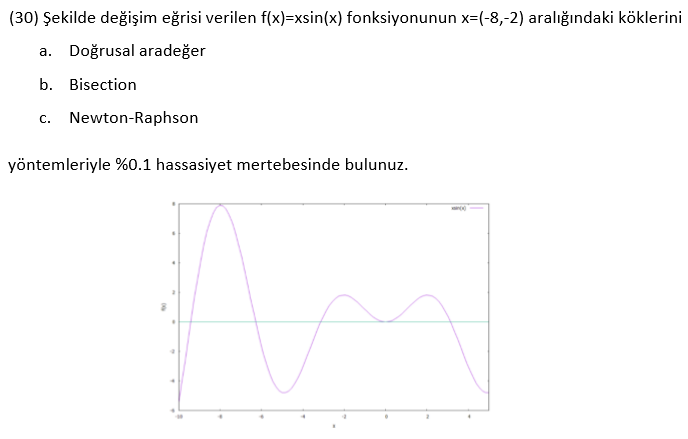
**Kaya Han Taş**

**20183405003**

**UBT**

***NÜMERİK YÖNTEMLER ARASINAV ÖDEVİ***

***SORU 1:***



***ÇÖZÜM:***

***a) Doğrusal Aradeğer***

İlk önce verilen f(x) fonksiyonunun bu aralıkta sürekli olduğunu belirliyoruz. Fonksiyonun bu aralıkta sürekli olması nedeniyle bu metodu uygulayabiliriz.

Bu yöntemde kullanacağımız formül: olup iterasyon yapılırken durma koşulu olarak kullanılacak denklemimiz ise ε = hassasiyet değeri, xn = n. iterasyonda bulunan kök adayı, xn-1 = n. iterasyondan bir önceki iterasyonda bulunan kök adayı olmak üzere: olacaktır. Bu şart sağlandığında iterasyonumuz duracak ve kökümüzü belli bir hassasiyet ile elde etmiş olacağız. Sorumuzda verilen hassasiyet değerine göre epsilon değerimiz ’dir. Buna göre sorumuzun çözümüne başlıyoruz.

Şimdi sırasıyla aralıkları inceliyoruz.

1-) *(-3,-2)* Aralığı:

ve olarak bulunur. koşulu sağlanmadığından dolayı bu aralıkta kök yoktur.

2-) *(-4,-3)* Aralığı:

veolarak bulunur. koşulu sağlandığından dolayı bu aralıkta kök vardır. Buna göre aralığımız olmak üzere iterasyona başlıyoruz. (Yani ve ’tür.)

*1. İterasyon:*

Bu bulunan değeri fonksiyonda yerine yazarak aralığı üstten mi yoksa alttan mı daraltacağımıza karar veriyoruz. Bunun için fonksiyonda x1 yerine yazıldığında çıkan sonucun işaretine bakmamız gerekmektedir.

İşaret pozitif çıktı. Buna göre koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Şimdi durma koşulu için kontrol yapıyoruz. Xn-1=-3 ve xn=-3.1226927 olmak üzere:

Bu değer 0.001’den küçük olmadığından dolayı iterasyonumuza devam ediyoruz.

Sonraki iterasyonlarda işlemler ayrıntılı olarak anlatılmayacaktır. Uygulanacak yöntem her iterasyonumuz için aynı biçimdedir.

*2. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*4. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x4 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun birinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 1. Kökü***: (%0.1 Hassasiyet ile)

3-) *(-5,-4)* Aralığı:

veolarak bulunur. koşulu sağlanmadığından dolayı bu aralıkta kök yoktur.

4-) *(-6,-5)* Aralığı:

veolarak bulunur. koşulu sağlanmadığından dolayı bu aralıkta kök yoktur.

5-) *(-7,-6)* Aralığı:

veolarak bulunur. koşulu sağlandığından dolayı bu aralıkta kök vardır. Buna göre aralığımız olmak üzere iterasyona başlıyoruz. (Yani ve ’dır.)

*1. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*2. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x3 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun ikinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 2. Kökü***: (%0.1 Hassasiyet ile)

6-) *(-8,-7)* Aralığı:

veolarak bulunur. koşulu sağlanmadığından dolayı bu aralıkta kök yoktur.

**Sonuç:** fonksiyonunun aralığında doğrusal aradeğer metoduyla yapılan, %0.1lik hassasiyet ile yaklaşım sonucu iki kök bulunmuştur. Bu kökler: ve ’dir.

***b) Bisection***

İlk önce verilen f(x) fonksiyonunun bu aralıkta sürekli olduğunu belirliyoruz. Fonksiyonun bu aralıkta sürekli olması nedeniyle bu metodu uygulayabiliriz.

Bu yöntemde kullanacağımız formül: olup iterasyon yapılırken durma koşulu olarak kullanılacak denklemimiz ise ε = hassasiyet değeri, xn = n. iterasyonda bulunan kök adayı, xn-1 = n. iterasyondan bir önceki iterasyonda bulunan kök adayı olmak üzere: olacaktır. Bu şart sağlandığında iterasyonumuz duracak ve kökümüzü belli bir hassasiyet ile elde etmiş olacağız. Sorumuzda verilen hassasiyet değerine göre epsilon değerimiz ’dir. Buna göre sorumuzun çözümüne başlıyoruz.

Bisection metodu ile Doğrusal Aradeğer metodunda kullanılan koşul aynıdır. ise o aralıkta bir kökün varlığından bahsedilebilir. İki metodun koşulu aynı olduğundan dolayı aralıkları teker teker incelemek yerine kısalık açısından, doğrusal aradeğer metodu çözümünde zaten yaptığımız testler ile bir kökün varlığını bulduğumuz aralıkları inceliyoruz. Bu aralıklar ve aralıklarıdır. Bu iki aralıkta bisection metodu ile %0.1 hassasiyet ile köke yaklaşacağız.

1-) *(-4,-3)* Aralığı:

veolarak bulunur. koşulu sağlandığından dolayı bu aralıkta kök vardır. Buna göre aralığımız olmak üzere iterasyona başlıyoruz. (Yani ve ’tür.)

*1. İterasyon:*

Şimdi bu bulduğumuz “kök adayını” fonksiyonda yerine yazarak aralığı üstten mi yoksa alttan mı daraltacağımıza karar veriyoruz. Bunun için fonksiyonda x1 yerine yazıldığında çıkan sonucun işaretine bakmamız gerekmektedir. Kısacası “Doğrusal Aradeğer” çözümünde kullandığımız aynı yöntemi uyguluyoruz. Amaç aralığı daraltmak.

İşaret negatif çıktı. Dikkat etmemiz gereken kısım aralığı üst mü yoksa alt sınırdan mı sınırlayacağımızdır. Bu durumu daha ayrıntılı olarak konuşursak yani negatif bir değer olduğundan ve bu değeri başka bir negatif değer ile değiştirirsek yine koşulu sağlanacağından dolayı üst sınır yani -4’ten sınırlamak doğrudur. Bunun için -4 değerini -3.5 değeri ile değiştirilir. Böylece hem koşulu sağlanmaya devam eder hem de aralığımızı sınırlandırmış oluruz.

Buna göre koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Şimdi durma koşulu için kontrol yapıyoruz. Xn-1=-3 ve xn=-3.5 olmak üzere:

Bu değer 0.001’den küçük olmadığından dolayı iterasyonumuza devam ediyoruz.

Sonraki iterasyonlarda işlemler ayrıntılı olarak anlatılmayacaktır. Uygulanacak yöntem her iterasyonumuz için aynı biçimdedir.

*2. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*4. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*5. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*6. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*7. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*8. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*9. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*10. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*11. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x11 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun birinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 1. Kökü***: (%0.1 Hassasiyet ile)

2-) *(-7,-6)* Aralığı:

veolarak bulunur. koşulu sağlandığından dolayı bu aralıkta kök vardır. Buna göre aralığımız olmak üzere iterasyona başlıyoruz. (Yani ve ’dır.)

*1. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*2. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*4. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*5. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*6. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*7. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*8. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*9. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret pozitif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*10. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*11. İterasyon:*

İşaret kontrolü için fonksiyonda yerine yazıyoruz.

İşaret negatif çıktı. koşulu sağlandığından yeni aralığımız olup bu aralıkta bir kök bulunmaktadır. Durma koşulu kontrolü:

Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x11 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun ikinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 2. Kökü***: (%0.1 Hassasiyet ile)

**Sonuç:** fonksiyonunun aralığında Bisection metoduyla yapılan, %0.1lik hassasiyet ile yaklaşım sonucu iki kök bulunmuştur. Bu kökler: ve ’dir.

Ayrıca buradan çıkarılabilecek bir diğer sonuç Bisection metodunun, Doğrusal Aradeğer metoduna göre köke çok daha yavaş bir biçimde yaklaşmasıdır.

***c) Newton-Raphson***

Bu yöntem kapalı aralık bir yöntem olmayıp diğer iki yöntemden farklı olarak açık aralık bir yöntemdir. Yani köke yaklaşılırken aralık dışına çıkma ihtimali bulunmaktadır. Bu yüzden kökün yaklaşık olarak hangi aralıkta olduğunu bilmek bu yöntemi kullanmada önemli bir husustur. Burada yine kısa olması açısından önceki iki yöntemde bulduğumuz aralıkları kullanıyoruz: ve

Bu iki aralıkta kökün bulunduğunu ayrıca grafiğe baktığımızda da görebiliriz. Diğer işlemler olmadan yine kökün yaklaşık olarak hangi aralıkta olduğu bu sayede bulunabilir.

Bu yöntemde kullanacağımız formül: olup iterasyon yapılırken durma koşulu olarak kullanılacak denklemimiz ise ε = hassasiyet değeri, xn = n. iterasyonda bulunan kök adayı, xn-1 = n. iterasyondan bir önceki iterasyonda bulunan kök adayı olmak üzere: olacaktır. Bu şart sağlandığında iterasyonumuz duracak ve kökümüzü belli bir hassasiyet ile elde etmiş olacağız. Sorumuzda verilen hassasiyet değerine göre epsilon değerimiz ’dir.

Çözüme başlamadan önce fonksiyonun türevini bulmak işimizi kolaylaştıracaktır. fonksiyonunun türevini alırsak bulacağımız denklem: olacaktır.

Ayrıca aralıklarda ilk değer olarak x0’ı aralığın alt sınırı olarak almak genellikle tavsiye edilir. Bunlar göz önünde bulundurularak çözüme başlayabiliriz.

1-) *(-4,-3)* Aralığı:

Bu yöntemde Bisection ve Doğrusal Aradeğer metodlarında olduğu gibi bir koşul bulunmamaktadır. Bu aralıkta alt sınırı x0 alıyoruz. Yani: olarak alınır.

*1. İterasyon:*

Bu yöntemdeki iterasyonlarda öncelikle ve değerlerini hesaplayıp sonrasında yerine yazarak kökü bulmak uygundur. Şimdi bunu dikkate alarak ilk iterasyonumuzu yapıyoruz.

ve

Bu değerler bulunduktan sonra formülümüzde bu değerleri yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*2. İterasyon:*

ve değerlerini hesaplıyoruz.

ve

Bu değerleri formülde yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

ve değerlerini hesaplıyoruz.

ve

Bu değerleri formülde yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x3 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun birinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 1. Kökü***: (%0.1 Hassasiyet ile)

1-) *(-7,-6)* Aralığı:

Bu yöntemde Bisection ve Doğrusal Aradeğer metodlarında olduğu gibi bir koşul bulunmamaktadır. Bu aralıkta alt sınırı x0 alıyoruz. Yani: olarak alınır.

*1. İterasyon:*

ve değerlerini hesaplıyoruz.

ve

Bu değerleri formülde yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*2. İterasyon:*

ve değerlerini hesaplıyoruz.

ve

Bu değerleri formülde yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

Bu nedenle iterasyonumuza devam ediyoruz.

*3. İterasyon:*

ve değerlerini hesaplıyoruz.

ve

Bu değerleri formülde yerine yazarak yeni kökümüzü elde ediyoruz.

Durma koşulu kontrolü:

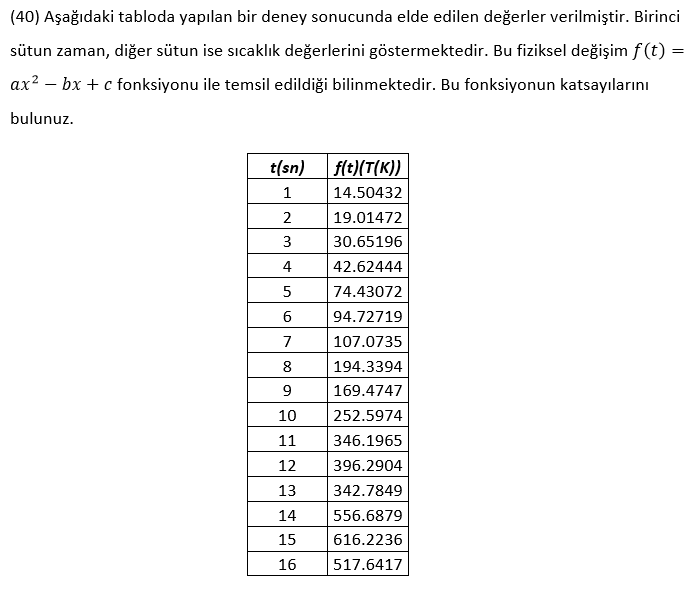
Durma koşulumuz sağlandı. Burada iterasyon yapmayı durduruyoruz. Bulduğumuz x3 değeri bizim kökümüze %0.1 (0.001) hassasiyet ile yaklaşmamız sonucu elde edilen bir değer olup f(x) fonksiyonun ikinci kökü aşağıdaki gibidir.

***f(x) fonksiyonun 2. Kökü***: (%0.1 Hassasiyet ile)

**Sonuç:** fonksiyonunun aralığında Newton-Raphson metoduyla yapılan, %0.1lik hassasiyet ile yaklaşım sonucu iki kök bulunmuştur. Bu kökler: ve ’dür.

Ayrıca buradan çıkarılabilecek bir diğer sonuç Newton-Raphson metodunun çözüm yaptığımız diğer iki metod olan Bisection ve Doğrusal Aradeğer metodlarından çok daha hızlı bir şekilde köke yaklaştığıdır.

***SORU 2:***

******

***ÇÖZÜM:***

a, b ve c değerleri birer katsayıdır. Bu kat sayıları aşağıda olan bir denklem sistemi ile bulmayı planlıyoruz.

Burada matrisi katsayılarımız a,b,c’yi temsil etmektedir. Kolaylık olması açısından c,b,a’ya sırasıyla c0, c1 ve c2 diyoruz.

Yani fonksiyonunu aşağıdaki şekilde yeniden yazabiliriz.

Şimdi ilk olarak matrisini bulmayla çözümümüze başlıyoruz. Bunun için formül olarak kullanılacak olup bu formül bize V vektörünün k. elemanını vermektedir.

Bize 3 tane katsayı gereklidir. Bu nedenle formülde ve değerleri kullanılacaktır. 3. Dereceden bir polinom uydurulması istenseydi c3 katsayısına ihtiyaç duyulacak ve bu nedenle k=3 değeri de kullanılacaktı. Fakat bu soruda 2. Dereceden bir fonksiyonun bu datayı/veriyi temsil ettiği söylendiği için sadece değerlerini bu çözümde kullanıyoruz.

Şimdi üç tane k değerimiz olması nedeniyle matrisinde 3 tane sonuç olması gerektiği ortadadır. Buna göre ve değerlerini formülünü kullanarak aşağıdaki şekilde bulabiliriz:

Sonuç olarak tekrar özetlersek:

Buna göre matrisi aşağıdaki gibi yazılabilir.

Şimdi matrisini formülü ile bulmayı hedefliyoruz. matrisini aşağıdaki gibi yazabiliriz.

x değerlerinin sıfırıncı derecelerini toplamayı, x değerlerinin birinci derecelerini toplamayı, x değerlerinin ikinci derecelerini toplamayı vb. söylemektedir. Bunları dikkate alarak matrisini toplama işlemlerini yaptıktan sonra aşağıdaki biçimde yazabiliriz.

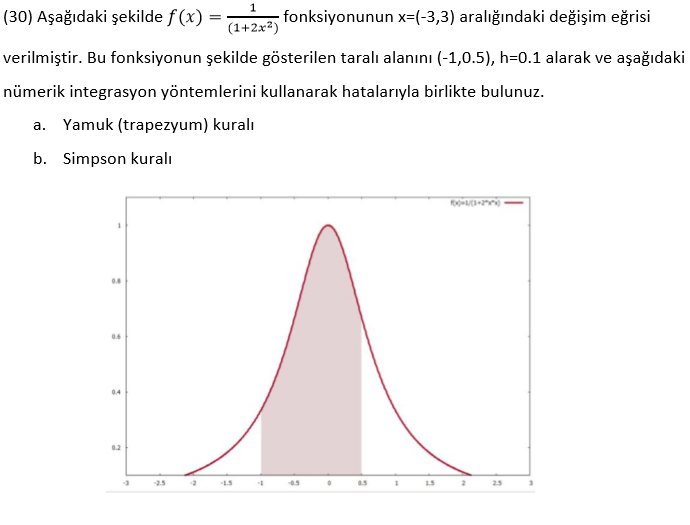
Son olarak işlemimiz aşağıdaki hali almıştır.

Bu denklem sistemini çözdüğümüzde ortaya çıkan katsayılar aşağıdaki gibidir. (*Son sayfada python kodu olarak çözümü ekstra olarak verilmiştir.(\*))*

ve sırasıyla c,b,a katsayılarını temsil ettiğinden dolayı bizden soruda istenen katsayılar aşağıdaki gibidir.

Denklemde yerine yazarsak elde edeceğimiz fonksiyonu ise aşağıdaki gibi yazılabilir.

***SORU 3:***



***ÇÖZÜM:***

***a) Yamuk (Trapezyum) Kuralı***

Herhangi bir taralı alanın altında kalan alanı bulmak için integral hesaplaması yapılması gerekmekte olduğunu biliyoruz. Bu bilgiyi dikkate alarak bu eğrinin altında kalan alanı bir integral olarak yazıyoruz.

Artık alanımızı yazdığımıza göre şimdi bu integrali nümerik integrasyon yöntemlerini kullanarak çözmeye başlayabiliriz.

İlk yöntemimiz Yamuk (Trapezyum) Kuralı olup formülü aşağıdaki gibidir. Bu işlemin sonucu bize integralin sonucunu yaklaşık olarak verecektir.

veya formül düzenlenirse

Bu formülde a= integralin alt sınırı, b= integralin üst sınırı ve h= adım aralığıdır. Bu değerler zaten bize soruda aşağıdaki gibi verilmiştir.

, ,

Formülümüze baktığımızda fonksiyona 0.1 adım aralığıyla aralığındaki bütün değerleri koymamız gerektiği anlaşılmaktadır. Yani şeklinde 16 tane hesap yapılması gerekmektedir. Burada yapmamız gereken fonksiyonun içindeki değeri x’de yerine yazmaktır.

Bahsettiğimiz fonksiyon f(x) anlaşılacağı üzere integralin içerisinde yer alan fonksiyondur. Burada her değer için integral çözümü yapmak gibi bir durum söz konusu değildir. Önemli olan integralin içerisindeki fonksiyondur.

Ayrıca Trapezyum Kuralı için hata hesabı formülü aşağıdaki gibi verilmektedir.

Yani hata hesabı için fonksiyonumuzun ikinci türevi alınmalıdır. Sonrasında yine bu soru için çözümümüzde fonksiyonun ikinci türevine 0.1 adım aralığıyla aralığındaki bütün değerleri koymamız gerekmektedir. <> ifadesi ise bu değerlerin toplanıp ortalamasının alınması gerektiğini belirtmektedir. Yani değerleri koyduktan sonra yine 16 tane işlem yapılıp, bu sonuçlar toplanıp işlem sayısına (16) bölünecektir.

Şimdi sırasıyla fonksiyonda değerleri yerine yazarak çözümümüze başlayabiliriz. Burada yine kısalık açısından sadece bir tane işlemi örnek olarak gösterecek olup diğer işlemlerin direkt sonuçları yazılacaktır.

Örnek işlemimiz aşağıdaki gibidir.

Bu şekilde 0.1 adım aralığıyla aralığındaki bütün değerler için aynı işlem uygulanacaktır. İşlemlerin sonuçları aşağıdaki gibidir. (*Son sayfada hesaplama için python kodu ekstra olarak verilmiştir.(\*)*)

Bu değerler hesaplandıktan sonra artık Trapezyum Kuralını uygulayabiliriz.

Şimdi hata hesabımızı yapmamız gerekmekte. Bu hesap için elimizde ve değerleri bulunmaktadır. Fakat değeri elimizde bulunmamaktadır. Şimdi bunun için işlemlere başlıyoruz.

Öncelikle fonksiyonumuzun ikinci türevini almamız gerekmektedir. Fonksiyonumuzun 2 kere türevinin alınması sonucu ortaya çıkan fonksiyon aşağıdaki gibidir.

Yine 0.1 adım aralığıyla aralığındaki bütün değerler için için yaptığımız işlemlerin aynısı uygulanacaktır.

Bu işlemlerin sonuçları bir sonraki sayfadaki gibidir.

Şimdi yapmamız gereken bu değerleri toplayıp işlem sayısına yani 16’ya bölmektir.

Artık Trapezyum Kuralı için hata hesabını yapabiliriz.

Buradan eğri altında kalan alan Trapezyum Kuralına göre

***b) Simpson Kuralı***

Simpson Kuralı için formülümüz aşağıdaki gibidir.

Bu yöntemde öncelikle Trapezyum Kuralı çözümünde yaptığımız gibi fonksiyona 0.1 adım aralığıyla aralığındaki bütün değerleri koymamız gerekmekte olup toplamda 16 tane işlem yapmamız gerekmektedir. Bu işlemler yapıldıktan sonra sonuçlar 4 farklı kısma ayrılacaktır.

değeri aslında bizim Trapezyum Kuralında hesapladığımız değeri yani integralin alt sınırıdır. Aynı şekilde değeri de bir önceki çözümde değeri ile aynı değerdir.

Yaptığımız işlemler sonucu bulduğumuz diğer değerler ise 2 kısma ayrılmaktadır, tek ve çift. İşlemin sonucu tek bir sayı ise , işlemin sonucu çift bir sayı ise olarak formülümüze dahil edilecektir. Bunu çözümde daha ayrıntılı olarak açıklayacağız.

Simpson Kuralı için hata hesabı ise aşağıdaki formül ile yapılmaktadır.

Bu kuralın hata hesabı için fonksiyonumuzun 4. türevi alınıp bu türeve 0.1 adım aralığıyla aralığındaki bütün değerleri koymamız gerekmektedir. Sonrasında işlem sonucu çıkan değerleri toplayıp işlem sayısına yani 16’ya bölmemiz gerekecektir. ve zaten soruda verilmiş durumdadır.

Şimdi Simpson Kuralı ile eğri altında kalan alanı bulmak üzere çözümümüze başlıyoruz.

İlk olarak yapacağımız işlem Trapezyum Kuralında yaptığımız işlem ile aynıdır. Fonksiyona 0.1 adım aralığıyla aralığındaki bütün değerleri koymamız gerekmektedir. Önceki çözümde bunu yaptığımız için kısalık açısından zaten aynı değerler olduğu için direkt o değerleri kullanıyoruz.

Şimdi Simpson Kuralı için formülümüzü kullanabiliriz. Bunun için önce sonuçları tek ve çift olmak üzere gruplandırmak karışıklığı önleme açısından faydalıdır.

*Tek değerler:*

*Çift değerler:*

Bu gruplandırmayı yaptıktan sonra Simpson Kuralını açıklarken bahsettiğimiz üzere tek sayılar olarak yazılıp toplamları 4 ile çarpılacaktır. Çift sayılar ise olarak yazılıp toplamları 2 ile çarpılacaktır.

Bütün bunlar göz önünde bulundurularak öncelikle ve değerlerini buluyoruz. Yani direk tek değerleri ve çift değerleri topluyoruz.

ve

Bu değerler de bulunduktan sonra artık Simpson Kuralı formülümüzü kullanabiliriz.

Şimdi hata hesabını yapıyoruz. ve değerleri zaten elimizde bulunmakta. Fakat değeri elimizde bulunmamaktadır. Şimdi bunun için işlemlere başlıyoruz.

Öncelikle fonksiyonumuzun dördüncü türevini almamız gerekmektedir. Fonksiyonumuzun 4 kere türevinin alınması sonucu ortaya çıkan fonksiyon aşağıdaki gibidir.

Yine 0.1 adım aralığıyla aralığındaki bütün değerler için için yaptığımız işlemlerin aynısı uygulanacaktır. Bu işlemlerin sonuçları aşağıdaki gibidir.

Şimdi yapmamız gereken bu değerleri toplayıp işlem sayısına yani 16’ya bölmektir.

Artık Simpson Kuralı için hata hesabı yapabiliriz.

Buradan eğri altında kalan alan Simpson Kuralına göre

**Matris çözümü için python kodu**

import numpy as np

A = np.array([[16, 136, 1496], [136, 1496, 18496], [1496, 18496, 243848]])

B = np.array([3775.26335, 45950.08194, 600154.5383])

print("Solutions:\n",np.linalg.solve(A, B ))

**Fonksiyon değeri hesaplama için python kodu**

a=float(input("Başlangıç değerinizi giriniz: "))

fx=(1/(1+(2\*pow(a,2))))

b=1

Results=[]

i=0

while i<16:

fx=(1/(1+(2\*pow(a,2))))

Results.append(fx)

a=a+0.1

i=i+1

print("Hesaplama işlemi tamamlanmıştır.")

print(Results)