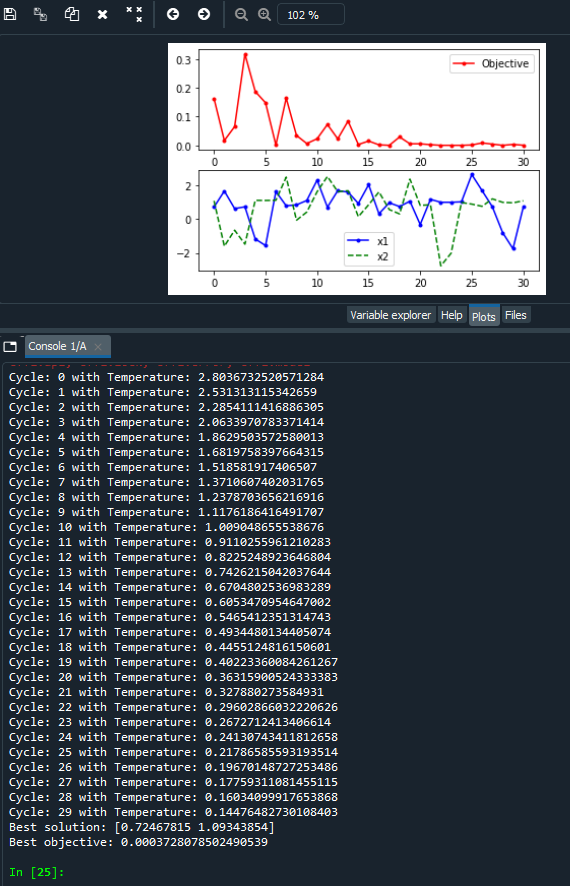
# Python ile kodlanmış “Simulated Annealing” ve”Whale Optimization algorithm” optimizasyon algoritmaları, “Ackley”, “Levi”, “Beale” ve “Goldstein - Price” fonksiyonları üzerinde uygulanmıştır. Optimizasyon sonuçları aşağıda verildiği gibi çıkmıştır

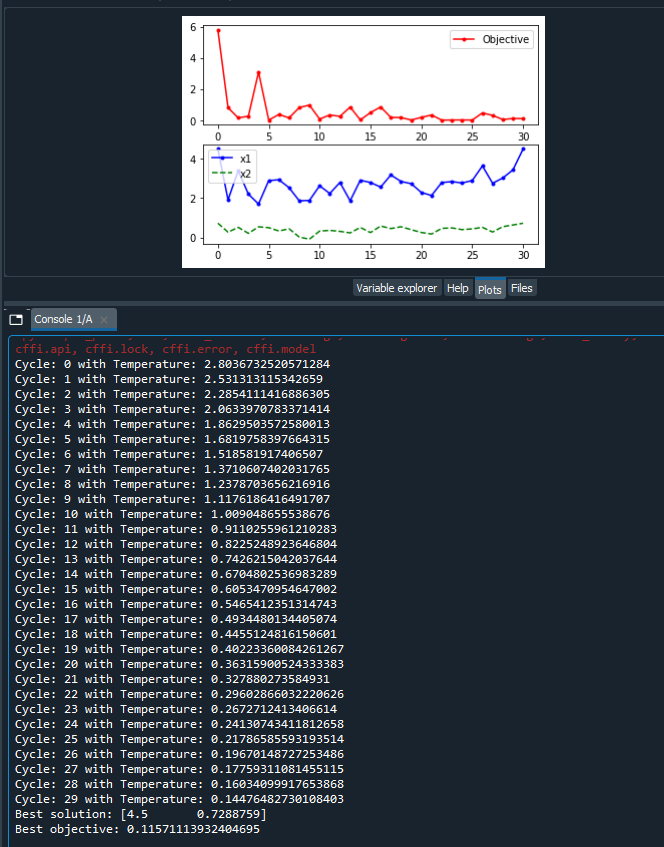
# **Simulated annealing - Ackley Function**



# **Simulated annealing - Levi Function**



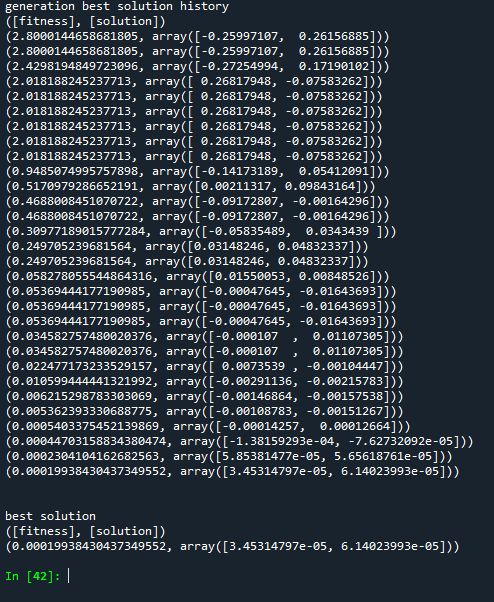
# **Simulated annealing - Beale Function**



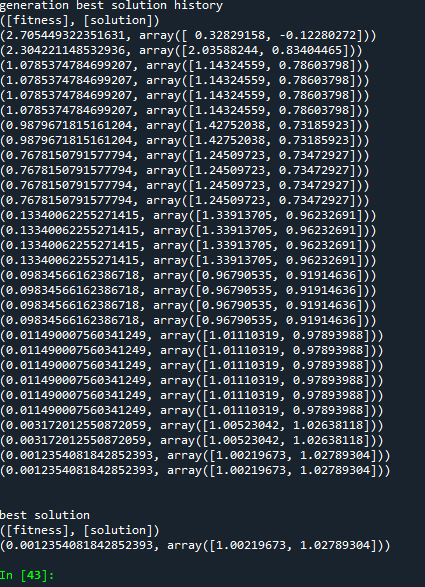
# **Simulated annealing - Goldstein Function**



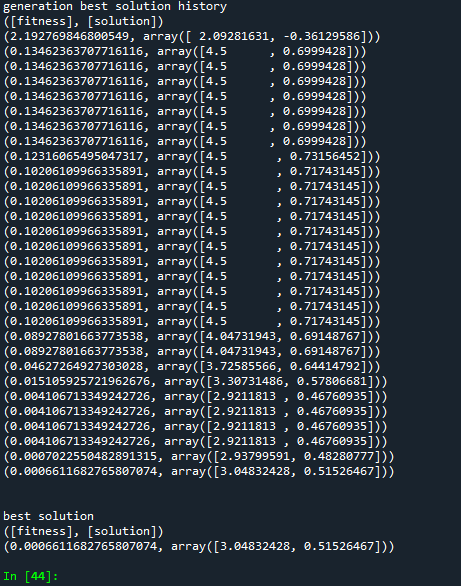
**Whale Optimization Algorithm - Ackley Function**



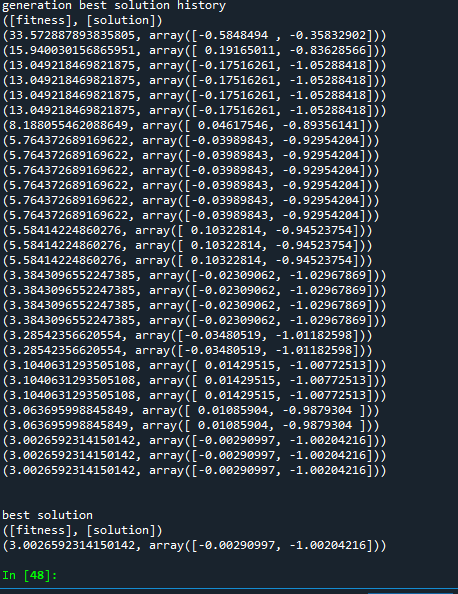
**Whale Optimization Algorithm - Levi Function**

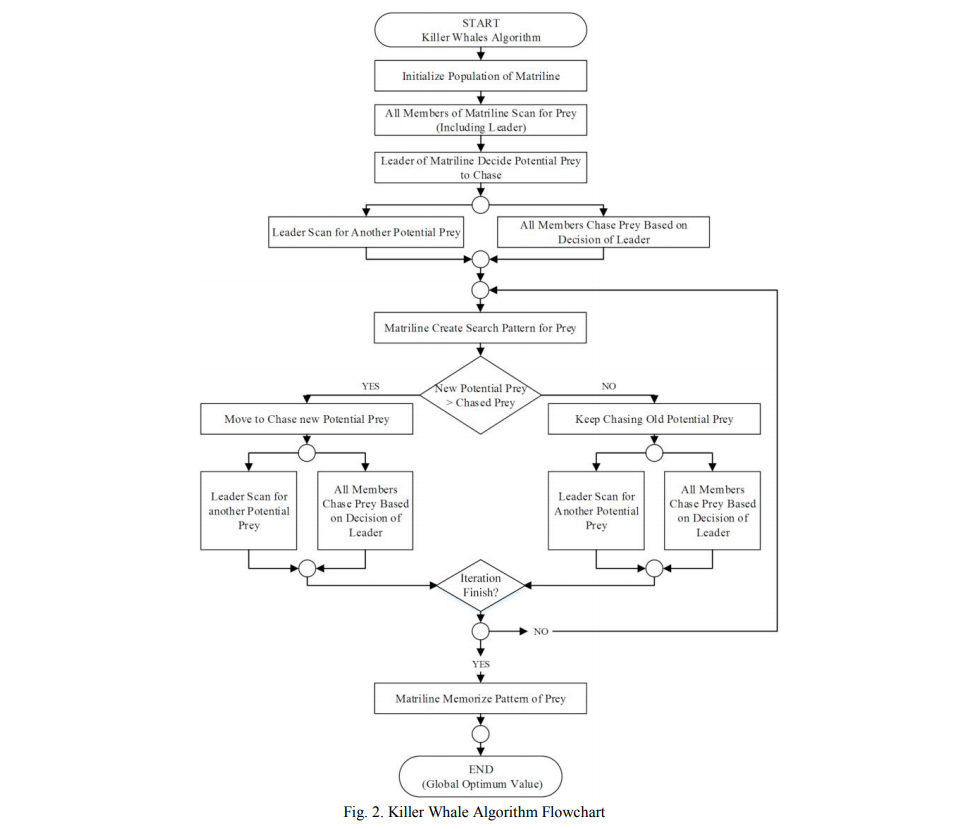


**Whale Optimization Algorithm - Beale Function**



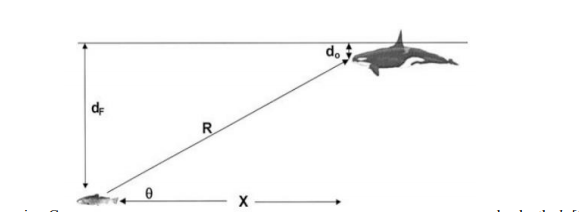
**Whale Optimization Algorithm - Goldstein Function**



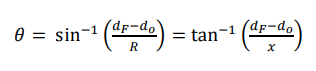


Whale Algorithm yani balina algoritması söyle çalışır; lider bir av belirler ve kendisi de dahil olmak üzere tüm sürü peşinden giderler, ardından lider yeni potansiyel avları inceler. Liderin belirlediği potansiyel av şuan sürünün peşinde olduğu avdan daha iyi ise şuan ki av bırakılarak potansiyel avın peşine düşülür.

Lider yine avın peşinden gitmek yerine yeni potansiyel bir av aramaya başlar ve bu durum istenilen iterasyon sayısına ulaşılana kadar devam eder.



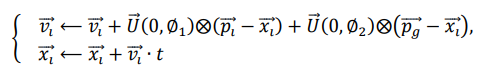
Yukarıda gördüğümüz görselde d0 ve dF balinanın ve avın derinliklerini gösteriyor, R aralarındaki doğrusal uzunluğu, X yataydaki uzaklıklarını θ ise yatay uzunluk ile doğrusal uzunluğun arasındaki açıyı gösteriyor.



Omega açısı bu şekilde hesaplanır.

**Hareketin ivmesi**

Her arama ajanı şuan ki konumundan avın konumuna doğru gitmek için ivmeye ihtiyaç duyar. Bu yüzden balinanın ava ulaşması için gereken hareket şu matematiksel formül ile ifade edilmiştir:



Her arama ajanının 3 tane yönlü vektörü vardır; 𝑥 ⃗⃗⃗şuan ki konumu ifade ediyor, 𝑝 ⃗önceki en iyi pozisyonu gösterirken 𝑣 ⃗ ise ivmeyi ifade ediyor, ayrıca yönlü vektör olmayan ve zamanı ifade eden

t ‘ye sahiptir. X ikili koordinat seti olarak ifade edilir ve uzaydaki konumumuzu gösterir. Algoritmanın her iterasyonunda şuan ki konum problemin çözümü olarak değerlendirilir. Eğer şuan ki pozisyon önceki en iyi konumdan daha iyi ise, şuan ki konum eski en iyi konum olur ve algoritma bir sonraki adıma geçer. Sonraki pozisyonlar 𝑣 ⃗⃗⃗ ye göre hesaplanır. Arama uzayımız [x,-x] şeklinde girilen sınırlar ile belirlenir ve sonra ki konum bu sınırlara göre belirlenir.