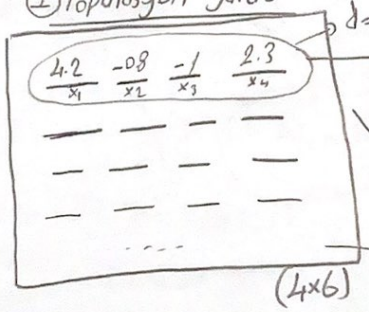
#MATLAB İLE GENETİK ALGORİTMA

→ $x_1^2+x_2^2+\dots+x_n^2$ gibi bir problemi genetik algoritma ile çözmeye çalışacağız.
→ Tavlama benzetiminden farkı, genetik algoritma → popülasyon bazlı çalışır
↳ yerel arama yapar.
↳ Ayrıca genetik algoritma yerel değil/ global arama yapmaktadır.

Current Folder → New File → Function → gaim

- Minimize edilecek problem: $(x_1^2+x_2^2+\dots+x_d^2)$
alt, üst sınır belirle? $-5 \le x_i \le 5$
d (Problem boyutu)? 2, 3, 100, 5?

① Popülasyon Yarat:



d=4 (problem boyutu, sütün sayısı)
→ Aday çözüm (kromozom)
· Genetik algoritma 1'den çok kromozom aday çözüm ile başlar.
→ Popülasyon büyüklüğü kadar kromozom (çözüm) var. (psize) satır sayısı
→ Popülasyon = kromozom topluluğu.
↳ Önce popülasyonu yarat.

(unifrnd, unitrnd, randperm) → Rastsal sayı yaratır.

(popülasyon oluştur)

```
population = unifrnd(as, us, [psize, d]);
```

- unifrnd: Uniform, sürekli rastsal sayı oluşturur
- unitrnd: Uniform, tam rastsal sayı üretir.
- randperm: 1,...,n arası sayıların rastsal permütasyonunu oluşturur.

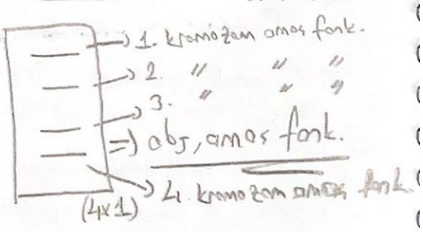
as: alt sınır.
us: üst sınır.
psize: popülasyon büyüklüğü (satır sayısı)
d: problem boyutu (sütün sayısı)

(Amaç Fonk Hesapla)

```
- obj = zeros(psize, 1);
```

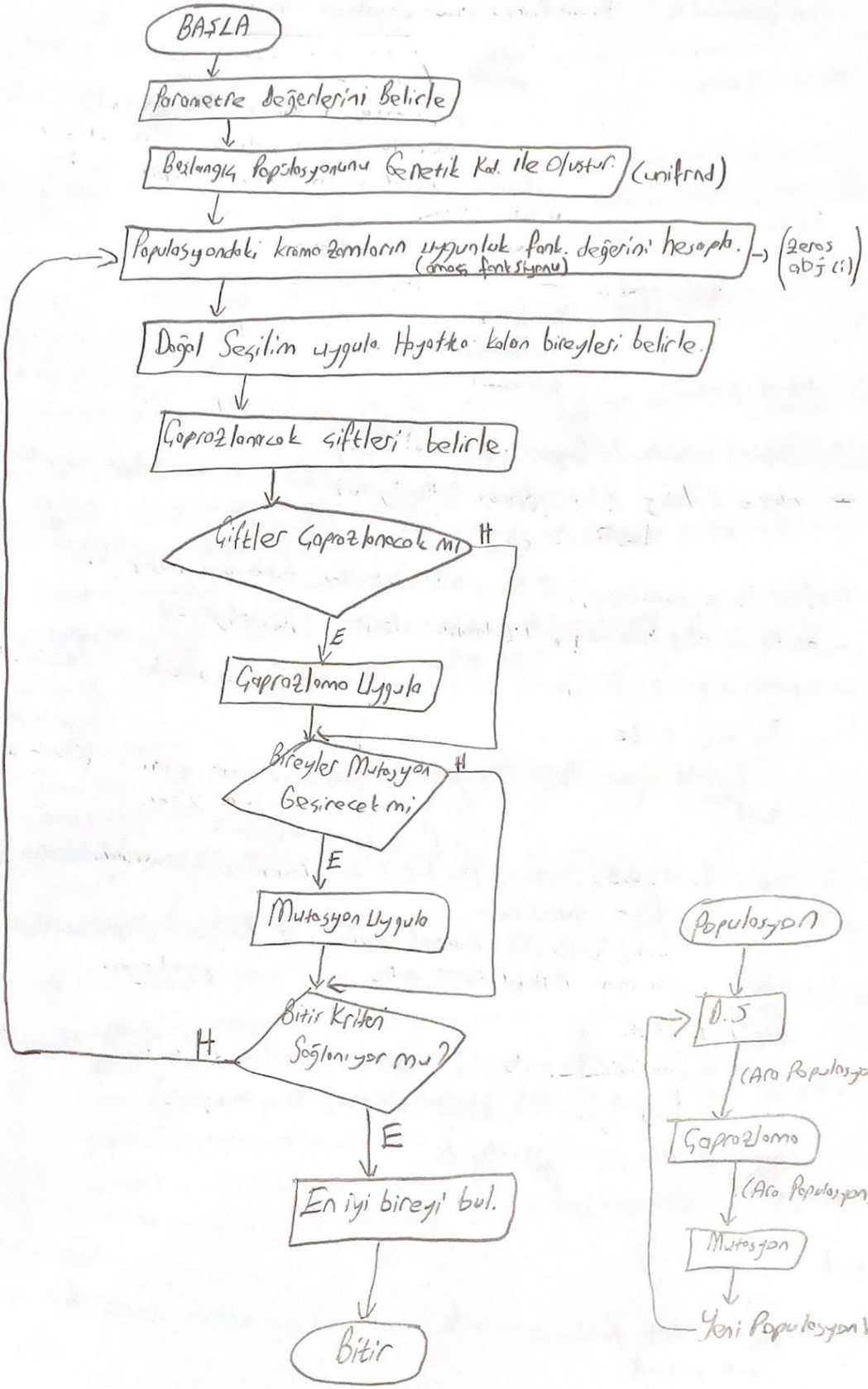
zeros(satır, sütün)
zeros(7) → 7×7 kare matris oluşturur.

obj = amaç fonksiyonu.
· psize çift sayı belirlenmelidir.




```
- for i=1:psize
    obj(i,1) = sum(population(i,:).^2);
  end
```

Her kromozomun elemanlarının karesini alıp, amaç fonk. yazıyoruz.


BAŞLA
Parametre değerlerini Belirle
Başlangıç Popülasyonunu Genetik Kod. ile Oluştur. (unifrnd)
Populasyondaki kromozomların uygunluk fonk. değerini hesapla. (amaç fonksiyonu)
(zeros adj(:))
Doğal Seçilim uygula. Hayatta kalan bireyleri belirle.
Çaprazlanacak çiftleri belirle.
Çiftler Çaprazlanacak mı
H
E
Çaprazlama Uygula
Bireyler Mutasyon Gesirecek mi
H
E
Mutasyon Uygula
Bitiş Kriteri Sağlanıyor mu?
H
E
En iyi bireyi bul.
Bitir
Populasyon
0.5
(Ara Populasyon)*
Çaprazlama
(Ara Populasyon)*
Mutasyon
Yeni Populasyon

→ current folder → New File → Function → dogalsecilim.m

Rulet Sarkı, Beklenen Değer... uygulanabilir

- iyi çözümlerin (bu problem için en küçük olan) hayatta kalması, diğerlerinin öldürülmesi.
- Rulet sarkı maksimi zasyona göre çalışır. Bundan dolayı elde ettiğimiz sonucu ters çevirerek sonuç elde edeceğiz.

• Rulet Sarkı:

	Uygunluk Değeri	Olasılık	Birikimli Olasılık (Kümülatif olasılık)
Kromozom 1	5	5/25 = 0,20	0 + 0.20 = 0,20
Kromozom 2	4	4/25 = 0.16	0.20 + 0.16 = 0.36
Kromozom 3	6	6/25 = 0.24	0.36 + 0.24 = 0.60
// 4	10	10/25 = 0.40	0.60 + 0.40 = 1.00
	25		

```
function [arapop] = dogalsecilim(population, obj, psize)
    - obj = 1./obj  // Rulet Sarkı kullandığımız için elde ettiğimiz sonuçları ters çevirdik.
                    └→ Artık problemimiz maximizasyon.
    - sumobj = sum(obj); // Uygunluk değerleri toplamını hesapladık
    - probs = obj/sumobj; // Olasılık değerini hesaplıyoruz.
    - cprobs = probs; // Birikimli olasılık sekli olasılık gibi olacak.
        for i=2:psize
            cprobs(i) = cprobs(i-1) + cprobs(i);
        end
      ⇒ 1. eleman aynı olduğu için 2'den başlattık.
    - rs = unifrnd(0,1,[psize, 1]); // 0-1 arası rastsal sayı oluşturduk. (4x1).
            └→ rs: rastsal sayı
            └→ Oluşturulan rastsal sayılar ile Birikimli olasılıkları karşılaştırıyoruz.
    - arapop = population; // Doğal Seçilimden sonraki ara popülasyon.
        for i=1:psize
            idx = find(rs(i) ≤ cprobs, 1) // Birikimli olasılığın rs'de küçük olduğu ilk indexi bul.
                  └→ Sayının ilk küçük olduğu index hangisi?
            arapop(i) = population(idx, :);
        end
end
```

0-0.20 = Kromozom 1
0.20-0.36 = // 2
0.36-0.60 = // 3
0.60-1 = // 4

NOT⇒ dogalsecilim fonksiyonunuda ga fonksiyonunun içerisine atıyoruz.

→ current folder → New File → Function → crossover.M

Çaprazlama işlemi için oluşturduk.

Çözümlerde çeşitliliği sağlar

[0.6, 1]

• Arapopulasyon elde ederek çaprazlanacak bireyleri belirlemiz olduk

Bu parametre çiftlerde çaprazlama olup olmayacağını söyler.

• Her çift için çaprazlama olasılığı parametresi atıyoruz.

• Çaprazlama olacaksa çaprazlama uygulanır ve 2 ebeveyden 2 çocuk elde edilerek mutasyona sokulur, çaprazlama olmayacaksa doğrudan mutasyona sokulur.

• Tek noktalı çaprazlama yapacağız ve rastsal bir çaprazlama noktası seçeceğiz.

↳ 11001011 + 11011111 = 11001111 + 11011011 (d=8)

• crossover.m 'deki fonk. için arapopulasyon, ve kaç tane çift olacağını belirleyeceğimiz psize değişkenlerini girdi olarak alacağız.

↳ Çiftleri buradan belirlediğimiz için, bundan dolayı psize çift sayı olmalı.

```
function [arapop] = crossover(arapop, psize, pcross, d)
- pairs = randperm(psize); // çiftleri belirliyoruz. Randperm tekrarsız şekilde rastgele çiftleri belirliyor.
                              ↳ psize/2 tane çaprazlama.
- for i = 1 : psize/2
    parent1idx = pairs(2*i-1); // 1. kromozom çifti indexi
    parent2idx = pairs(2*i);   // 2. kromozom çifti indexi
    (d=1 olduğu için toplamda 2 çift olacak)
    parent1 = arapop(parent1idx, :); // Çiftlerin 1. ebeveyni.
    parent2 = arapop(parent2idx, :); // Çiftlerin 2. "
    rs = unifrnd(0,1); // çaprazlama olasılığını bununla karşılaştıracağız.
                       ↳ rs < çaprazlama olasılığı → parentları çaprazlayacağız.
    if (rs < pcross)   (pcross: çaprazlama olasılığı)
        cpoint = unidrnd(d-1); // çaprazlama noktası seçiyoruz. (cpoint: çaprazlama noktası)
        dummy = parent1(cpoint+1:end);
        parent1(cpoint+1:end) = parent2(cpoint+1:end);
        parent2(cpoint+1:end) = dummy;
        Swap işlemi yaptık!
        arapop(parent1idx, :) = parent1;
        arapop(parent2idx, :) = parent2;
        → Çaprazlama sonrası ara populasyon.
    end
end
end
```

Current Folder → New File → Function → mutation.m

- Mutasyon fonk. olusturduk

- Mutasyon olasiligi [0.001, 0.01]
- Mutasyon gen basina yapilan bir seydir.

① İkili Kodlama = 1100 => 1009 (Gen Değişimi)

Swap ← ② Permütasyon Bazlı Kodlama = (1 2 3 4 5 6 7 8 9) => (1 8 3 4 5 6 7 2 9) (Gen çevrimi)

Biz Buna Bakacaz ← ③ Değer Bazlı Kodlama = (1.29 2.86 4.11 5.55) => (1.29 2.73 4.22 5.55) (Küçük Bir Sayı Ekleme Çikarma)

Tüm genler tek tek mutasyon olasiligi değeri ile kiyaslanacak

Mutasyon Olasiligi > Gen ⇒ Mutasyon Yap

Gen Değeri (-/+)(rastsal Sayı) * (komşuluk Payı) * (Aralık)

↳ Artıp - Azalacağında rastsal belirliyoruz.

```
function [arapop] = mutation (arapop, pmutation, psize, d, delta, us, as)
  - rs = unifrnd (0,1, [psize, d]); // 0-1 arası rastsal sayı olustur.
                  ↳ Buradaki sonuç ile mutasyon olasiligi kiyaslanacak.
    for i=1 : psize   // satir tarama.
      for j=1 : d     // sutün tarama.
        if (rs(i,j) < pmutation)  // (pmutation = mutasyon olasiligi)
          rs2 = unifrnd(-1, 1);
          arapop(i,j) = arapop(i,j) + rs2 * delta * (us - as);   (delta: komşuluk payı, Aralık = us - as)
        end
      end
    end
end
```

(Rastsal olarak küçük sayı Ekleme Çikarma)

```
function [eniyicozum, eniyideger objit] = ga(as,us,d, psize, pcross, pmutation, delta)
    population = unifrnd(as, us, [psize, d]);
    iteration = 1;
    eniyideger = 1000000;
    while (iteration < 50)
        obj = zeros(psize,1);
        for i = 1 : psize
            obj(i) = sum ( population(i,:).^2);
        end
    %En iyi değer kime onu bul ve nihai çözüm olarak al
        if(min(obj) < eniyideger)
            eniyideger = min(obj);
            idx = find(obj = eniyideger);
            eniyicozum = population(idx, :);
        end
    objit(iteration) = eniyideger;
    [arapop] = dogalsecilim (population, obj, psize);
    [arapop] = crossover (arapop, psize, pcross, d);
    population = mutation (arapop, pmutation, psize, d, delta, us, as);
    iteration = iteration +1;
    end
end
```

GA-6