**1.1 Giriş**

Bu rapor, referans kitap "Engineering Optimization"ta sayfa 298'de yer alan 5.8 numaralı örnekte anlatılan Altın Oran Arama algoritmasını kullanarak Bracketing Yöntemi'nin uygulanmasını değerlendirmeyi amaçlamaktadır. Proje, bir fonksiyonun belirli bir aralıkta minimumunu bulmaya odaklanır ve Altın Oran kullanılarak yapılan iterasyonlarla arama aralığı daraltılır. Bu yöntem, arama bölgesini sistematik olarak daraltarak istenen tolerans seviyesine ulaşılmasını sağlar ve minimum değerin yaklaşık çözümünü sunar. Aynı zamanda, bu yöntem hem hesaplama maliyetini azaltarak hem de daha hızlı bir yakınsama sağlayarak optimizasyon problemlerinde kullanışllı bir seçenek sunar.

Altın Oran Arama yöntemi, özellikle tek tepe noktalarına sahip fonksiyonlarda kullanılmak üzere tasarlanmış bir tekniktir ve bu yöntemle elde edilen sonuçlar, hem teorik olarak hem de pratik uygulamalarda tatmin edici bir doğruluk sunar. Bu raporda, bahsi geçen yöntemin matematiksel altyapısı detaylandırılacak ve kodlama aşamasında izlenen adımlar ayrıntılarıyla analiz edilecektir. Ayrıca, algoritmanın güçlü ve zayıf yönleri de ortaya konularak gelecekteki olası iyileştirmelere ve çeşitli uygulama alanlarına da değinilecektir.

Bu raporun amacı, kodun ders kitabındaki metodolojiye sıkı sıkıya bağlı olup olmadığını değerlendirmek, varsa eksiklikleri tespit etmek ve çözümün nasıl elde edildiğini modern teknikler ve kodlama uygulamalarıyla analiz etmektir. Ayrıca, internet kaynaklarından elde edilen ek bilgiler ve güncel optimizasyon teknikleriyle de bu uygulamanın etkinliğini değerlendirmek de raporun önemli bir parçasıdır. Bu çalışmanın sonunda, yöntemin uygulanabilirliği ve iyileştirme potansiyeli hakkında kapsamlı bir görüş oluşturulacaktır.

**1.2 Kullanılan Kütüphaneler ve Araçlar**

Bu projede Python programlama dili kullanılarak Altın Oran Arama yöntemi kodlanmıştır. Python'un sunduğu zengin kütüphane desteği ve esnek yapısı, bu türden optimizasyon problemlerini etkin bir şekilde çözmeyi mümkün kılmaktadır. Kullanılan temel kütüphaneler şunlardır:

* **Numpy**: Sayısal hesaplamalar ve matematiksel işlemler için kullanılan güçlü bir kütüphane. Bu projede Altın Oran'ın hesaplanmasında ve fonksiyon değerlerinin değerlendirilmesinde kullanılmıştır. Numpy, vektör ve matris bazlı işlemleri kolay ve verimli bir şekilde gerçekleştirmek için ideal bir kütüphanedir ve bu proje boyunca fonksiyon değerlendirmelerinde yaygın olarak kullanılmıştır.
* **Python İteratif Yapılar**: Algoritmanın adım adım ilerleyebilmesi için döngü yapıları kullanılmıştır. Bu sayede, belirli bir hassasiyete ulaşılana kadar iterasyonlar tekrarlanmıştır. Bu döngü yapıları, algoritmanın mantıksal akışını sağlar ve her bir adımda elde edilen sonuçların analizi için çok önemlidir.
* **Matematiksel Sabitler ve Fonksiyonlar**: Altın Oran gibi sabitler, optimizasyon sürecinde arama alanını daraltmak ve en küçük değeri bulmak için kullanılmıştır. Bu matematiksel sabitler, hesaplama sürecinde denge ve hassasiyet sağlar. Altın Oran, arama sürecini dengelemek için çok faydalıdır ve algoritmanın hızlı bir şekilde en uygun sonuca ulaşmasını sağlar.

Bu araçlar, optimizasyon sürecinin daha etkin ve doğru bir şekilde yürütülmesini sağlamaktadır. Kodun genel yapısı, bu araçları kullanarak matematiksel analizleri etkin bir şekilde gerçekleştirmek üzere tasarlanmıştır. Kullanılan her bir kütüphane ve fonksiyon, kodun esnekliğini ve güçlü çalışma kapasitesini arttırmaya yöneliktir. Özellikle, İteratif yapılar ve numpy kütüphanesi bu türden sayısal hesaplamalarda verimliliği önemli ölçüde arttırmaktadır.

Ayrıca, Python'un esnek veri yapıları ve kolay kullanılabilir fonksiyonellikleri, bu tür karmaşık optimizasyon problemlerinde kullanıcı dostu bir ortam sunar. Bu proje, Python'un bu avantajlarından faydalanarak, hem teorik bilgileri pratiğe dökme hem de kodlama sürecinde daha hızlı ve anlaşılır bir çözüm sağlama konusunda etkin olmuştur. İşlem süresince Python'un hata ayıklama araçları ve interaktif yapısı, optimizasyon algoritmasının farklı aşamalarını takip etmek ve geliştirme yapmak için çok faydalı olmuştur.

Sonuç olarak, kullanılan kütüphaneler ve iteratif yapılar, Altın Oran Arama yönteminin başarılı bir şekilde uygulanmasını sağlamış ve elde edilen sonuçların doğruluğunu arttırmıştır. Bu proje, teorik optimizasyon bilgilerini, Python'un sunduğu zengin kütüphaneler ve esnek kodlama araçları ile gerçeğe dökerek, başarılı bir uygulama ortaya koymuştur.

**Bölüm 2: Yöntemler ve Matematiksel Teorik Yapılar**

**2.1 Bracketing Yöntemi Genel Bakış**

Bracketing Yöntemi, bir fonksiyonun verilen bir aralık içindeki minimumunu bulmak için kullanılan temel bir optimizasyon tekniğidir. Bu yöntem, arama alanını sistematik olarak daraltarak en uygun çözümü bulmayı amaçlar. Bracketing yöntemi, her iterasyonda aralığı daraltarak daha küçük bir aralık içinde minimuma yaklaşır ve sonunda belirlenen tolerans değerine ulaşılır. Bu yöntem, adım adım ilerleyerek belirli bir hassasiyete ulaşınca kadar devam eder ve her adımda arama alanı daha dar bir bölgeye indirgenir.

Bu yöntemde, her iterasyon adımında aralıkta iki nokta belirlenir ve bu noktalardaki fonksiyon değerleri karşılaştırılarak aralık daraltılır. Bu şekilde, her seferinde minimumun yer aldığına inanılan alt aralık seçilerek bu alt aralık içinde arama devam eder. Bracketing yöntemi, özellikle çok fazla tepe veya çukur noktası bulunmayan ve yalnızca bir minimum veya maksimum noktasına sahip olan fonksiyonlar için uygun bir yöntemdir. Bu yöntemin etkinliği, aralığı daraltma sürecinin dengeli bir şekilde yürütülmesine bağlıdır. Bu dengenin sağlanması, minimum değerin bulunduğu noktaya en hızlı ve en verimli şekilde ulaşılmasını garanti eder.

Bracketing yönteminin önemli avantajlarından biri, fonksiyonun türev bilgisine gerek duyulmamasıdır. Bu yöntem yalnızca fonksiyon değerlerinin karşılaştırılmasına dayanarak ilerler, bu da türevi zor veya olanaksız olan fonksiyonlar için bu yöntemi çok kullanışlı hale getirir. Dolayısıyla, bu yöntem türev alınmayan veya karmaşık yapıya sahip fonksiyonlarda etkili bir optimizasyon aracı olarak kullanılabilir. Bunun yanı sıra, algoritma uygulama kolaylığı nedeniyle de yaygın olarak tercih edilmektedir.

Bracketing yöntemi ayrıca, aralık içindeki maksimumu veya minimumu bulmak için belirli bir büyüklükteki adımları kullanarak sürecini yürütür. Bu adımlar sayesinde algoritma her bir aşamada arama aralığını daha verimli şekilde daraltır ve sınırlar arasındaki mesafeyi küçülterek nihai sonuca ulaşır. Her iterasyonda yapılan bu daraltma işlemi, fonksiyonun minimum noktasının bulunduğu aralığı giderek daraltır ve bu da algoritmanın hassasiyetini arttırır. Özellikle, belirsizliğin azaltılması ve arama alanının etkin bir şekilde daraltılması algoritmanın hızlı ve doğru sonuçlara ulaşmasını sağlar.

**2.2 Altın Oran Arama Yöntemi**

Altın Oran Arama Yöntemi, Bracketing Yöntemi'nin özel bir türü olup, değerlendirme noktalarını seçmek için Altın Oran'ı kullanır. Altın Oran, antik çağlardan beri estetik ve matematikte kullanılan bir orantıdır ve optimizasyon problemlerinde de dengenin sağlanması amacıyla kullanılabilir. Altın Oran kullanılarak yapılan arama, aralığın dengeli bir şekilde daraltılmasını sağlar ve bu sayede arama sürecinde gereksiz değerlendirmelerin önüne geçilir. Altın Oran'ın bu özelliği, optimizasyon sürecinin etkinliğini önemli ölçüde arttırır ve algoritmanın daha hızlı sonuç vermesini sağlar.

Altın Oran, şu şekilde tanımlanır:

Bu oran, sadece matematiksel estetikle değil, aynı zamanda optimizasyon sürecinin en verimli şekilde yürütülmesiyle de ilgilidir. Altın Oran kullanılarak arama aralığı öyle bölünür ki, yeni alt aralıklardan biri diğerine oranla Altın Oran oranında olur. Bu durum, arama sürecini dengeli ve etkili bir şekilde yürütmeyi sağlar. Altın Oran Arama Yöntemi'nde, iki değerlendirme noktası belirlenir ve bu noktalardaki fonksiyon değerleri karşılaştırılır. Daha küçük fonksiyon değerine sahip olan taraf, minimuma daha yakın olduğuna inanılan taraf olarak kabul edilir ve aralık bu yönde daraltılır.

Altın Oran Arama Yöntemi'nin en önemli avantajı, her iterasyonda aralığı dengeli bir şekilde daraltarak en küçük değeri bulmaya çalışmasıdır. Bu sayede, algoritmanın daha az iterasyonla ve daha hızlı bir şekilde hedeflenen sonuca ulaşması sağlanır. Altın Oran, arama sürecini her zaman dengenin sağlanacağı şekilde yönlendirir ve bu da algoritmanın verimliliğini arttırır. Bu teknik, aralık daraltma sürecinde dengenin bozulmasını önler ve fonksiyonun minimum noktasına daha doğrudan ve verimli bir şekilde ulaşılmasını sağlar. Bu da hesaplama sürecini hızlandırır ve daha hassas sonuçlar elde edilmesine olanak tanır.

**2.3 Matematiksel Süreç ve Teorik Analiz**

Altın Oran Arama yöntemi, iteratif bir süreçtir ve her adımda aralığı iki yeni değerlendirme noktası belirleyerek daraltır. Bu yöntem, iki değerlendirme noktası (x1 ve x2) arasında yapılan kıyaslamalarla arama aralığını azaltmayı amaçlar. Bu noktalar, Altın Oran kullanılarak aralıkta belirlenir ve şu işlemler yapılır:

1. **Aralığı Belirleme**: Başlangıçta a ve b olarak iki sınır belirlenir. Bu iki sınır, aramanın başlayacağı ve minimumun bulunması beklenen aralığı temsil eder. Bu aralıklar, arama süresince daraltılacak ve algoritmanın sonuca ulaşması için kullanılacak temel referans noktalarıdır.
2. **Değerlendirme Noktalarını Belirleme**: x1 ve x2 noktaları, aralık içinde Altın Oran oranı kullanılarak belirlenir. x1, a'ya daha yakın bir nokta, x2 ise b'ye daha yakın bir nokta olarak seçilir. Bu noktalar, Altın Oran oranı kullanılarak dengeli bir şekilde belirlenir ve arama sürecinin etkili bir şekilde yürütülmesine olanak tanır.
3. **Fonksiyon Değerlerini Hesaplama**: f(x1) ve f(x2) değerleri hesaplanır ve bu değerler karşılaştırılır. Bu karşılaştırma sonucunda daha küçük fonksiyon değerine sahip olan alt aralık, minimumun bulunduğu aralık olarak kabul edilir ve bu alt aralık seçilerek arama devam eder.
4. **Aralığı Daraltma**: Elde edilen f(x1) ve f(x2) sonuçlarına göre aralık daraltılır. Eğer f(x1) > f(x2) ise, a sınırı x1 noktasına taşınır; aksi halde b sınırı x2 noktasına taşınır. Bu işlem, aralığın daraltılarak minimum noktanın daha dar bir aralıkta aranmasını sağlar. Böylece her iterasyonda arama alanı küçülür ve minimum noktaya daha fazla yaklaşılır.
5. **Yakınsama ve Durdurma Kriteri**: Aralık genişliği belirli bir tolerans seviyesinden küçük olana kadar veya maksimum iterasyon sayısına ulaşılınca bu süreç devam eder. Tolerans değeri, algoritmanın hassasiyetini belirler ve aramanın ne zaman sona ereceğini kontrol eder. Maksimum iterasyon sayısı ise, algoritmanın çok uzun süre çalışmasını önlemek ve belirli bir süre sonunda durmasını sağlamak için kullanılır.

Altın Oran Arama Yöntemi, bu iteratif süreç sayesinde en küçük fonksiyon değerini bulmak için etkin bir yöntem sunar. Yöntemin matematiksel temeli, arama aralığını dengeleyerek her adımda fonksiyonun minimum değerini daha dar bir aralıkta aramaya dayanır. Bu da yöntemin hem hesaplama maliyetini düşürmesine hem de daha hassas sonuçlar elde edilmesine olanak tanır. Teorik olarak, Altın Oran Arama Yöntemi, optimizasyon problemlerinde arama sürecini dengenin sağlanacağı şekilde yönlendirir ve gereksiz hesaplamaların önüne geçer.

Altın Oran'ın bu dengeli yapısı, algoritmanın verimliliğini artıran temel faktörlerden biridir. Bu sayede, iterasyon sayısı azalır ve her bir adımda hedefe daha hızlı ulaşılır. Ayrıca, Altın Oran Arama Yöntemi'nde aralığın her zaman belirli bir oranda bölünmesi, minimum noktaya ulaşmak için gereken adımların sayısını azaltır. Bu, özellikle çok büyük veya çok karmaşık aralıklar söz konusu olduğunda büyük bir avantaj sağlar ve arama sürecini daha verimli hale getirir. Aynı zamanda, aralığın her iki taraftan dengeli bir şekilde daraltılması, lokal minimum noktalarından kaçınarak gerçek minimuma ulaşmayı kolaylaştırır.

Bu teorik süreçlerin sonunda, Altın Oran Arama Yöntemi ile elde edilen sonuçlar, fonksiyonun minimum değerine oldukça yakın bir tahmin sunar. Bu yöntem, sadece optimizasyon problemlerinde değil, aynı zamanda mühendislik ve ekonomide de geniş bir uygulama alanına sahiptir. Çünkü bu tür problemlerde genellikle bir sistemin en iyi performansını veya en düşük maliyetini bulmak için arama yapılır ve Altın Oran Arama Yöntemi bu tür problemlerin çözümünde oldukça etkili bir yöntemdir.

**3.1 Kodlama İncelemesi**

Bu bölümde, Altın Oran Arama Yöntemi kullanılarak yazılan Python kodunun yapısı ve fonksiyonelliği detaylıca incelenecektir. Kodlama süreci, yöntemin teorik temelini uygulamaya dökme aşamasıdır ve bu nedenle her adımda dikkatli bir analiz gerekmektedir. Bu inceleme kapsamında, kullanılan Python kütüphaneleri, algoritmanın temel yapısı ve optimizasyon hedeflerinin başarıya ulaşıp ulaşmadığı değerlendirilecektir. Kodlama sürecinde karşılaşılan zorluklar, bu zorlukların nasıl aşıldığı ve bu süreçlerin teorik temele uygunluğu da ele alınacaktır.

Kodda öne çıkan yapılar, iterasyon bazlı arama algoritmasının doğru ve etkili bir şekilde uygulanmasını sağlamak için kullanılan temel programlama yapılarıdır. Altın Oran değerinin belirlenmesi, bu oranın fonksiyon üzerinde nasıl kullanıldığı ve sonuçlara göre aralığın daraltılması süreci, algoritmanın en kritik bileşenleridir. Bu adımlar, Python'un matematik kütüphaneleri (özellikle math modülü) ve iteratif yapılarını kullanarak başarıyla gerçekleştirilmiştir. Bu süreç, hem Python'un esnekliğini hem de matematiksel doğruluğunu kullanarak optimizasyon amacına ulaşmayı mümkün kılar.

Kodun yapısında dikkat çeken bir diğer unsur, başlangıç ve bitiş noktaları arasında belirli bir hassasiyete ulaşana kadar iterasyonların tekrar edilmesidir. Bu iterasyon sürecinde, x1 ve x2 noktaları Altın Oran'a göre belirlenir ve bu noktalar arasındaki fonksiyon değerleri karşılaştırılır. Daha sonra aralığın daraltılması sağlanarak, aramanın daha dar bir bölgede yoğunlaşması sağlanır. Kodun etkinliği, iterasyon sayısını minimize ederken doğruluğu maksimize etme yeteneğine dayanır. Her iterasyonda fonksiyonun minimum değerine daha fazla yaklaşılması, algoritmanın doğruluğunu arttırmakta ve en uygun çözümü bulma sürecini hızlandırmaktadır.

Kodun başarılı bir şekilde çalıştığını gösteren bir diğer önemli nokta ise, belirli bir tolerans seviyesine ulaşıldığında algoritmanın durması ve elde edilen sonucun rapor edilmesidir. Tolerans seviyesi, sonucun hassasiyetini belirleyen önemli bir parametredir ve bu değer ne kadar küçük olursa elde edilen sonuç o kadar doğru olur. Kod, belirlenen bu tolerans değerine ulaşıldığında yakınsadığını ve sonucun bulunduğunu bildirir. Bu yaklaşım, kodun hem doğruluğunu hem de verimliliğini garanti altına almak için kullanılmıştır. Bu süreç, kullanıcıya belirli bir hata payı içinde en doğru sonucu sunmayı amaçlar.

Ayrıca, algoritmanın performansını ve doğruluğunu artırmak için, iterasyon sürecinde gereksiz adımlardan kaçınılmış ve her iterasyonun minimum maliyetle gerçekleşmesi sağlanmıştır. Bu durum, özellikle büyük ölçekli optimizasyon problemlerinde hesaplama maliyetini önemli ölçüde azaltır. Kodun verimliliği, iterasyon sayısını düşük tutarak ve gereksiz hesaplamalardan kaçınarak sağlanmıştır. Bu sayede, algoritma hem hızlı hem de güvenilir sonuçlar elde etmiştir.

**3.2 Sonuçların Analizi ve Yorumlanması**

Altın Oran Arama Yöntemi kullanılarak gerçekleştirilen bu optimizasyon çalışmasında elde edilen sonuçlar, teorik olarak beklenen sonuçlarla büyük ölçüde uyumludur. Kodun çıktılarında, her iterasyon adımında aralığın daraldığı ve minimuma yaklaşıldığı açık bir şekilde görülmektedir. Her iterasyonun sonunda hesaplanan f(x) değerleri, minimum değeri bulmak için aralığın doğru bir şekilde daraltıldığını ve aramanın doğru yolda ilerlediğini göstermektedir. Bu durum, aramanın en uygun çözümü bulma sürecinde başarılı bir şekilde ilerlediğinin göstergesidir.

Sonuçlar, belirlenen tolerans değerine ulaşıldığında algoritmanın durduğunu ve elde edilen minimum noktayı rapor ettiğini göstermektedir. Bu, algoritmanın etkin çalıştığının ve verilen problemin çözümü için uygun bir sonuca ulaşıldığının bir kanıtıdır. Özellikle, Altın Oran Arama Yöntemi'nin dengeli arama yapısı, minimum değerin en az sayıda iterasyonla bulunmasına olanak tanımıştır. Bu da hesaplama süresini azaltarak daha verimli bir optimizasyon sağlamıştır. Ayrıca, aralığın dengeli bir şekilde daraltılması sayesinde, minimum noktanın doğru bir şekilde belirlenmesi sağlanmış ve sonuçların hassasiyeti arttırılmıştır.

Algoritmanın doğruluğu ve verimliliği, iterasyonlar sırasında elde edilen değerlerin Altın Oran prensibine uygun bir şekilde ilerlediğini gösteren çıktılarla desteklenmektedir. Altın Oran'ın kullanılması, her iki değerlendirme noktası arasındaki oranın sabit tutulmasını sağlar ve bu da algoritmanın stabilitesini ve hassasiyetini artırır. Bu sayede, her iterasyonla birlikte arama alanı sistematik olarak daraltılarak daha dar bir aralıkta en uygun sonuca ulaşılması sağlanır. Kodun sonunda rapor edilen minimum değer, fonksiyonun gerçek minimumu ile uyum içerisindedir ve bu da yöntemin başarılı bir şekilde uygulandığını göstermektedir.

Sonuçların analizi, aynı zamanda algoritmanın çeşitli senaryolarda nasıl performans gösterdiğini de gözler önüne sermektedir. Algoritmanın performansı, başlangıç aralıklarının genişliği ve belirlenen tolerans seviyesine göre değişiklik gösterebilir. Daha geniş aralıklar veya daha küçük tolerans seviyeleri, daha fazla iterasyon gerektirebilir ancak bu durum sonuçların doğruluğunu artırır. Kodun bu tür değişkenlere karşı gösterdiği esnek yapı, algoritmanın farklı optimizasyon problemlerine uyarlanabilirliğini artırmaktadır.

**3.3 Sonuç ve Değerlendirme**

Bu bölümde, Altın Oran Arama Yöntemi ile gerçekleştirilen optimizasyon süreci incelenmiş ve kodun etkinliği değerlendirilmiştir. Yapılan incelemeler sonucunda, bu yöntemin verilen bir fonksiyonun minimumunu bulma konusunda başarılı ve etkili olduğu görülmüştür. Altın Oran prensibine dayalı olarak yapılan bu optimizasyon, arama sürecinin dengeli ve sistematik bir şekilde yürütülmesini sağlamış ve bu sayede aramanın daha az iterasyonla sonuçlanmasına olanak tanımıştır.

Altın Oran Arama Yöntemi'nin avantajlarından biri, her iterasyonda arama aralığını dengeli bir şekilde daraltarak minimum noktaya yaklaşmasıdır. Bu durum, hesaplama süresini azaltmakta ve daha hızlı bir şekilde sonuca ulaşılmasını sağlamaktadır. Ayrıca, algoritmanın türev bilgisine ihtiyaç duymaması, bu yöntemi özellikle türev alınması zor veya mümkün olmayan fonksiyonlar için çok kullanışlı hale getirmektedir. Bu özellik, yöntemin geniş bir uygulama alanında kullanılmasına olanak tanır.

Yapılan kod incelemesi ve sonuç analizine göre, kullanılan Python kodu hem teorik temellere uygun hem de pratikte başarılı sonuçlar elde edecek şekilde yazılmıştır. Her iterasyon adımında aralığın dengeli bir şekilde daraltılması, Altın Oran'ın sağladığı avantajlarla birlikte algoritmanın hassasiyetini ve doğruluğunu artırmıştır. Sonuç olarak, bu yöntemle gerçekleştirilen optimizasyonun hem doğruluk hem de verimlilik açısından tatmin edici olduğu ve belirlenen problemler için uygun bir çözüm sunduğu belirlenmiştir.

Özellikle büyük ölçekli ve karmaşık optimizasyon problemlerinde Altın Oran Arama Yöntemi, etkin bir çözüm yöntemi olarak öne çıkmaktadır. Bu yöntemin uygulanması, optimizasyon problemlerinin çözümünde sadece matematiksel doğruluk değil, aynı zamanda hesaplama maliyetlerinin azaltılması ve daha hızlı çözümler elde edilmesi açısından da önemli avantajlar sunmaktadır. Bu sebeple, bu çalışmada incelenen yöntem ve kullanılan kodun, optimizasyon süreçlerinde pratik ve etkili bir araç olduğu sonucuna varılmıştır.