4. Hafta - 1

İlişkisel ve mantıksal operatörler
Öncelik kuralları
Mantıksal indeksleme

`find` fonksiyonu

Program akış denetimi

`if` – durumları
`switch` – durumları

Örnekler:
parçalı fonksiyonlar, birim basamak fonksiyonu, gösterge fonksiyonu, sinc fonksiyonu, yankılar

İlişkisel ve mantıksal operatörler

İlişkisel ve mantıksal fonksiyonlar

```
find, logical, true, false, any, all

ischar, isequal, isfinite, isinf, isinteger,
islogical, isnan, isreal
```

```
>> doc is*          % tüm 'is' fonksiyonlarının listesi
>> help logical     % mantıksal ifadeye dönüştürür
>> help true        % mantıksal 1
>> help false       % mantıksal 0
>> help relop       % ilişkisel operatörler
>> help ops         % help // ile aynı
>> help find        % sıfır olmayan elemanların indisleri
```

```
>> help precedence
```

İlişkisel operatörler

```
==      eşittir
~=      eşit değildir
<       küçüktür
>       büyüktür
<=      küçük eşittir
>=      büyük eşittir
```

Mantıksal operatörler

```
&       mantıksal AND,  örneğin, A&B, A,B=ifadeler
|       mantıksal OR,   örneğin, A|B
~       mantıksal NOT,  örneğin, ~A
&&      skalerler için mantıksal AND
||      skalerler için mantıksal OR
xor     exclusive OR,   örneğin, xor(A,B)
any     herhangi bir eleman sıfır değilse doğru
all     hiçbir eleman sıfır değilse doğru
```

MATLAB'da işlem önceliği (en yüksekten en düşüğe)

1. transpoze (.'), üs (.*), eşlenik transpoze ('), matris üs (^)
2. tekli toplama (+), tekli çıkarma (-), mantıksal olumsuzluk (~)
3. çarpma (.*), sağ bölme (./), sol bölme (.\), matris çarpım (*), matris sağ bölme (/), matris sol bölme (\)
4. toplama (+), çıkarma (-)
5. kolon operatörü (:)
6. küçüktür (<), küçük eşittir (=<), büyüktür (>), büyük eşittir (>=), eşittir (==), eşit değildir (~=)
7. terim terime mantıksal AND (&)
8. terim terime mantıksal OR (|)
9. kısa yoldan mantıksal AND (&&)
10. kısa yoldan mantıksal OR (||)

```
>> help precedence
```

```
>> a = [1, 0, 2, -3, 7];
>> b = [3, 4, 2, -1, 7];
>> a == b
ans =
  1×5 logical array
   0   0   1   0   1
>> k = a == b
k =
  1×5 logical array
   0   0   1   0   1
>> class(k)
ans =
    'logical'
>> a(k)
ans =
     2     7
```

mantıksal indeksleme

```
>> a = [1, 0, 2, -3, 7];
>> b = [3, 4, 2, -1, 7];
>> a == b
ans =
  1×5 logical array
   0   0   1   0   1
>> k = a==b
k =
  1×5 logical array
   0   0   1   0   1
>> i = find(a==b)
i =
     3     5
>> a(i)
ans =
     2     7
```

find kullanımı

sıradan indeksleme

```
a(a==b), a(find(a==b))
```

```
>> a = [1, 0, 2, -3, 7];
>> b = [3, 4, 2, -1, 7];
>> a == b
ans =
  1×5 logical array
   0   0   1   0   1
>> a ~= b
ans =
  1×5 logical array
   1   1   0   1   0
>> i = find(a~=b)
i =
     1     2     4
>> a(i), b(i)
ans =
     1     0    -3
ans =
     3     4    -1
```

```
>> a = [1, 0, 2, -3, 7];
>> ~a
ans =
  1×5 logical array
   0   1   0   0   0
>> a == 0
ans =
  1×5 logical array
   0   1   0   0   0
>> i = find(~a)
i =
     2
```

a’nın sıfır olan girdilerini bulur.

```
>> a = [1, 0, 2, -3, 7];
>> a ~= 0
ans =
  1×5 logical array
   1   0   1   1   1
>> ~~a
ans =
  1×5 logical array
   1   0   1   1   1
>> logical(a)
ans =
  1×5 logical array
   1   0   1   1   1
>> i = find(a)
i =
     1     3     4     5
>> a(find(a))
ans =
     1     2    -3     7
```

a’nın sıfır olmayan girdilerini bulur.

```
>> a = [1, 0, 2, -3, 7];
>> b = [3, 4, 2, -1, 7];
>> a<b, a>=b
ans =
  1×5 logical array
   1   1   0   1   0
ans =
  1×5 logical array
   0   0   1   0   1
>> i = find(a<b)
i =
     1     2     4
>> a(a<b), a(find(a<b))
ans =
     1     0    -3
ans =
     1     0    -3
```

Durum 1: a ve b vektör

a ve b terim terime karşılaştırılır.

Durum 2: a ve b vektör ya da skaler

a'nın her elemanını b skaleri ile karşılaştırır.

```
>> a = [1, 0, 2, -3, 7];
>> b = 1;
>> a>=b
ans =
  1×5 logical array
   1   0   1   0   1
>> i = find(a>=b)
i =
     1     3     5
>> a(a>=b), a(find(a>=b)), a(a<b)
ans =
     1     2     7
ans =
     1     2     7
ans =
     0    -3
```

```
>> a = [1, 0, 2, -3, 7];
>> b = [3, 4, 2, -1, 7];
>> a>=1
ans =
  1×5 logical array
   1   0   1   0   1
>> b<=2
ans =
  1×5 logical array
   0   0   1   1   0
>> a>=1 & b<=2                    % mantıksal AND
ans =
  1×5 logical array
   0   0   1   0   0
>> a>=1 | b<=2                    % mantıksal OR
ans =
  1×5 logical array
   1   0   1   1   1
```

Mantıksal işlemler

Mantıksal indeksleme

```
>> a = [1, 3, 4, -3, 7];
>> k = (a>=2), i = find(a>=2)
k =
  1×5 logical array
   0   1   1   0   1
i =
     2     3     5
>> a(i), a(k)
ans =
     3     4     7
ans =
     3     4     7
>> n = [0 1 1 0 1]
n =
     0     1     1     0     1
>> a(n)
```

Array indices must be positive integers or logical values.

class(k) mantıksal

Mantıksal indeksleme

a(a>=2)

class(n) double, fakat n == k doğru

% ama not edelim ki, a(logical(n)) çalışır.

Mantıksal indeksleme üzerine daha fazlası

```
>> A = [3 4 nan; -5 inf 2]
A =
     3     4   NaN
    -5   Inf     2
>> k = isfinite(A)
k =
  2×3 logical array
   1   1   0
   1   0   1
>> A(k)            % sütun bazlı listeleme
ans =
     3
    -5
     4
     2
>> A(~k)=0         % sonlu olmayan girdileri sıfırlar
A =
     3     4     0
    -5     0     2
```

```
>> find(k)
ans =
     1
     2
     3
     6
```

```
>> [i,j] = find(k)
[i,j] =
     1   1
     2   1
     1   2
     2   3
```

```
>> A = [3 4 0; -5 5 2]
A =
     3     4     0
    -5     5     2
>> A>2
ans =
  2×3 logical array
   1   1   0
   0   1   0
>> k = find(A>2)
k =
     1
     3
     4
```

```
>> [i,j] = find(A>2)
[i,j] =
     1   1
     1   2
     2   2

>> A(find(A>2))
ans =
     3
     4
     5
```

```
>> K = ['1'    '2'    '3'
        '4'    '5'    '6'
        '7'    '8'    '9'
        '*'    '0'    '#'];
>> K=='8'
ans =
  4×3 logical array
   0   0   0
   0   0   0
   0   1   0
   0   0   0
>> [i,j] = find(K=='8')
i =
     3
j =
     2
>> q = find(K=='8')
q =
     7
```

find bir karakter matrisine de uygulanabilir.

K'nin her elemanını '8' ile karşılaştırır.

K'nin doğru elemanının yerini bulur.

'8' in yerinin i,j matris indisleri

q, K'de '8' 'in sütun bazlı indeksidir

```
>> A = [9 9 2      B = [7 1 7
        2 5 4           3 4 8
        9 8 9];         9 4 2];
```

```
>> A<B
ans =
  3×3 logical array
   0   0   1
   1   0   1
   0   0   0
```

```
>> find(A<B)
ans =
     2
     7
     8
```

```
>> [i,j] = find(A<B)
i =       j =
     2         1
     1         3
     2         3
```

```
>> A == 9
ans =
  3×3 logical array
   1   1   0
   0   0   0
   1   0   1
```

```
>> find(A==9)
ans =
     1
     3
     4
     9
```

```
>> A(A==9)=-9
A =
    -9    -9     2
     2     5     4
    -9     8    -9
```

any, all

```
>> A = [9 9 2      B = [7 1 7
        2 5 4           3 4 8
        9 8 9];         9 4 2];
```

```
>> any(A==2)
ans =
  1×3 logical array
   1   0   1
>> any(A==2,2)
ans =
  3×1 logical array
   1
   1
   0
```

```
>> all(A>B)
ans =
  1×3 logical array
   0   1   0
>> all(A>B,2)
ans =
  3×1 logical array
   0
   0
   0
```

```
>> A == B
ans =
  3×3 logical array
   0   0   0
   0   0   0
   1   0   0
>> any(A==B)
ans =
  1×3 logical array
   1   0   0
>> any(any(A==B))
ans =
  logical
   1
```

`any, all` **sütun bazlı işlem yapar ve ekstra argüman ile satır bazlı çalışabilir.**

```
>> all(all(A==B));
```

```
>> A = [36 -4 9; 16 9 -25], B=A;
A =
    36    -4     9
    16     9   -25
>> k = (B>=0)
k =
  2×3 logical array
   1   0   1
   1   1   0
>> B(k) = sqrt(B(k))
B =
     6    -4     3
     4     3   -25
>> B(~k) = -sqrt(-B(~k))
B =
     6    -2     3
     4     3    -5
```

Örnek:
Mutlak değerlerin karekökünü işaretleri koruyarak alalım.

Karakter dizilerini karşılaştırmak

String'ler karakterlerden oluşan dizilerdir. `s1==s2` koşulu `s1` ve `s2`'nin aynı uzunlukta olmasını gerektirir.

```
>> s1 = 'short'; s2 = 'shore';
>> s1==s1
ans =
  1×5 logical array
   1   1   1   1   1
>> s1==s2
ans =
  1×5 logical array
   1   1   1   1   0
>> s1 = 'short'; s2 = 'long';
>> s1==s2
Matrix dimensions must agree.
```

Karakter dizilerini karşılaştırmak

Aynı uzunlukta olmayan karakter dizilerinin karşılaştırılması için `strcmp` Kullanılır ve ikili bir sonuç çıktısı verir.

```
>> s1 = 'short'; s2 = 'shore';
>> strcmp(s1,s1)
ans =
  logical
   1
>> strcmp(s1,s2)
ans =
  logical
   0
>> s1 = 'short'; s2 = 'long';
>> strcmp(s1,s2)
ans =
  logical
   0
```

>> doc strcmp
>> doc strcmpi

Matris veya dizi içeriklerini karşılaştırmak için `isequal` kullanın, İkili bir sonuç çıktısı alacaksınız.

Program Akış Kontrolü

Program akışı aşağıdaki kontrol yapıları ile kontrol edilir:

1. for ... end % döngüler
2. while ... end
3. break, continue
4. if ... end % koşullu
5. if ... else ... end
6. if ... elseif ... else ... end
7. switch ... case ... otherwise ... end
8. return

for-döngüleri ve koşullu if'ler en çok kullanılan kontrol yapılarıdır.

if - durumları

if – durumlarının üç biçimi

```
if condition
    statements ...
end
```

```
if condition
    statements ...
else
    statements ...
end
```

```
if condition1
    statements ...
elseif condition2
    statements ...
elseif condition3
    statements ...
else
    statements ...
end
```

Birkaç `elseif` durumu olabilir, `elseif end`'e gerek duymaz.

Örnek

```
>> x = 1;
>> % x = 0/0
>> % x = 1/0

if isinf(x)
    disp('x sonsuz');
elseif isnan(x)
    disp('x bir sayi degil');
else
    disp('x sonlu bir sayi');
end

x sonlu bir sayi
% x bir sayi degil
% x sonsuz
```

switch - durumları

```
switch expression0

  case expression1
    statements ...

  case expression2
    statements ...

  otherwise
    statements ...

end
```

Önce expression0 değerlendirilir, ve eğer bunun değeri expression1, expression2 case'lerine uyuyorsa, uyan durum çalıştırılır.

Birkaç case durumu olabilir.

expression karşılaştırma kuralları:

sayılar: isequal(expression0, expression1)
string'ler: strcmp(expression0, expression1)

Örnek: Bir vektörün L_1, L_2 ve L_∞ normları

$$x = [x_1, x_2, ..., x_N]$$

$$\| x \|_1 = \sum_{n=1}^{N} | x_n |$$

$$\| x \|_2 = \sqrt{\sum_{n=1}^{N} | x_n |^2}$$

$$\| x \|_\infty = \max\left(| x_1 |, | x_2 |, ..., | x_N | \right)$$

$$d(x, y) = \| x - y \|$$

İki vektör veya matris arasında uzaklık ölçümü için kullanılır.

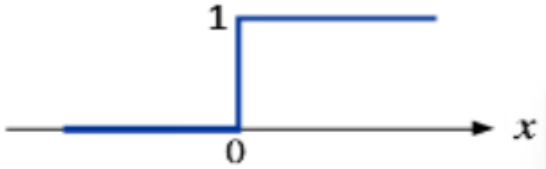
```
>> help norm        % vektör ve matris normları
```

```
x = [1, 4, -5, 3];
p = inf;
% p = 1;
% p = 2;

switch p
    case 1
        N = sum(abs(x));                % N = norm(x,1);
    case 2
        N = sqrt(sum(abs(x).^2));       % N = norm(x,2);
    case inf
        N = max(abs(x));                % N = norm(x,inf);
    otherwise
        N = sqrt(sum(abs(x).^2));       % N = norm(x,2)
end

>> N
N =
    5
```

norm hazır fonksiyonunu kullanarak eşdeğer hesaplamalar

Örnek: Birim basamak fonksiyonu

$$u(x)=\begin{cases}1, & x\geq 0\\ 0, & \textbf{diğer durumlar}\end{cases}$$

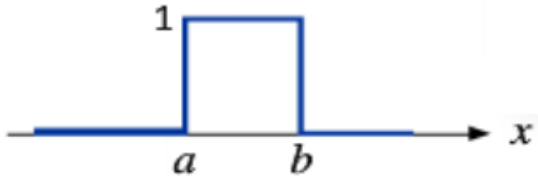



```
u = @(x) (x>=0);    % birim basamak fonksiyonu
```

örneğin x=-3, -2, -1, 0, 1, 2, 3
u(x)=0, 0, 0, 1, 1, 1, 1

Örnek: Gösterge fonksiyonu

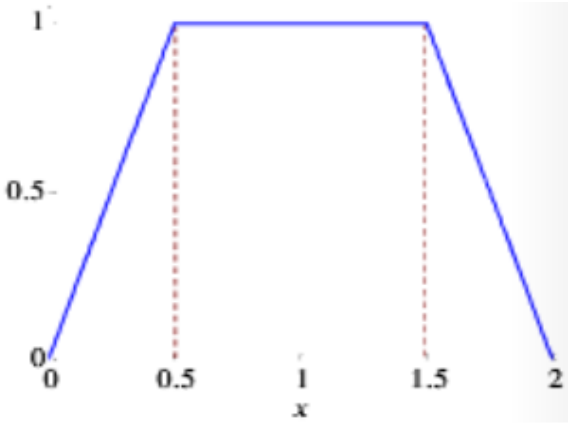
$$v(x,a,b)=u(x-a)-u(x-b)$$




```
v = @(x,a,b) u(x-a)-u(x-b);    % gösterge
% v = @(x,a,b) (x>=a & x<b);   % alternatif
```

Örnek: Parçalı fonksiyon tanımalamak (Metod 1)

$$f(x)=\begin{cases} 2x, & 0\le x\le 0.5 \\ 1, & 0.5\le x\le 1.5 \\ 4-2x, & 1.5\le x\le 2 \end{cases}$$



$$v(x,a,b)=\begin{cases} 1, & a\le x\le b \\ 0, & \textbf{diğer durumlar} \end{cases} = \textbf{(gösterge fonksiyonu)}$$

$$f(x)=2xv(x,0,0.5)+v(x,0.5,1.5)+(4-2x)v(x,15,2)$$

```
f = @(x) 2*x .* (x>=0 & x<0.5) + ...
                (x>=0.5 & x<1.5) + ...
        (4-2*x).* (x>=1.5 & x<2);

x = linspace(-0.5,2.5,301);
figure; plot(x,f(x),'b-');
```

Metod 1 – vektörleştirme
Metod 2 – vektörleştirme
Metod 3 – vektörleştirme değil