

Vectorized Code, Indexing

Operations on Vectors and Matrices

- When you can, avoid for loops for operating on vectors and matrices.
- E.g., multiply each element in matrix `m` with 3:
- For loop version:

```
[R,C]=size(m);  
for i=1:R  
    for j=1:C  
        m(i,j) = m(i,j) * 3;  
    end  
end
```
- Vectorized code:

```
m = m * 3;
```

Logical Vectors

```
>> vec = [ 1 5 8 9 2 7 ];
```

```
>> isg = vec > 4
```

```
>> doubleres = isg + 5
```

```
>> whos
```

Name	Size	Bytes	Class
doubres	1x6	48	double array
isg	1x6	6	logical array
vec	1x6	48	double array

Logical Indexing

- A logical vector/matrix can be used to index (select) elements of a vector/matrix.
 - The logical vector/matrix should have the same size as the vector/matrix being indexed.

```
>> vec = [ 1 5 8 9 2 7 ];
```

```
>> isg = vec > 4
```

```
>> vec(isg)
```

```
>> vec(isg) = vec(isg) + 1
```

Exercise

- $a = [5 \ 2 \ 1 \ 7 \ 9 \ 6 \ 8]$;
- Create another matrix b that contains the even elements of a .
- Add 1 to odd elements of a .
- Remove all elements of a that are divisible by 3.
- Repeat steps above for $a=[5 \ 2 \ 1; 7 \ 9 \ 6]$;

Beware of 0/1.

```
>> a = [ 5 2 1 7 9 6 ];
```

```
>> a ( [1 0 1 0 1 0] )
```

??? Subscript indices must either be real positive integers or logicals.

```
>> a( logical( [1 0 1 0 1 0] ) )
```

```
>> a( [true false true false true false] )
```

find

- `find(x)` function returns the indices of non-zero elements of `x`.

```
>> find ( [ 3 0 2 5 0 ] )
```

```
>> find ( [ false true false true ] )
```

```
2 4
```

```
>> vec = [ 1 5 8 9 2 7 ];
```

```
>> find ( vec > 5 )
```

Equivalence of logical indexing & indexing with find()

- Assuming I is a logical vector/matrix, the following expressions are equivalent:
 - $m(I)$
 - $m(\text{find}(I))$

Logical Functions

```
>> vec1 = [1 3 1 1 2];
```

```
>> any ( vec1 )
```

```
>> all ( vec1 )
```

```
>> vec2 = [ 1 1 0 1 ]
```

```
>> any ( vec2 )
```

```
>> all ( vec2 )
```

Element-wise and/or

```
>> v1 = [ 3 0 5 1 ];
```

```
>> v2 = [ 0 0 -5 0 ];
```

```
>> v1 & v2
```

```
>> v1 | v2
```

Table 5.1 Operator Precedence Rules

Operators	Precedence
parentheses ()	highest
transpose and power ', ^, .^	
unary: negation (−), not (~)	
multiplication, division *, /, \, .*, ./, .\	
addition, subtraction +, −	
colon operator :	
relational <, <=, >, >=, ==, ~=	
elementwise and &	
elementwise or	
and &&	
or	
assignment =	lowest

Programming Concept

- When you want to operate on elements of a matrix that satisfy a condition, first construct an Index for elements that satisfy the condition.

Exercise

- Let $A = \text{magic}(4)$
- Replace with 0, all elements of A that are odd and whose $\log()$ is greater than 2.

Exercise

- Determine which built-in Matlab function the code below is similar to.

```
function ret = xxx( m )  
ret = false;  
for i=1:numel(m)  
    if m(i)  
        ret = true;  
    end  
end
```

Exercise: Zero-crossing

- Find the indices of the elements of vector v where the numbers change sign. Hints: sign, diff, find

$$v = [5 \quad -1 \quad -2 \quad -1 \quad 3 \quad 8 \quad -2]$$

meshgrid

- meshgrid creates a mesh (all combinations) of coordinates

x =

1 2 3

1 2 3

y =

1 1 1

2 2 2

Example

- Make a 3D plot of the following function:

$$f(x, y) = x * \sin(x + 2 * y)$$

- Use `x=0:.1:5`, `y=0:.1:5`
- Use `plot3` and `surf` functions.

Revisiting any, all, mean, sum

```
>> mean( [ 5 7 4 9] )
```

```
>> mean( [ 5 7; 4 9] )
```

- Specify `dim=2` to operate on each row and generate a column vector:

```
>> mean( [ 5 7; 4 9] , 2)
```

Revisiting min,max

- min/max can return the index of the min/max elements in the matrix.

```
>> [x, I] = min( [ 5 7 4 9] )
```

```
>> [x, I] = min( [ 5 7; 4 9] )
```

- Second input argument has another meaning for min/max. Specify dim=2 as third argument when you need it.

```
>> min( [ 5 7; 4 9] , 2)
```

```
>> min( [ 5 7; 4 9] , [], 2)
```