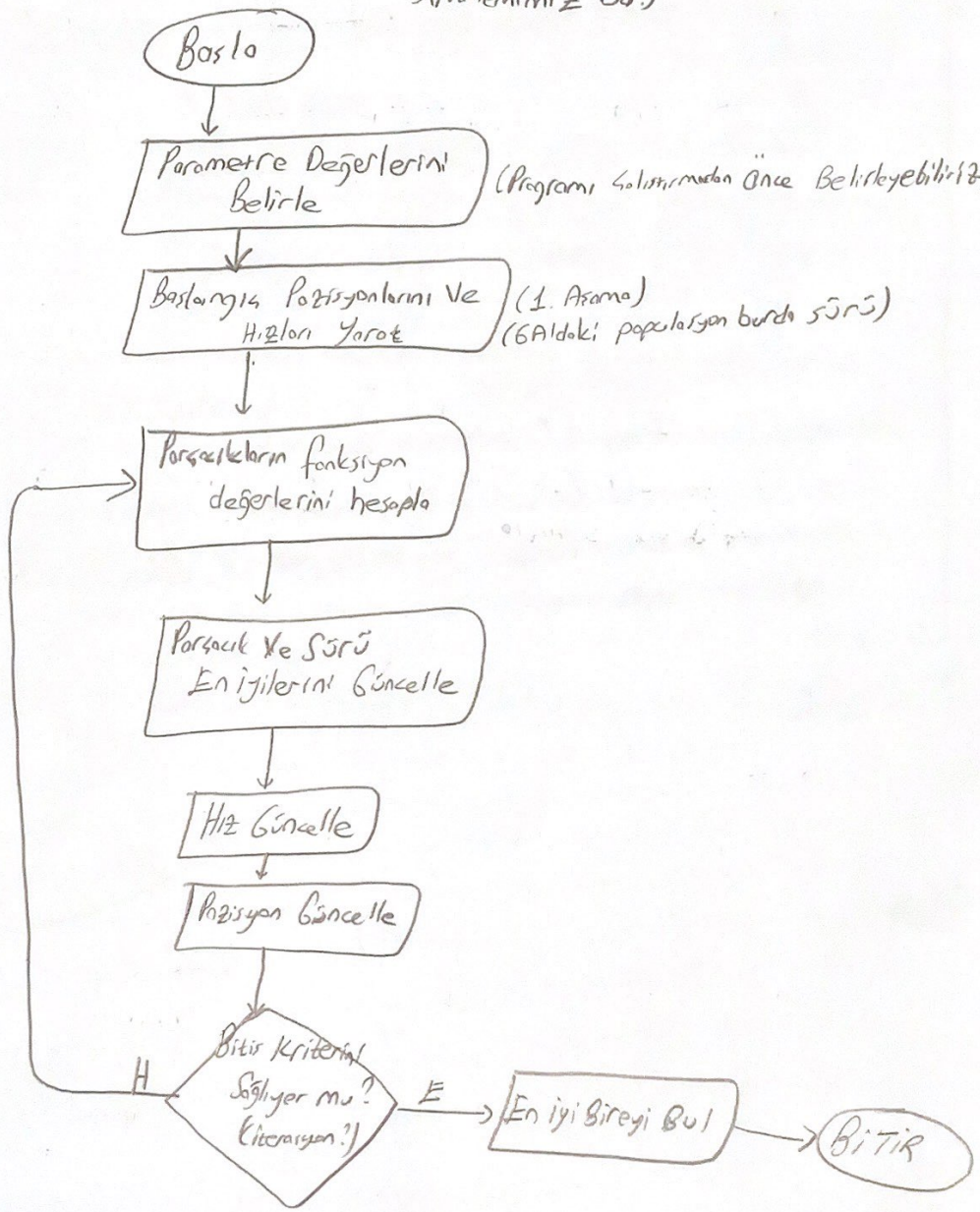
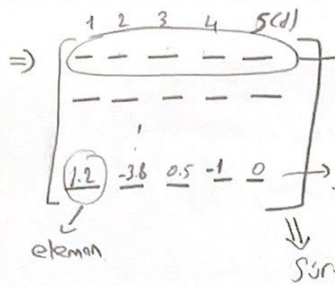
Matlabda PSO (Particle Swarm Optimization)
(Parçacık Sürü Optimizasyonu)

=) Current Folder → New File → Function → pso.m

- Çözeceğimiz problem $\left(\sum x_i^2,\ i = 1, 2, \ldots, d\right)$ minimizasyon
  ↳ (Problemimiz Bu.)



- function [sbestval sbestpos] = pso (as, us, d, ssize, w, c1, c2)
  ↓



⇒ 1 çözüm (Her satır çözümdür) /parçacık (Aslında pozisyonlar bu değerler)
(d=5) (as=alt sınır, us=üst sınır) → (simetrik olmak zorunda değil.)
(ssize=sürü boyutu) (as ≤ x_i ≤ us) /Sürekli
→ ssize. çözüm

satır sayısı, sütün sayısı

- swarm = Unifrnd(as, us, [ssize, d]); // Başlangıç sürümüzü belirledik.
→ Sürekli Uniform sürü oluşturduk.
↳ Bu kısmı çalıştırdıktan sonra workspace'i veriler diye kayıt edebilirsin.

- obj = $\begin{bmatrix} - \\ - \\ \vdots \\ - \end{bmatrix}$ = zeros(ssize, 1); // amaç fonksiyonunun kalıbını oluşturduk.
ssize x 1
↳ Toplama işlemi yaptığımız için toplam yaptık.

```
for i=1:ssize
    obj(i) = sum(sworm(i,:).^2);
end
```
// Amaç fonksiyonunu yazdık

→ Hız güncellemesi yapacağız. Ya başlangıç hızlarını 0 seçmeliyim yada çok küçük hızlar seçmeliyiz. Biz 0 hızla başlıyoruz.

$\begin{bmatrix} - & - & - & - & - \\ - & - & - & - & - \\ & & \vdots & & \\ - & - & - & - & - \end{bmatrix}$ = Velocity (Hız Değişkeni) → Her elemanın kendine göre hızı vardır. Hepsi tek tek güncellenmeli
(ssize x d)
d sütün

- Velocity = zeros(ssize, d); // Başlangıçları sıfır seçtik.

$$V_{ij} = w.V_{ij} + c_1.r_1.(x_{ij}^{pb} - x_{ij}) + c_2 r_2.(x_{ij}^{gb} - x_{ij}),$$

$r_{1,2}$ = rastgele sayı
$c_{1,2}$ = sabit sayı

- (parçacık hızını korumak istiyor. (Eylemsizlik))
- (Parçacık kendi en iyi halini hatırlıyor. pb = personal best)
- (Parçacık sürünün en iyisini hatırlıyor ve güncellemek istiyor. gb = global best)

↳ Bunların bileşkesi hızı oluşturuyor.

```
- pbestpos = swarm;
  pbestval = obj;

  sbestval = min(obj);
  idx = find(sbestval == obj); // index buluyoruz.
  sbestpos = sworm(idx,:); // Bu en iyi çözümümüzdür
```

1. iterasyon için en iyi çözümü bulduk.

Hız Güncelleş

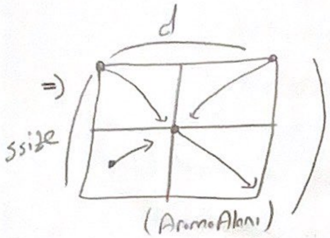
```
iteration = 1;      objfit = gbest val;
while(iteratin <= 50)
  for i=1, ssize
      Velocity(i,:) = w * Velocity(i,:) + c1 * unifrnd(0,1) * (pbestpos(i,:) - swarm(i,:))
                      + c2 * unifrnd(0,1) * (gbestpos(i,:) - swarm(i,:));
  end
```

Eylemsizlik Katsayısı (w)

Bilişsel Katsayı (c_1)

Sosyal Katsayı (c_2)

(Hız Güncelleme Formülü)



Parçacık Max. $(us - as)/2$ → hızına sahip olmalı.

↳ Eğer hızı çok yüksek olursa arama alanını aşabilir.

```
vmax = (us-as)/2;

for i=1: ssize
    for j=1: d
        if(velocity(i,j) > Vmax)
            velocity(i,j) = Vmax;
        elseif (velocity(i,j) < -Vmax)
            Velocity(i,j) = -Vmax;
        end
    end
end
```

" Hız sınırlaması Yaptık Parçacıklar Arama Alanı Dışına Çıkmasın Diye "

Pozisyon Güncelleş

```
swarm = swarm + velocity; // (Yeni pozisyon: Eski pozisyon + Hız)
```

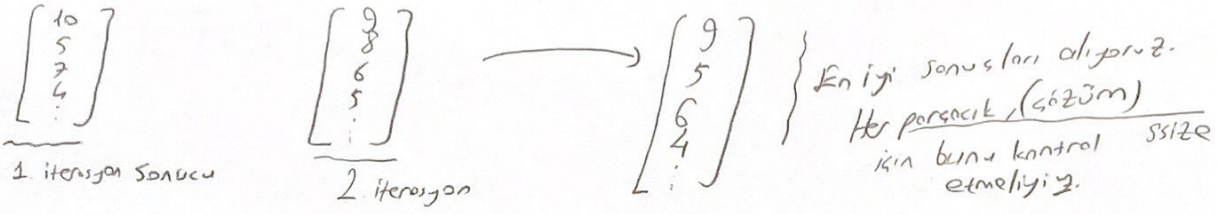
→ Pozisyon güncelle

```
for i=1: ssize
    for j=1: d
        if (swarm(i,j) > us)
            Swarm(i,j) = us;
        elseif(swarm(i,j) < as)
            Swarm(i,j) = as;
        end
    end
end
```

Pozisyon Sınırlaması Yaptık.

```
for i=1 : ssize
    objf(i) = sum(swarm(i,:).^2);
end
```

% Parçacık En iyisini Güncelle:




```
- for i=1:ssize
    if(obj(i) < pbestval(i))
        pbestval = obj(i) // Parçacığın şuana kadarki en iyi amaç fonk değeri
        pbestpos(i,:) = swarm(i,:); // Parçacığın şuana kadarki en iyi pozisyonu.
    end
end
```

% Sürünün En iyisin Güncellenmesi

```
- if(min(obj) < sbestval)
    sbestval = min(obj);
    idx = find(sbestval == obj);
    sbestpos = swarm(idx,:);
end
```

Aslında bunlar isteniliyor!!

```
iteration = iteration + 1;
obj_it(iteration) = sbestval; // En iyi değeri göreceğiz ve iterasyonlardaki ilerlemeleri görürüz.
end  → iterasyona giden end
end  → functiona giden end.
```

PSO-4