on an element-by-element basis

Array Operations

- Array or Matrix operations (no difference)
  - Addition `a + b`
  - Subtraction `a - b`
- Array (element-by-element) operations
  - Multiplication `a .* b`
  - Right Division `a ./ b`
  - Left Division `a .\ b`
  - Exponentiation `a .^ b`

Using Element-by-Element Operations

- Calculate the following for z = 1,…,1000

$$y = \frac{z^3 + 5z}{4z^2 - 10}$$

- `z = 1:1000`
- `y = (z.^3 + 5*z) ./ (4*z.^2 - 10)`

Gilat Sec. 3.4

Matrix Operations

- Multiplication `a * b`
- Right Division `a / b`
- Left Division `a \ b`
- Exponentiation `a ^ b`

- Multiplication is standard matrix multiplication.
  - # columns of `a` must equal # rows of `b`
- Division (conceptually) inverts the denominator matrix and multiplies by the numerator.
- Exponentiation: `a` must be square, `b` a scalar

Exercise

- Which matrix corresponds to the result of the operation?

```
a = [1 2 ; 5 -1 ]      b = [0 1; 1 2 ]
```

Q1: a + b Q2: a - b Q3: a .* b Q4: a * b

A

1	1
4	-3

B

0	2
5	-2

C

2	5
-1	3

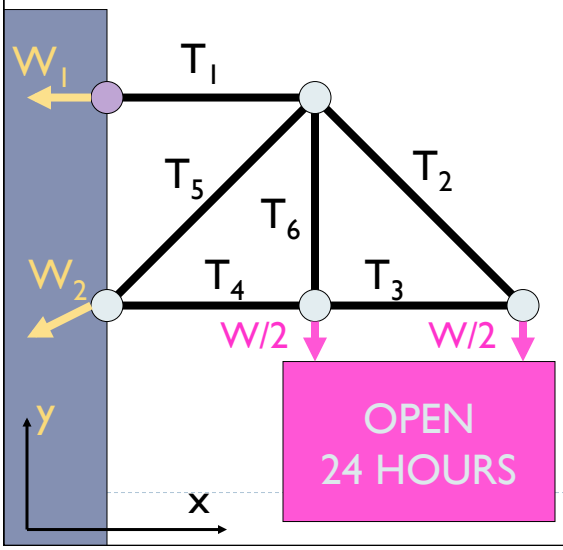
D

2	2
3	3

E

1	3
6	1

Recall Statics Problem



$$T_1 + W_{1x} = 0$$
$$W_{1y} = 0$$
$$-T_1 - T_5/\sqrt{2} + T_2/\sqrt{2} = 0$$
$$-T_5/\sqrt{2} - T_6 - T_2/\sqrt{2} = 0$$
$$W_{2x} + T_4 + T_5/\sqrt{2} = 0$$
$$W_{2y} + T_5/\sqrt{2} = 0$$
$$T_3 - T_4 = 0$$
$$-W/2 + T_6 = 0$$
$$-T_3 - T_2/\sqrt{2} = 0$$
$$-W/2 + T_2/\sqrt{2} = 0$$

Matrix Rep'n of Equations

General form:
$A\,x = b$

$$\begin{pmatrix} -1 & 1/\sqrt{2} & 0 & 0 & -1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/\sqrt{2} & 0 & 0 & -1/\sqrt{2} & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1/\sqrt{2} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1/\sqrt{2} & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ W_{1x} \\ W_{1y} \\ W_{2x} \\ W_{2y} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ W/2 \\ 0 \\ W/2 \end{pmatrix}$$

Solving Linear Equations in MATLAB

- Given a problem of the form:

  $A\,x = b$

where A is a matrix, b is a column vector, and we wish to derive values for the column vector x

- Derivation:

$$A\,x = b$$
$$A^{-1}A\,x = A^{-1}\,b$$
$$I\,x = x = A^{-1}\,b$$
$$x = A \setminus b$$

```
cs = 1/sqrt(2.0);
labels= ['T1 ';'T2 ';'T3 ';'T4 ';'T5 ';'T6 ';'W1x';'W1y';'W2x';'W2y'];
matrix = [ -1    cs    0     0     -cs   0     0     0     0     0   ;
           0     -cs   0     0     -cs   -1    0     0     0     0   ;
           0     0     0     1     cs    0     0     0     1     0   ;
           0     0     0     0     cs    0     0     0     0     1   ;
           1     0     0     0     0     0     1     0     0     0   ;
           0     0     0     0     0     0     0     1     0     0   ;
           0     -cs   -1    0     0     0     0     0     0     0   ;
           0     cs    0     0     0     0     0     0     0     0   ;
           0     0     1     -1    0     0     0     0     0     0   ;
           0     0     0     0     0     1     0     0     0     0 ];

W = 1.0;
b = [  0   0   0   0   0   0   0   W/2.0   0   W/2.0 ]';
sol = matrix\b;
strcat(labels, ' = ', num2str(sol))
```

char Function

- converts ASCII codes (integer between 0 and 127) into characters
- applies elt-by-elt to an array
- Examples:
  - `char(65:70)` → ABCDEF
  - `char([65:70]+1)` → BCDEFG
  - `3+'a'` → 100
  - `char(3+'a')` → d

Character Arithmetic

```
letters = char([0:25]+'a');
units = char(mod([1:26],10)+'0');
tens = char(floor([1:26]/10)+'0');

[tens; units; letters]
```

```
          00000000011111111112222222
          12345678901234567890123456
          abcdefghijklmnopqrstuvwxyz
```

Script Files

- A list of MATLAB commands, saved in a file
- Use extension .m
  - hence called "M-files"
  - can run scriptname.m by typing scriptname as command
- Running script file equivalent to typing in commands, in sequence

Defining Variables Used in Script Files

- in the command window prior to running the script file:

```
>> score1 = 50;
>> score2 = 75;
>> score3 = 100;

>> calculateAverage

average =
  75
```

file: `calculateAverage.m`

```
% This script file calculates the average
% of three values that are defined in the
% command window

    average=(score1+score2+score3) / 3
```

Defining Variables Used in Script Files

- Variables defined inside the script file remain "active" after the script file is finished

```
>> calculateAverage

average =
  75
```

file: `calculateAverage.m`

```
% This script file calculates the average
% of three values that are defined in the
% script file

    score1 = 50;
    score2 = 75;
    score3 = 100;

    average=(score1+score2+score3) / 3
```

Getting Data From the User

- input function: prompts user for input

```
val = input('Enter a value: ');
```

- The user may enter *any expression*, including:
  - a literal (e.g., numeric scalar, arrays in brackets)
  - a string (in quotes)
  - compound arithmetic expression
  - array-building expression
- To interpret *unquoted* input as a string, add second argument qualifier:

```
val = input('Enter a string: ', 's');
```

Input

- Ask for a value in the script using:

```
>> calculateAverage

Gimme score1!  50
Gimme score2!  75
Gimme score3!  100

average =
  75
```

file: `calculateAverage.m`

```
% This script file calculates the average
% of three values that are input by the user

    score1 = input('Gimme score1! ');
    score2 = input('Gimme score2! ');
    score3 = input('Gimme score3! ');

    average=(score1+score2+score3) / 3
```

disp Procedure

- Displays the value of an expression on the screen
- form: `disp(`*expr*`)`
- Example:

```
x = [5 : 9];
disp(x);
```

Output:

```
    5    6    7    8    9
```

display Procedure

- Like disp, but precedes output with associated variable and '='
- form: `display(`*expr*`)`
- Example:

```
x = 59;
display(x);
display(59);
```

Output:

```
x =
    59
ans =
    59
```

Read/Eval/Display Loop

- The MATLAB interpreter processes input (from keyboard or scripts) as follows:

1. read the next statement
   - a command, assignment, or expression
2. execute the statement, evaluating expressions as needed
3. if an assignment or expression stmt, call `display` on the result
   - unless statement ends in semicolon (`;`)