Arrays Revisited

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

Initializing Vectors

- colon operator
  - x = 1:2:10
    x =
    1 3 5 7 9
- linspace(x1,x2,N) generates a row vector of N linearly equally spaced points between x1 and x2
  - x = linspace(10,20,5)
    x =
    10.00 12.50 15.00 17.50 20.00
- logspace(x1,x2,N) can be used for logarithmically equally spaced points

Vector Input to Functions

- You can call many built-in functions with array inputs
- The function is applied to all elements of the array
- The result is an array with the same size as the input array

Vector Input to Functions

- Examples:
  - x = [3 -2 9 4 -5 6 2];
  - abs(x)
    ans =
    3 2 9 4 5 6 2
  - sin([0 pi/6 pi/4 pi/3 pi/2])
    ans =
    0 0.5000 0.7071 0.8660 1.0000
  - a = 1:5;
  - log(a)
    ans =
    0 0.6931 1.0986 1.3863 1.6094

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(:)
    ans =
    1
    10
    3
    12
    5
    14
    7
    16

2-D Arrays

- Adding the elements of a matrix

```
[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(sum(a))
    ```
    ans =
        68
    ```
  - sum(a(:))
    ```
    ans =
        68
    ```


2-D Arrays

- Finding the maximum value in each row

```
[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 in each row (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(max(a))
    ```
    ans =
        16
    ```
  - max(a(:))
    ```
    ans =
        16
    ```


2-D Arrays

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

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


2-D Arrays

- Replace elements that are greater than t with the number t (cont.)

```
a( a > t ) = t
```

(“a > t” is a logical array)

Array Operations

- Scalar-array operations
  - x = 1:5
    ```
    x =
        1    2    3    4    5
    ```
  - y = 2 * x ← scalar multiplication
    ```
    y =
        2    4    6    8    10
    ```
  - z = x + 10 ← scalar addition
    ```
    z =
        11    12    13    14    15
    ```

Array Operations

- Array-array operations (element-by-element operations)
  - x = [1 2 3 4 5];   y = [2 -1 4 3 -2];
  - z = x + y

    z =
        3   1   7   7   3
  - z = x .* y

    z =
        2   -2   12   12   -10
  - z = x ./ y

    z =
        0.5000   -2.0000   0.7500   1.3333   -2.5000

Array Operations

- Array-array operations (element-by-element operations)
  - z = x .^ y
  - z =
        1.00   0.50   81.00   64.00   0.04
- Use .*, ./, .^ for element-by-element operations
- Array dimensions must be the same

Loops vs. Vectorization

- Problem: Find the maximum value in a vector
  - Soln. 1: Write a loop
  - Soln. 2: Use the built-in function “max”
- Use built-in MATLAB functions as much as possible instead of reimplementing them

Loops vs. Vectorization

```
%Compares execution times of loops and vectors
%
%by Selim Aksoy, 7/3/2004

%Create a vector of random values
x = rand(1,10000);

%Find the maximum value using a loop
tic;                %reset the time counter
m = 0;
for ii = 1:length(x),
    if ( x(ii) > m ),
        m = x(ii);
    end
end
t1 = toc;           %elapsed time since last call to tic

%Find the maximum using the built-in function
tic;                %reset the time counter
m = max(x);
t2 = toc;           %elapsed time since last call to tic

%Display timing results
fprintf( 'Timing for loop is %f\n', t1 );
fprintf( 'Timing for built-in function is %f\n', t2 );
```


Loops vs. Vectorization

- Problem: Compute $3x^2+4x+5$ for a given set of values
  - Soln. 1: Write a loop
  - Soln. 2: Use 3*x.^2 + 4*x + 5
- Allocate all arrays used in a loop before executing the loop
- If it is possible to implement a calculation either with a loop or using vectors, always use vectors

Loops vs. Vectorization

```
%Compares execution times of loops and vectors
%
%by Selim Aksoy, 7/3/2004

%Use a loop
tic;                %reset the time counter
clear y;
for x = 1:10000,
    y(x) = 3 * x^2 + 4 * x + 5;
end
t1 = toc;           %elapsed time since last call to tic

%Use a loop again but also initialize the result vector
tic;                %reset the time counter
clear y;
y = zeros(1,10000);
for x = 1:10000,
    y(x) = 3 * x^2 + 4 * x + 5;
end
t2 = toc;           %elapsed time since last call to tic

%Use vector operations
tic;                %reset the time counter
clear y;
x = 1:10000;
y = 3 * x.^2 + 4 * x + 5;
t3 = toc;           %elapsed time since last call to tic

%Display timing results
fprintf( 'Timing for uninizialed vector is %f\n', t1 );
fprintf( 'Timing for inizialed vector is %f\n', t2 );
fprintf( 'Timing for vectorization is %f\n', t3 );
```

Matrix Operations

- Transpose operator → '
  - u = [1:3]'
    ```
    u =
        1
        2
        3
    ```
  - v = [u u]
    ```
    v =
        1   1
        2   2
        3   3
    ```
  - v = [u'; u']
    ```
    v =
        1   2   3
        1   2   3
    ```


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)