Additional Data Types: 2-D Arrays, Logical Arrays, Strings

Selim Aksoy
Bilkent University
Department of Computer Engineering
saksoy@cs.bilkent.edu.tr

2-D Arrays

- Recall matrices and subarrays
  - a = [1:2:7; 10:2:16]

```
a =
     1     3     5     7
    10    12    14    16
```

  - [x, y] = size(a)

```
x =
     2
y =
     4
```

  - a(2,:)

```
ans =
    10    12    14    16
```

  - a(:)

```
ans =
     1
    10
     3
    12
     5
    14
     7
    16
```


2-D Arrays

- Adding the elements of a matrix

```
function s = sum_elements(a)

[r,c] = size(a);
s = 0;
for ii = 1:r,
   for jj = 1:c,
      s = s + a(ii,jj);
   end
end
```


2-D Arrays

- Adding the elements of a matrix (cont.)
  - a = [1:2:7; 10:2:16];
  - sum(a)

```
ans =
    11    15    19    23
```

  - sum(a,1)

```
ans =
    11    15    19    23
```

  - sum(a,2)

```
ans =
    16
    52
```

  - sum(a(:))

```
ans =
    68
```


2-D Arrays

- Finding the maximum value in each row

```
function f = maxrow(a)

[r,c] = size(a);
f = zeros(r,1);

for ii = 1:r,
   m = a(ii,1);
   for jj = 2:c,
      if ( m < a(ii,jj) ),
         m = a(ii,jj);
      end
   end
   f(ii) = m;
end
```


2-D Arrays

- Finding the maximum value (cont.)
  - a = [1:2:7; 10:2:16];
  - max(a)

```
ans =
    10    12    14    16
```

  - max(a,[],1)

```
ans =
    10    12    14    16
```

  - max(a,[],2)

```
ans =
     7
    16
```

  - max(a(:))

```
ans =
    16
```

2-D Arrays

- Replace elements that are greater than t with the number t

```
function b = replace_elements(a,t)

[r,c] = size(a);
b = a;
for ii = 1:r,
    for jj = 1:c,
        if ( a(ii,jj) > t ),
            b(ii,jj) = t;
        end
    end
end
```


Logical Arrays

- Created by relational and logical operators
- Can be used as masks for arithmetic operations
- A mask is an array that selects the elements of another array so that the operation is applied to the selected elements but not to the remaining elements

Logical Arrays

- Examples
  - b = [1 2 3; 4 5 6; 7 8 9]

```
b =
     1     2     3
     4     5     6
     7     8     9
```

  - c = b > 5

```
c =
     0     0     0
     0     0     1
     1     1     1
```

  - whos

```
  Name        Size              Bytes  Class

  a           2x4                  64  double array
  b           3x3                  72  double array
  c           3x3                  72  double array (logical)
```


Logical Arrays

- Examples
  - b(c) = sqrt(b(c))

```
b =
    1.0000    2.0000    3.0000
    4.0000    5.0000    2.4495
    2.6458    2.8284    3.0000
```

  - b(~c) = b(~c).^2

```
b =
    1.0000    4.0000    9.0000
   16.0000   25.0000    2.4495
    2.6458    2.8284    3.0000
```


Logical Arrays

- Examples
  - c = b((b > 2) & (b < 8))

```
c =
     4
     7
     5
     3
     6
```

  - length(b((b > 2) & (b < 8)))

```
ans =
     5
```


Logical Arrays

- Examples
  - [r,c] = find((b > 2) & (b < 8))

```
r =
     2
     3
     2
     1
     2
c =
     1
     1
     2
     3
     3
```

Matrix Operations

- Scalar-matrix operations
  - a = [1 2; 3 4]

    a =
    ```
    1   2
    3   4
    ```
  - 2 * a

    ans =
    ```
    2   4
    6   8
    ```
  - a + 4

    ans =
    ```
    5   6
    7   8
    ```
- Element -by-element operations
  - a = [1 0; 2 1];
  - b = [-1 2; 0 1];
  - a + b

    ans =
    ```
    0   2
    2   2
    ```
  - a .* b

    ans =
    ```
    -1   0
    0   1
    ```


Matrix Operations

- Matrix multiplication: C = A * B
- If
  - A is a p-by-q matrix
  - B is a q-by-r matrix

  then
  - C will be a p-by-r matrix where

$$C(i,j) = \sum_{k=1}^{q} A(i,k)B(k,j)$$

Matrix Operations

- Matrix multiplication: C = A * B

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$C = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}) & (a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}) \\ (a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}) & (a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}) \\ (a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31}) & (a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32}) \end{bmatrix}$$

Matrix Operations

- Examples
  - a = [1 2; 3 4]

    a =
    ```
    1   2
    3   4
    ```
  - b = [-1 3; 2 -1]

    b =
    ```
    -1   3
    2   -1
    ```
  - a .* b

    ans =
    ```
    -1   6
    6   -4
    ```
  - a * b

    ans =
    ```
    3   1
    5   5
    ```


Matrix Operations

- Examples
  - a = [1 4 2; 5 7 3; 9 1 6]

    a =
    ```
    1   4   2
    5   7   3
    9   1   6
    ```
  - b = [6 1; 2 5; 7 3]

    b =
    ```
    6   1
    2   5
    7   3
    ```
  - c = a * b

    c =
    ```
    28   27
    65   49
    98   32
    ```
  - d = b * a

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

Matrix Operations

- Identity matrix: I
- A * I = I * A = A
- Examples
  - a = [1 4 2; 5 7 3; 9 1 6];
  - I = eye(3)

    I =
    ```
    1   0   0
    0   1   0
    0   0   1
    ```
  - a * I

    ans =
    ```
    1   4   2
    5   7   3
    9   1   6
    ```

Matrix Operations

- Inverse of a matrix: A^{-1}
- $A\,A^{-1} = A^{-1}\,A = I$
- Examples
  - a = [1 4 2; 5 7 3; 9 1 6];
  - b = inv(a)

```
b =
   -0.4382    0.2472    0.0225
    0.0337    0.1348   -0.0787
    0.6517   -0.3933    0.1461
```

  - a * b

```
ans =
    1.0000         0         0
    0.0000    1.0000         0
         0   -0.0000    1.0000
```


Matrix Operations

- Matrix left division: C = A \ B
- Used to solve the matrix equation A X = B where $X = A^{-1}\,B$
- In MATLAB, you can write
  - x = inv(a) * b
  - x = a \ b

  (second version is recommended)

Matrix Operations

- Example: Solving a system of linear equations

$$\begin{aligned} 4x - 2y + 6z &= 8 \\ 2x + 8y + 2z &= 4 \\ 6x + 10y + 3z &= 0 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} 4 & -2 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix}$$

  - A = [4 -2 6; 2 8 2; 6 10 3];
  - B = [8 4 0]';
  - X = A \ B

```
X =
   -1.8049
    0.2927
    2.6341
```


Matrix Operations

- Matrix right division: C = A / B
- Used to solve the matrix equation X A = B where $X = B\,A^{-1}$
- In MATLAB, you can write
  - x = b * inv(a)
  - x = b / a

  (second version is recommended)

Strings

- A string is an array of characters
  - s = 'abc'

    is equivalent to s = ['a' 'b' 'c']
- All operations that apply to vectors and arrays can be used together with strings as well
  - s(1) → 'a'
  - s([1 2]) = 'XX' → s = 'XXc'
  - s(end) → 'c'

String Conversion

- Conversion of strings to numerical arrays
  - double('abc xyz')

```
ans =
    97    98    99    32   120   121   122
```

  - double('ABC XYZ')

```
ans =
    65    66    67    32    88    89    90
```

- Conversion of numerical arrays to strings
  - char([72 101 108 108 111 33])

```
ans =
Hello!
```

Character Arrays

- 2-D character arrays
  - s = ['my first string'; 'my second string']
    ??? Error
  - s = char('my first string', 'my second string')
    s =
    my first string
    my second string
    } char function automatically pads strings
  - size(s) → [2 16]
  - size(deblank(s(1,:))) → [1 15]

String Tests

- ischar() : returns 1 for a character array
  - ischar ('CS 111')
    ans =
    1
- isletter() : returns 1 for letters of the alphabet
  - isletter('CS 111')
    ans =
    1 1 0 0 0 0
- isspace() : returns 1 for whitespace (blank, tab, new line)
  - isspace('CS 111')
    ans =
    0 0 1 0 0 0

String Comparison

- Comparing two characters
  - 'a' < 'e'
    ans =
    1
- Comparing two strings character by character
  - 'fate' == 'cake'
    ans =
    0 1 0 1
  - 'fate' > 'cake'
    ans =
    1 0 1 0

String Comparison

- strcmp() : returns 1 if two strings are identical
  - a = 'Bilkent';
  - strcmp(a, 'Bilkent')
    ans =
    1
  - strcmp('Hello', 'hello')
    ans =
    0
- strcmpi() : returns 1 if two strings are identical ignoring case
  - strcmpi('Hello', 'hello')
    ans =
    1

String Case Conversion

- Lowercase-to-uppercase
  - a = upper('This is test 1!')
    a =
    THIS IS TEST 1!
- Uppercase-to-lowercase
  - a = lower('This is test 1!')
    a =
    this is test 1!

Searching in Strings

- findstr() : finds one string within another one
  - test = 'This is a test!';
  - pos = findstr(test, 'is')
    pos =
    3 6
  - pos = findstr(test, ' ')
    pos =
    5 8 10

Searching in Strings

- strtok() : finds a token in a string
  - [token, remainder] = strtok('This is a test!', ' ')

    token =
    This
    remainder =
     is a test!
  - remainder = 'This is a test!';
    while (any(remainder)),
        [word, remainder] = strtok(remainder);
        disp(word);
    end

Replacing in Strings

- strrep() : replaces one string with another
  - s1 = 'This is a good example';
  - s2 = strrep(s1, 'good', 'great')

    s2 =
    This is a great example

String Conversion

- Recall num2str() for numeric-to-string conversion
  - str = ['Plot for x = ' num2str(10.3)]

    str =
    Plot for x = 10.3
- str2num() : converts strings containing numbers to numeric form
  - x = str2num('3.1415')

    x =
      3.1415

String Conversion

- sprintf() is identical to fprintf() but output is a string
  - str = sprintf('Plot for angle = %0.4f', pi)

    str =
    Plot for angle = 3.1416

String Comparison

- Example: Write a function that takes two strings and returns
  - -1 if the first string is lexicographically less than the second
  - 0 if they are equal
  - +1 if the first string is lexicographically greater than the second
- Pseudocode:
  - Get input strings
  - Pad strings to equal length
  - Compare characters from beginning to end and find the first difference
  - Return a value depending on the first distance

String Comparison

```
function result = c_strcmp(str1,str2)
%C_STRCMP Compare strings like C function "strcmp"
% Function C_STRCMP compares two strings, and returns
% a -1 of str1 < str2, a 0 if str1 == str2, and a
% +1 if str1 > str2.

%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    10/18/98    S. J. Chapman        Original code

% Check for a legal number of input arguments.
msg = nargchk(2,2,nargin);
error(msg);

% Check to see if the arguments are strings
if ~(isstr(str1) & isstr(str2))
   error('Both str1 and str2 must both be strings!')
else
   % Pad strings
   strings = strvcat(str1,str2);
   % Compare strings
   diff = strings(1,:) -= strings(2,:);
   if sum(diff) == 0
      % Strings match, so return a zero!
      result = 0;
   else
      % Find first difference between strings
      ival = find(diff);
      if strings(1,ival(1)) > strings(2,ival(1))
         result = 1;
      else
         result = -1;
      end
   end
end
```