KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SONLU FARK YAKLAŞIMLARIYLA SAYISAL TÜREV İFADELERİNİN ÜRETİLMESİ, DEĞERLENDİRİLMESİ VE DOKÜMANTASYONU

YÜKSEK LİSANS TEZİ

İbrahim Uğur YILMAZ

HAZİRAN 2024
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SONLU FARK YAKLAŞIMLARIYLA SAYISAL TÜREV İFADELERİNİN ÜRETİLMESİ, DEĞERLENDİRİLMESİ VE DOKÜMANTASYONU

İbrahim Uğur YILMAZ

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih: 05 / 06 / 2024
Tezin Savunma Tarihi : 28 / 06 / 2024

Tez Danışmanı : Doç. Dr. Hüseyin PEHLİVAN

Trabzon 2024

ÖNSÖZ

Bu tez Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Bilimleri Yüksek Lisans Programı'nda yürütülmüş bir çalışmadır. "Sonlu Fark Yaklaşımları ile Sayısal Türev İfadelerinin Üretilmesi, Değerlendirilmesi ve Dokümantasyonu" başlıklı bu çalışma, sayısal türev hesaplamalarında sonlu fark yöntemlerinin uygulanması, değerlendirilmesi ve dokümantasyonu ile ilgilidir.

Bu zorlu akademik yolculukta bana yol gösteren, bilgi ve tecrübelerini benimle paylaşan danışman hocam Doç. Dr. Hüseyin Pehlivan'a şükranlarımı sunarım. Kendisinin değerli tavsiyeleri, bu tezin şekillenmesinde ve nihai sonuçlara ulaşmasında çok önemli olmuştur. Ayrıca, akademik süreçte bana rehberlik eden ve sürecin işleyişini anlatan Teknoloji Fakültesi Yazılım Mühendisliği Bölümü'ndeki mesai arkadaşlarıma sonsuz teşekkür ederim.

Bu süreç boyunca bana her zaman destek olan ve sabır gösteren aileme en içten teşekkürlerimi sunarım. Çalışmalarım sırasında gösterdiği anlayış ve destek için eşim Pınar Yılmaz'a minnettarım. Sevgi ve motivasyonu ile yanımda olan kızım Aybüke Yılmaz'a da sonsuz teşekkürler. Ayrıca, her daim yanımda olan, anlayış gösteren ve destekleyen annem Havva Yılmaz ve babam Cafer Yılmaz'a da şükranlarımı sunarım.

Son olarak, bu çalışmanın alanımdaki bilgi birikimine katkıda bulunmasını ve gelecekteki araştırmalara yol göstermesini umuyorum. Umarım okuyucular sayısal türevler hakkında derinlemesine bilgi sahibi olabilir ve bu bilgileri kendi alanlarında uygulayabilirler.

İbrahim Uğur YILMAZ

Trabzon 2024

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum “Sonlu Fark Yaklaşımlarıyla Sayısal Türev İfadelerinin Üretilmesi, Değerlendirilmesi Ve Dokümantasyonu” başlıklı bu çalışmayı baştan sona kadar danışmanım Doç. Dr. Hüseyin Pehlivan’ın sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 28 / 06 / 2024

İbrahim Uğur YILMAZ

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

SONLU FARK YAKLAŞIMLARIYLA SAYISAL TÜREV İFADELERİNİN ÜRETİLMESİ, DEĞERLENDİRİLMESİ VE DOKÜMANTASYONU

İbrahim Uğur YILMAZ

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. Hüseyin Pehlivan
2024, 73 Sayfa

Sonlu fark denklemleri, analitik çözümlerin uygulanamadığı çeşitli bilimsel ve mühendislik problemleri için yaygın olarak kullanılan matematiksel araçlardır. Mühendislik disiplinlerindeki etkili uygulama alanlarından biri sayısal türev hesaplamalarıdır. Bu tez çalışmasında, sonlu fark yaklaşımları kullanılarak sayısal türev ifadelerinin türetimi, belirli bir noktalar kümesi için değerlendirmesi ve işlem adımlarının dokümantasyonunu gerçekleştiren bir yazılım uygulaması geliştirilmiştir. Uygulama bileşenleri Java programlama çevrelerine entegre edilerek sayısal türev hesaplamalarına destek verilebilmektedir. Türev verileri formal yapıda bir dosya içerisinden okunabileceği gibi başka bir uygulama çevresinden de gönderilebilmektedir. Bu veriler bir LL(k) ayrıştırıcısı ile analiz edildikten sonra Taylor serisi açılımları için soyut sözdizim ağacı denilen bir veri yapısı oluşturulmuştur. Her bir noktanın yönüne uygun olarak seçilen sonlu fark yaklaşımının türüne göre sözdizim ağacının farklı bir düzenlemesi yapılmıştır. Ağaç düzenlemeleri ile ortaya çıkan sonsuz çözümlü doğrusal denklem sistemine Gauss eliminasyon yöntemi uygulanarak çözümlerden biri elde edilmiştir. Türevin mertebesine göre belirlenen ifadelerin türetim adımları bir PDF dokümanına, değerlendirme sonucu ise diğer uygulamalara aktarılabilmektedir.

Anahtar Kelimeler: Sayısal türev, biçimsel gramerler, simgesel hesaplama, belgeleme

Master Thesis

ABSTRACT

GENERATION, EVALUATION AND DOCUMENTATION OF NUMERICAL DERIVATIVE EXPRESSIONS USING FINITE DIFFERENCE APPROXIMATIONS

İbrahim Uğur YILMAZ

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Assoc. Prof. Hüseyin Pehlivan
2024, 73 Pages

Finite difference equations are widely used mathematical tools for a variety of scientific and engineering problems for which analytical solutions cannot be applied. One of the influential application areas in engineering disciplines is numerical derivative calculations. In this thesis, using finite difference approaches, a software application is developed that performs the derivation of numerical differentiation expressions, their evaluation for a specific set of points, and their documentation of the operation steps. Application components can be integrated into Java programming environments to support numerical derivative calculations. Derivative data can be either read from a formal file or sent from another application environment. After these data are analyzed with an LL(k) parser, a data structure called abstract syntax tree is created for Taylor series expansions. A different arrangement of the syntax tree is made depending on the type of finite difference approximation chosen in accordance with the direction of each point. One of the solutions is obtained by applying the Gauss elimination method to the system of linear equations with infinite solutions, which emerge with tree arrangements. The derived steps of the expressions determined according to the order of the derivative can be transferred to a PDF document, and the evaluation result can be done to other applications.

Keywords: Numerical differentiation, formal grammars, symbolic calculation, documentation

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER VE KISALTMALAR DİZİNİ

ASC	: Abstract Syntax Class (Soyut Sözdizim Sınıfı)
AST	: Abstract Syntax Tree (Soyut Sözdizim Ağacı)
BNF	: Backus Naur Form
CFG	: Context Free Grammar (Bağlamdan Bağımsız Gramer)
CSG	: Context Sensitive Grammar (Bağlama Duyarlı Gramer)
EBNF	: Extended Backus Naur Form
JavaCC	: Java Compiler Compiler
LL	: Left-to-right, Leftmost derivation (Soldan Sağa, En Soldan Türetim)
LR	: Left-to-right, Rightmost derivation (Soldan Sağa, En Sağdan Türetim)

1. GENEL BİLGİLER

1.1. Giriş

Bilimsel hesaplamanın temel taşlarından biri olarak kabul edilen sayısal yaklaşımlar, mühendislik ve diğer birçok bilimsel disiplinde kritik bir role sahiptir. Özellikle karmaşık sistemlerin analizi ve kontrolüne yönelik geliştirilen matematiksel modeller, analitik yöntemlerle değerlendirilemeyen çeşitli fonksiyonlar içerebilmektedir. Bu durum, sistem modellerinin bileşenleri üzerinde gerçekleştirilecek matematiksel hesaplamalarda sayısal yaklaşım tekniklerinin kullanımını zorunlu kılmaktadır. Sayısal yöntemler, genel olarak problemlerin belirli bir hata payını karşılayan yaklaşık çözümlerine ulaşmak için kullanılır. Analitik çözüm yöntemlerinin matematiksel modellere uygulanmasının mümkün olmadığı durumlarda, sayısal yöntemler yardımıyla alternatif çözümler sunulabilmektedir.

Sayısal yaklaşımların uygulandığı matematiksel model çözümlemelerinde kesme, hassasiyet ve yuvarlama hataları ortaya çıkmaktadır. Bu hatalar, bilgisayarların sınırlı bellek ve işlem kapasitesi nedeniyle meydana gelir. Kesme hatası, sayısal verilerin belirli bir hassasiyet düzeyinde sınırlandırılması gerektiğinde oluşur. Hassasiyet hatası, sayısal hesaplamaların doğruluğunu etkileyen küçük farklardan kaynaklanır. Yuvarlama hatası ise, sayısal değerlerin belirli bir noktada yuvarlanmasından kaynaklanır ve bu da hesaplama sonuçlarını etkileyebilir. Bu hatalar, özellikle türev hesaplamalarında büyük önem taşır, çünkü türev hesaplamaları, küçük değişikliklerin etkilerini belirlemek için hassas değerlere ihtiyaç duyar. Kesme ve yuvarlama hataları, türev hesaplamalarının doğruluğunu ciddi şekilde etkileyebilir ve sonuçların güvenilirliğini azaltabilir. Bu nedenle, sayısal türev hesaplamalarında bu tür hataların minimize edilmesi büyük önem taşır.

Günümüzde, türev hesaplaması yapabilen pek çok araç bulunmaktadır. Bu araçlar arasında bazıları doğrudan türev hesaplama hizmeti sunarken, diğerleri programlama dillerinin kütüphaneleri olarak daha küçük matematiksel problemler için destek sağlar. Örneğin, matematiksel problemleri çözmek için güçlü bir araç olan Wolfram, doğrudan hizmet sunan bir servistir. Öte yandan, Gradient gibi bir kütüphane, bir programlama diline entegre edilerek türev alma problemleri için kullanılmaktadır. Ancak, bu kütüphaneler belirli matematiksel hesaplamaları gerçekleştirmede yetersiz kalabilir ve Wolfram gibi kapsamlı araçlar ise özel problemler için tasarlanmış uygulamalarla entegre edilemeyebilir

(Pehlivan, 2019). Bu durum mühendisler ve matematik araçları kullanan geliştiriciler için sorunlar yaratmaktadır. Bu eksiklikleri gidermek ve daha doğru türev hesaplamaları yapabilmek için bir Java aracı geliştirilmesi gereklidir. Bu araç, kesme, hassasiyet ve yuvarlama hatalarını en aza indirmek için özel algoritmalar ve tasarım yapıları kullanır. Özellikle mühendislik ve bilimsel araştırma projelerinde sayısal türev hesaplamalarının doğruluğunu artırmayı amaçlayan bu araç, kullanıcılara yüksek doğrulukta türev hesaplamaları yapma imkânı sunar ve mevcut araçların kısıtlamalarını aşar.

Sayısal yaklaşımların simgesel hesaplama yöntemleri ile birleşimi, matematiksel problemleri çözme konusunda faydalar sunmaktadır. Bu birleşim, problemlerin anlaşılmasını ve çözülmesini kolaylaştırabilmektedir. Matlab, Maple, Mathematica ve Maxima gibi simgesel hesaplama da yapabilen genel amaçlı hesaplama araçları, birçok problemin üstesinden gelebilmektedir. Fakat bu hesaplama araçları çözümü istenen problemin bazı özel durumlarını ihmal edebilmektedir. Örneğin, sonlu fark yaklaşımlarının uygulandığı bir problemin çözümü, ilgili noktalar kümesinin yönlerine uygun ve tümünü kapsayacak bir yöntem ile yapılmayabilmektedir.

Matematiksel problemlerin simgesel alana taşınması, özellikle türev problemlerinin ele alınmasında büyük önem taşır. Simgesel programlama araçları, sayısal yöntemlerin aksine, türev işlemlerini tam ve kesin bir şekilde çözebilme avantajına sahiptir. Bu araçlar, çok değişkenli fonksiyonlar veya yüksek dereceli türevler gibi karmaşık türev problemleri üzerinde çalışma kapasitesine sahiptir. Ancak, literatür incelendiğinde sayısal türev problemlerinin Java programlama çevrelerinde simgesel yöntemler ile desteklenmediği görülmektedir. Bu eksikliğin giderilmesi, bilimsel hesaplamalarda sayısal türev problemlerine yönelik çözümlerin etkinliğini artıracaktır. Tezin konusu olan sayısal türev hesaplaması yapabilen Java aracı geliştirilirken literatürdeki bu çalışmalar göz önüne alınmış ve sayısal türev hesaplamalarında daha yüksek doğruluk ve güvenilirlik sağlamayı amaçlamaktadır. Kullanıcılar, bu aracı kullanarak kesme, hassasiyet ve yuvarlama hatalarını minimize edebilir ve daha doğru sonuçlar elde edebilirler.

1.2. Tezin Amacı

Simgesel yaklaşım, matematiksel ifadelerin temsil edilmesi ve değerlendirilmesinde sayılardan ziyade sembollerin kullanılabildiği bir çözümleme tekniğidir. Bu teknikte, harfler ve semboller bir matematiksel ifadenin veya denklemin bileşenlerini temsil eder.

Sembolik yapı, matematik problemlerini anlama ve çözme sürecinde yaygın olarak kullanılır (Kramarski vd., 2003). Simgesel yaklaşımlar, matematiksel problemleri çözmede, algoritma geliştirmede, yazılım tasarım süreçlerinde, bilgisayar cebir uygulamalarında ve bilimsel araştırmalarda kendine yer bulmuştur (URL-6, 2024). Ayrıca sembolik hesaplama, sayısal hesaplama ve yazılım geliştirme süreçlerinin gerektirdiği matematiksel işlemlerin gerçekleştirilmesinde de yaygın olarak kullanılmaktadır.

Bu tezin amacı, bir biçimsel gramer yardımıyla oluşturulan JavaCC ayrıştırıcısına sunulacak noktalar kümesini kullanarak sayısal türev hesaplaması yapabilen ve bu hesaplamaları matematiksel notasyonlara uygun olarak bir dokümana aktarabilen bir aracın geliştirilmesidir. Bu araç, matematiksel ifadelerin simgesel olarak işlenmesi ve türev hesaplamalarının simgesel yaklaşımlarla gerçekleştirilmesi açısından önemli bir yenilik sunmaktadır. Sembolik yaklaşımlar, fonksiyonların türev ifadelerinin noktalar kümesi üzerinden türetilmesinde de güçlü bir yöntem olarak kullanılmaktadır. Bu kapsamda, sembolik hesaplama yöntemleriyle elde edilen sonuçların doğruluğu ve verimliliği, sayısal analiz ve mühendislik uygulamaları için büyük önem taşımaktadır.

Ayrıca, ileri, geri ve merkezi farklar yöntemleri kullanılarak aynı türev probleminin farklı hesaplama teknikleri ile ele alınması hedeflenmiştir. Bunun sonucu olarak, sayısal türev hesaplamaları için en uygun tekniğin kullanılması amaçlanmıştır. Sayısal türev probleminin seçilmesindeki amaç, Java içerisinde problemi simgesel olarak işleyebilen ve sayısal türev hesaplayabilen herhangi bir aracın bulunmamasıdır.

Bu genel amaç doğrultusunda çalışmanın aşağıda listelenen alt amaçları ortaya çıkmıştır.

- Fonksiyon noktalarının yönlerine (ileri ve/veya geri) en uygun türev formüllerinin belirlenmesi,
- Nokta sayısına bağlı olarak yüksek dereceli türev ifadelerinin türetilmesi,
- Türetilen ifadelerin uygun matematiksel notasyonlar kullanılarak gösterimlerini yapabilen kullanıcı platformlarının geliştirilmesi,
- Türetilen ifadeler üzerinde noktasal verilerin değerlendirilmesi,
- Sonlu fark yaklaşımlarının Taylor serisine dayalı olarak kullanılması,
- Hesaplama işlemlerinde farklı sayı tiplerinin desteklenmesi
- Yuvarlama hataları olmadan sayısal türev hesaplaması yapabilen bir Java aracının oluşturulması,

- Türev hesaplamalarını içeren bir dokümanın üretilmesi ve bu dokümanın mühendislik alanında, özellikle sayısal analiz derslerinde yardımcı bir kaynak olarak kullanılabilmesi.

1.3. Literatür Taraması

Bilgisayar bilimi, veri işleme, algoritma tasarımı ve bilgisayar sistemlerinin çalışma prensipleri ile ilgilenen bir disiplindir. Sayısal analiz, matematiksel problemleri sayısal olarak çözmek için kullanılan yöntemlerin ve algoritmaların incelendiği bir matematik dalıdır. 1950 yılından itibaren kapasite artışının yanında maliyeti azalan bilgisayarların teknolojik gelişimi, sayısal analiz alanında yapılan çalışmaları hızlandırmıştır (Chapra, 2010). Özellikle türev hesabının bilgisayarlar yardımıyla daha kolay yapılabilmesi nedeniyle sayısal analiz yöntemlerine başvurulmuş ve birçok bilgisayar destekli uygulamada tercih edilmiştir. Son yıllarda yüksek hesaplama kapasiteli dijital bilgisayarların gelişmesiyle birlikte sayısal yöntemlerin mühendislik problemi çözmedeki rolü artmıştır (Chapra, 2010).

Sayısal çözümleme yöntemleri, matematikçilerin yanı sıra mühendisler tarafından da birçok bilimsel hesaplamada önemli bir problem çözme aracı olarak kullanılmaktadır. Özellikle analitik yaklaşımların uygulanamadığı veya yetersiz kaldığı durumlarda anahtar bir rol oynarlar (Vatansever, 2018). Ayrıca çeşitli matematiksel denklemlerde ve sayısal analiz işlemlerinde etkin işlevler icra ederler. Bu yaklaşımlar, fonksiyonları belirlenen bir doğruluk seviyesinde ve mevcut hesaplama kapasiteleri dâhilinde temsil etmeye yönelik karmaşık matematiksel işlemlerden oluşur. Ayrıca, zor ve anlaşılması güç yapıların ile sistemlerin daha iyi anlaşılmasını sağlar ve teorik modellerin pratikte uygulanabilirliğini artırır, böylece bilim ve mühendislik alanlarındaki yeniliklere kapı açar.

Sayısal analiz, mühendislik problemlerinin çözümüne yardımcı olan ve farklı matematiksel teknikleri içeren bir hesaplama disiplinidir. Bu disipline yönelik birçok çalışma bulunmaktadır, fakat bilimsel alanlara bakıldığında, yalnızca kendi kapsamı ile sınırlı çalışmalar değil, aynı zamanda farklı bilimsel alanlardaki problemlerin çözüm sürecine destek vermek üzere çeşitli araştırmaların da yapıldığı görülmektedir. Sayısal yaklaşımlar birçok çalışmada yardımcı araç olarak kullanılmaktadır. Bu açıdan, söz konusu yaklaşımlar karmaşık matematiksel problemlerin çözümünde mühendislerin ve

araştırmacıların temel hesaplama araçları arasında bulunmaktadır (Deuflhard & Hohmann, 2003).

Çalışmalarda kullanılan sayısal yaklaşımlar problemi ele alış şekli itibariyle araştırmacılara kolaylık sağlamaktadır. Sonlu fark yöntemleri, diferansiyel denklemleri ve diğer matematiksel problemleri çözmek için kullanılır (URL-11, 2024). Bu yaklaşım, denklem çözümü için gereken matematik işlemleri basitleştirir ve özellikle bilgisayarlarla yapılan hesaplamalar için uygun sayısal sonuçlar elde etmeye olanak sağlar. Araştırmacılar, yerel olmayan difüzyon, modelleme ve dinamik sistemler de dahil olmak üzere birçok problem türünü çözebilecek sayısal yöntemler ortaya çıkarmıştır (Du vd, 2012).

Programlama dilleri iki ana kategoriye ayrılır; genel amaçlı ve özel amaçlı diller. Genel amaçlı diller, çeşitli bilgisayar uygulamalarını yüksek seviyeli programlama yapıları ile geliştirmek için oluşturulmuştur. Ancak, bu diller matematiksel hesaplamalar için her zaman uygun olmayabilir. Özel amaçlı diller ise, matematiksel notasyonlardan farklı söz dizimleri ve fonksiyonel davranışları nedeniyle, kod geliştirme ve doğrulama süreçlerini karmaşık hale getirebilir (Pehlivan, 2019).

Bilgisayar cebir sistemleri, fonksiyonlar, integraller, denklem sistemleri, vektör ve matris manipülasyonları gibi matematiksel nesnelerin sembolik olarak işlenmesi için tasarlanmış yazılımlardır (Belu vd., 2009). Bu sistemler, çeşitli matematiksel görevleri bilgisayar yardımıyla çözmeyi ve hesaplamaların tüm aşamalarını görselleştirmeyi sağlar (Velychko vd., 2019). Ayrıca, bu sistemler sayısal hesaplamalar için algoritmalar üreterek, hesaplamalı bilimlerde önemli araçlar haline gelmiştir (Gander, 2007). Sayısal türev hesaplamalarını yapabilen birçok uygulama ve platform geliştirilmiştir (Pehlivan, 2019). Bu uygulamalar, zor problemleri çözmenin yanı sıra, diğer birçok matematik işlemini de gerçekleştirebilen genel amaçlı yazılımlardır.

Matematik problemlerine çözüm üreten programlar arasında özel amaçlı olanlar da bulunmaktadır. Örneğin, AMPL, Lingo, GAMS, Gurobi ve CPLEX gibi programlar, belirli matematiksel ve optimizasyon problemlerini çözmek için geliştirilmiştir. AMPL, matematiksel modelleme ve optimizasyon problemlerini çözmek için kullanılan bir dildir ve lineer programlama, karmaşık programlama, tamsayı programlama gibi çeşitli problemleri çözebilir. Farklı optimizasyon motorlarıyla entegre olabilen bir arayüz sunar (Fourer vd., 1990). Lingo, lineer, karışık tamsayı ve doğrusal olmayan programlama problemlerini çözmek için kullanılan bir program olup, ağ modelleme, zaman serisi analizi

ve simülasyon gibi problemleri de çözebilir. Kullanıcı dostu bir arayüze sahiptir ve çeşitli optimizasyon tekniklerini destekler (Schrage, 2006). GAMS, matematiksel modelleme ve optimizasyon problemlerini çözmek için kullanılan bir dil ve çevredir. Lineer programlama, karar verme analizi ve karma tamsayı programlama gibi çeşitli matematiksel problemleri çözebilir (Rosenthal, 2007). Gurobi, yüksek performanslı matematiksel modelleme ve optimizasyon yazılımıdır. Lineer programlama, karışık tamsayı programlama ve doğrusal olmayan programlama gibi problemleri çözebilir, kullanıcı dostu araçlar ve API'ler sunar (Anand vd., 2017). IBM tarafından geliştirilen CPLEX, güçlü optimizasyon algoritmaları ve kullanıcı dostu arayüzüyle matematiksel modelleme ve optimizasyon problemlerini çözmek için kullanılır (Bliek, 2014).

Literatürde birçok algoritma, bilgisayarlar aracılığıyla matematik problemlerini çözmek için geliştirilmiştir. M.Ö. 3. yüzyılda Euclides tarafından keşfedilen tamsayıların en büyük ortak bölenini hesaplama yöntemi, günümüzdeki simgesel hesaplama algoritmalarına temel teşkil eden ilk örneklerden biri olarak kabul edilebilir (Aydoğdu, 2016). Simgesel hesaplama, cebir gibi matematiksel yapıları analiz etmek ve sınıflandırmak için matematikte de uygulanmaktadır (Kutsenko, 2021). Bu yöntemler, temel olarak matematiksel işlemlerin bilgisayarlar aracılığıyla kesin olarak çözülebilmesi amacıyla tasarlanmıştır. Etkin bir şekilde kullanılabilmeleri için ilk adım, üzerinde çalışılacak matematiksel problemlerin açık ve net bir şekilde tanımlanmasıdır (Yılmaz, 2019). Daha sonra, çözümler bilgisayar algoritmaları olarak formüle edilmelidir.

Matematik problemleri üzerine literatürde geniş bir araştırma yelpazesi mevcuttur. Bu problemlerin çözüm yolları, simgesel programlama veya genel amaçlı programlama gibi alanlarda ele alınmıştır. Simgesel çözüm üreten birçok araç bulunmaktadır; bu araçlar arasında Maple (URL-7, 2024), Mathematica (URL-8, 2024), Matlab (URL-9, 2024) ve Maxima (URL-10, 2024) gibi önde gelen programlar yer almaktadır. Simgesel çözüm üretemeyen genel amaçlı programlama dillerine örnek olarak Python (Lutz, 2001), R (Crawley, 2012) ve C++ (Josuttis, 2012) gösterilebilir.

Programlama dili oluşturma süreci, dilin söz diziminden yorumlamaya veya doğrudan makine diline çevirme işlemine kadar birçok karmaşık bileşeni kapsar. Bu süreçte önemli bir yer tutan YACC, SableCC ve JavaCC gibi araçlar, otomatik dil ayrıştırıcıları üretmektedir (Pehlivan, 2019). Bu tür araçlar, programlama dilinin gramerine dayalı olarak işlev görür ve her biri dilin söz dizimini tanımlamak için özgün formatlar kullanır. Örneğin, JavaCC, Java dilinin söz dizimini, dil gramerinden elde edilen çeşitli

tanım biçimleri kullanarak işler. Bu araçlar, programlama dillerinin gelişiminde hayati bir rol oynar, zira dilin temel öğelerini otomatik olarak tanımlama yeteneğine sahiptirler.

Bilgisayar bilimlerindeki temel sınırlamalardan biri, tamsayılar ve reel sayıların tam bir temsilinin mümkün olmamasıdır. Bu durum, kesme, hassasiyet ve yuvarlama hatalarının önemli olduğu senaryolarda büyük bir önem kazanmaktadır. Bilimsel alanlarda hataların etkilerini azaltmaya yönelik çeşitli çalışmalar yapılmıştır. Yeni bir otomatik yöntem, sayısal algoritmaların doğruluğunu ve kararlılığını artırmak için algoritmik farklılaşmayı, temel yuvarlama hatası hesaplamasını ve çalıştırma hatası analizini kullanarak kayan nokta yuvarlama hatalarını düzeltir (Langlois, 2001). Yuvarlama hatalarını en aza indirmek ve rastgele yürüyüş gibi yayılmak için simetrik çok adımlı yöntemler uygulanabilir (Console & Hairer, 2014). Gradyan minimizasyon yöntemleri için bir durdurma kriteri oluştururken yuvarlama hatalarını dikkate almak, gerçek uygulamalı problemler ve hesaplama açısından küçük problemler için çözümlerin kalitesini artırabilir (Lukyanenko vd., 2022). Bu çalışmalar, hataların azaltılması ve doğruluğun artırılmasına yönelik önemli katkılar sağlamaktadır

1.4. Sayısal Türev ve Uygulamaları

Türev, matematiğin en temel kavramlarından biridir ve hem matematiğin çeşitli dallarında hem de uygulamalı matematikte geniş bir kullanım alanına sahiptir. Ayrıca, birçok araştırmacı ve mühendis tarafından da önemli bir araç olarak kabul edilmektedir. Türevlerin, bir fonksiyonun değişim hızını açıklamak için kullanıldığı bilinmektedir (Pandey & Singh, 2021). Birçok pratik problemde, fiziksel sistemlerin anlaşılması, orijinal fonksiyonun türevlerinin hesaplanmasını gerektirir. Ancak, türev alma teknikleri mühendisler arasında yaygın olarak kullanılsa da süreç içerisinde karşılaşılan karmaşık ve zor problemlerin türevinin alınması mühendislere problem oluşturmaktadır. Hatta yüksek dereceli türevlerin hesaplaması gerektiren bir durumla karşılaşılırsa işler iyice çıkmaza girebilmektedir. Bu zorluk, bazen basit problemlerin bile oldukça karmaşık hale gelmesine neden olarak, türev hesaplamasından kaçınılmasına yol açabilmektedir. Analitik türev alma problemini ortadan kaldırmak için mühendisler projelerinde genellikle sayısal türev yaklaşımlarını tercih etmektedir.

Sayısal türev, bir fonksiyonun veya fonksiyonun bilinen noktasal değerlerinden değişim hızının yaklaşık olarak hesaplanması işlemidir. Bu işlem herhangi bir analitik

türev alma süreci gerektirmez. Böylece karmaşık fonksiyonların türevlerini hesaplamak veya deneysel verilerden türev elde etmek mümkün hale gelir. Sayısal türev, özellikle analitik çözümlemelerin zor veya imkânsız olduğu durumlarda büyük avantaj sağlar. Ayrıca, deneysel veriler üzerinde doğrudan çalışmaya olanak tanıyarak, çeşitli bilimsel ve mühendislik problemlerinin çözümüne destek verir. Bu yaklaşım, bilgisayarların yüksek hesaplama gücü sayesinde hızlı ve etkili bir şekilde gerçekleştirilebilir. Böylece, gerçek dünya verileri üzerinde analiz yapmak ve bu verilere dayalı tahminler ve optimizasyonlar gerçekleştirmek mümkün hale gelir. Sayısal türevi daha etkili kullanabilmek için bazı yeni ve kararlı algoritmalar geliştirilebilirse, mühendisler için çok daha güçlü bir araç haline gelebilir (Cheng, 2007). Ayrıca sayısal türevi hesaplayabilen bir araç geliştirilmesi de mühendislerin birçok problemi kolaylıkla aşmasını sağlayabilir.

Sayısal türev hesaplama için birçok yöntem bulunmaktadır. En yaygın ve en basit olarak kullanılan yöntem sonlu farklar yöntemidir (Cheng, 2007). Sonlu farklar yöntemi, fonksiyonun belirli noktalarındaki değerlerini kullanarak türev hesaplamayı sağlar. Bu yöntem, analitik türev almadan fonksiyon noktaları arasındaki farkları kullanarak türev hesaplar. Ancak, doğru sonuçlar elde etmek için seçilen noktaların önemi büyüktür. Özellikle yüksek dereceli türev hesaplamalarında, denklem sistemlerinden çeşitli katsayıların elde edilmesi gereklidir.

Sayısal türev hesaplamasında bir diğer yöntem olan interpolasyon yöntemleri de sıklıkla tercih edilmektedir. İnterpolasyon, verilen bir fonksiyonun belirli noktalardaki değerlerine dayanarak, bilinmeyen bir noktadaki değeri bulmamızı sağlayan fonksiyonlar üretme işlemidir. Bu yöntem, verilen noktaları üzerinde bulunduran bir fonksiyonun belirlenmesine ve elde edilen fonksiyona istenilen türev işlemi uygulanarak yüksek dereceli türevlerin hesaplamasına dayanır. Ancak, fonksiyonun türevini alma işlemi bir analitik türev alma işlemidir ve bu, sayısal türev hesaplamadan farklı bir yaklaşımdır.

Diğer sayısal türev hesaplama yöntemleri arasında diferansiyel quadrature yöntemi ve spektral yöntemler de bulunmaktadır. Diferansiyel quadrature yöntemi, yüksek hassasiyet gerektiren problemlerde kullanılır ve türev hesaplamalarını daha az noktada yüksek doğrulukla yapar (Bert & Malik,1996). Spektral yöntemler ise Fourier ve Chebyshev serilerini kullanarak sayısal türev hesaplamada yüksek doğruluk sağlar. Chebyshev serisi sayısal türev hesaplama noktasında Fourier serisine göre daha hızlı ve etkili çözümler sunar. Örneğin Chebyshev spektral yöntemi, doğrusal olmayan dinamik modeller için uzayda verimli bir şekilde ayrıklaştırma yapar, periyodik sınır koşullarından

kaçınır ve Fourier trigonometrik fonksiyonlara kıyasla doğruluğu ve yürütme süresini artırır (Saad, 2019).

Bir başka yaklaşım ise, sayısal türevin doğruluğunu artırabilen Richardson ekstrapolasyonudur (Richardson, 1910). Ayrıca, sayısal türev, daha ileri teknikler kullanılarak da gerçekleştirilebilir, örneğin kompleks adım türev yaklaşımı (Martins et al., 2003) ve adaptif sonlu fark yöntemi (Fornberg, 1996). Bu yöntemler daha doğru sonuçlar sağlayabilir ve genellikle hesaplamalı akışkanlar dinamiği ve diğer alanlarda kullanılır. Ek olarak, sayısal türev, makine öğrenimi ve veri analizinde de uygulamalara sahiptir. Örneğin, bir fonksiyonun gradyanını hesaplamak için kullanılabilir, bu da optimizasyon algoritmalarında esastır (Goodfellow et al., 2016).

Sonlu fark yöntemleri literatürde birçok alanda tercih edilmektedir. Sayısal türev teknikleri, çeşitli alanlarda önemli uygulamalara sahiptir. Örneğin, görüntü işlemekte süreksiz noktaların belirlenmesi ve Abel integral denkleminin çözümü gibi işlemlerde kullanılır (Gorenflo & Vessella, 1991). Ayrıca matematiksel ve fiziksel denklemlerden kaynaklanan bazı zor problemlerin çözümünde de sıklıkla tercih edilmektedir (Hanke & Scherzer, 2001).

Dadashi ve Sari (2022) doğal potansiyel verilerinin ikinci ve dördüncü mertebeden türev analizini kullanarak gömülü yapıların derinliğini ve şeklini belirlemek için sayısal türev değerlerini değerlendirmenin önemine vurgu yapar. Bu durum, sayısal türevin jeofizik alanında nasıl kullanılabileceğini göstermektedir. Hussain ve diğerleri (2023) ikinci dereceden bulanık adi diferansiyel denklemlerin sayısal çözümünü üçüncü ve dördüncü türevlerle iki adımlı bir blok yöntemi kullanarak tartışmakta ve karmaşık modellere yaklaşık çözümler sağlamada sayısal tekniklerin uygulanmasını göstermektedir. Pooseh ve diğerleri (2012) kesirli türevlerin sayısal yaklaşımlarına ve bunların uygulamalarına odaklanarak, ilgili yaklaşımların fonksiyonların kesirli türevlerini hesaplamak için nasıl değerli araçlar olduğunu göstermiştir. Ek olarak, Qureshi ve diğerleri (2019) Caputo-Fabrizio kesirli türev operatörünün yeni sayısal yönlerini keşfederek, hata analizi ve deneysel testler yoluyla difüzyon dalga denklemleri de dahil olmak üzere matematiksel modellerin sayısal çözümünü göstermektedir.

Yüksek mertebeden türevler, bir fonksiyonun değişim hızı veya eğriliği hakkında ayrıntılı bilgi gerektiren çeşitli alanlarda gereklidir. Örneğin, Secmen & Turhan-Sayan (2008), farklı alan dereceleriyle sonuçlanan doğrudan yansıma ve sürünen dalgaların kombinasyonu nedeniyle elektromanyetik hedef tanıma yöntemlerinde yüksek dereceli

türevlerin nasıl ilgili olduğunu tartışmaktadır. Ayrıca, malzeme bilimi ve yapı mühendisliğinde, yüksek dereceli türevler çok önemli bir rol oynamaktadır. Örneğin, Kotan (2018) katkı maddelerinin mekanik alaşımlama ile üretilen nano kristal östenitik paslanmaz çelik alaşımlarının tane büyümesi ve sertliği üzerindeki etkilerini araştırarak tane boyutunun azaltılmasının malzeme uygulamalarını nasıl genişletebileceğini göstermiştir. Kotan (2018), yüksek korozyon direnci, mukavemeti ve sertliği nedeniyle endüstriyel uygulamalarda yüksek dereceli östenitik paslanmaz çeliklerin kullanımını vurgulamakta ve çeşitli mühendislik uygulamalarında yüksek dereceli malzemelerin öneminin altını çizmektedir.

Son olarak bir fonksiyonun yaklaşık olarak temsil edilmesini sağlayan Taylor açılımında gerçek değere daha yakından yaklaşabilmek için yüksek mertebeden türev ihtiyacı vardır. Fizik alanında Jerk hesaplamalarında, yüksek dereceli dalga denklemlerin hesaplanmasında ve kuantum mekaniği alanında doğrusal olmayan Schrödinger denklemleri hesaplamalarında da yüksek dereceden türev alma ihtiyacı bulunur. Ayrıca elastik kirişlerin eğilme analizi için 4. dereceden türev hesabı gerekmektedir.

1.5. Taylor Serisi

Birçok bilim adamı seriler ile ilgilenmiştir. Filozoflar başlangıçta sonlu bir sonuç elde etmek için sonsuz seriyi toplama problemini düşünmüş fakat reddetmiştir (Url-3, 2023). 1715 yılına dek, fonksiyonların bu tür serilere dönüştürülmesi için bir yöntem Brook Taylor tarafından tanımlanmıştır ve bu seriye onun adı verilmiştir. 18. yüzyılda, Taylor'ın bu sonucunun bir özel durumu olan Maclaurin serisi, Edinburgh'da görev yapan Profesör Colin Maclaurin tarafından tanıtılmıştır (URL-2, 2023).

Taylor serisi ise bir Taylor teoreminin genelleştirilirmiş halidir. Fonksiyonun açılımı ne kadar genişler ise türev alma iş yükünü beraberinde getirir fakat doğru sonuca yaklaşımda bir o kadar artar. Eğer fonksiyon bir kuvvet serisi olarak gösterilebiliyorsa, Taylor serisinin toplamına eşit olduğunu gösterilebilir. Ancak Taylor serilerinin toplamına eşit olmayan fonksiyonlar da vardır (Stewart, 2007). Bu yüzden seçilecek olan fonksiyon önem arz etmektedir. Taylor serisinin fonksiyon açılımı aşağıda gösterilmiştir.

$$f(a) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \qquad (1)$$

Taylor serisinin uygulama alanları çok geniştir. Birçok navigasyon uygulamasında tipik olan çok ölçümlü karışık modlu konum belirleme problemlerinin çözümünde kullanışlıdır. Yakınsama kanıtlanmamış olmasına rağmen, örnekler çoğu problemin makul başlangıç tahminlerinden doğru çözüme yakınsadığını göstermektedir (Foy, 1976). Yöntem ayrıca çözüm hatalarının istatistiksel yayılımını da sağlamaktadır. Bir diğer çalışmada bir iletim kanalındaki bilinmeyen katkı gürültüsünün ve bilinmeyen doğrusal filtrelemenin konuşma istatistikleri üzerindeki etkilerini verimli ve doğru bir şekilde karakterize etmek için bir vektör Taylor serisi genişlemesinin kullanımı tanıtılmıştır (Moreno vd., 1996). Diferansiyel denklemlerin sayısal yöntemlerle çözülmesinde ve optimizasyon algoritmalarında da yeri bulunmaktadır (Bai, 1992; He, 2023). Ayrıca mekanik ve mühendislikte, özellikle elastik manipülatörlerin diferansiyel denklemlerinin doğrusal hale getirilmesinde kullanılır (Khang vd., 2019). Bilgisayar alanında makine öğrenimi ve optimizasyon algoritmalarında, özellikle politika optimizasyonunda kullanılır (Tang vd., 2020). Görüntü analizi ve işleme alanında, görüntü özelliklerinin analizi ve renk transferi gibi işlemler için kullanılır (Liu vd.,2004). Radar sistemlerinde ise görüntülemede sinyal özelliklerinin analizinde ve görüntüleme algoritmalarının geliştirilmesinde kullanılır (Xie vd., 2009). Finans ve ekonomi alanında iflas tahmin modellerinde, özellikle lojistik regresyon modellerinde kullanılarak sınıflandırma doğruluğunu artırır (Laitinen & Laitinen, 2000). Matematik ve teorik alanlarda Taylor açılımı, analitik fonksiyonların çok noktalı açılımlarında ve integral asimptotik açılımlarında kullanılır (López & Temme, 2002). Çok boyutlu vektör fonksiyonlarının GPU üzerinde verimli hesaplanmasında kullanılır (Skala, 2021).

Sonuç olarak Taylor açılımı, fizik, mühendislik, bilgisayar bilimi, finans ve matematik gibi çeşitli alanlarda geniş bir uygulama alanına sahiptir. Bu yöntem, karmaşık fonksiyonların daha basit fonksiyonlarla temsil edilmesini sağlayarak, sayısal yöntemler, optimizasyon, görüntü işleme ve iflas tahmini gibi birçok alanda önemli katkılar sunmaktadır. Taylor açılımının bu geniş kullanım alanları, onun bilimsel ve mühendislik problemlerinin çözümünde ne kadar önemli olduğunu göstermektedir.

Sonlu farklar yaklaşımı, analitik olarak türevi alınması zor olan fonksiyonların veya bilinmeyen bir fonksiyon tarafından üretilen örneklenmiş verinin türevini bulmak için yaygın olarak kullanılır (Khan & Ohba, 2000). Sonlu farklar yöntemi, Taylor açılımını içeren matematiksel ifadelerden türetilmiştir. Taylor serisi, bir fonksiyonun belirli bir noktadaki değerini ve türevlerini kullanarak o fonksiyonun yakın çevresindeki değerlerini

yaklaşık olarak hesaplamamızı sağladığı için sayısal türev yöntemleri genellikle Taylor serisi kullanılarak türetilir. (Collatz, 2012).

1.6. Sonlu Fark Yöntemleri

Sayısal analizin teorik yönlerinin bilgisayar programlama ve ilgili konularla olan hesaplarıyla birleştiği, farklı oranlarda birçok makale bulunmaktadır (Hildebrand, 1987). Bunlar üzerinde çalışma yapanlara Moursund ve Duris, Pennington ve Arden ve Astill dahildir (Hildebrand, 1987). Sayısal yöntemler, matematiksel problemlerin aritmetik işlemlerle çözülebilecek şekilde formüle edildiği tekniklerdir.

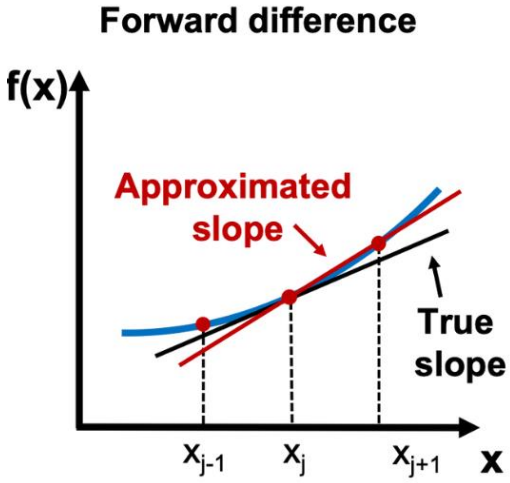
Sonlu fark yöntemleri, fonksiyonun türevini analitik olarak elde etmenin zor olduğu durumlarda kullanılan yaklaşık hesaplama yapabilen yöntemlerdir. Sayısal türev, matematiksel fonksiyonların türevlerini, analitik yöntemlerle değil de sayısal verilere dayanarak yaklaşık olarak hesaplamak için kullanılan bir yöntemdir (Kagiwada vd., 2013). Sayısal türevler, belirli bir noktada fonksiyonun eğimini, hesaplamak için önemli bir araçtır. Örneğin optimizasyon algoritmalarında sonlu fark aralıklarının dikkatlice seçildiği durumlarda türevler için sonlu fark yaklaşımlarıyla oldukça iyi performans gösterebilir (Barton, 1992).

Fonksiyon ve fonksiyonların sayısal türevlerinin hesaplanmasında üç temel sonlu fark yaklaşımı vardır; İleri farklar yöntemi, geri farklar yöntemi ve merkezi farklar yöntemi (Url-1, 2023).

1.6.1. İleri Farklar Yöntemi

İleri farklar yöntemi, bir fonksiyonun türevini hesaplamak için kullanılan basit bir sayısal türev yöntemidir. Bu yöntemde, bir noktada x ve bir sonraki noktada x+h konumunda bulunan iki noktadan elde edilen fonksiyon değerleri kullanılır. İleri farklar yönteminde, h adı verilen küçük bir artım değeri kullanılır.

$$f'(x) \approx \frac{\{f(x+h) - f(x)\}}{\{h\}} \tag{2}$$



Şekil 1. İleri farklar hesaplaması ile yaklaşık eğim (Url-4, 2024)

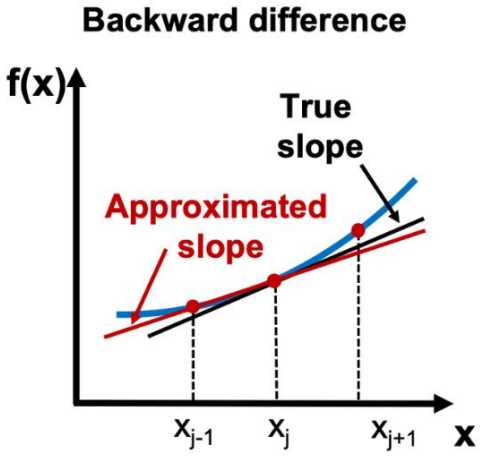
Şekil 1'deki grafikte, yatay eksen x eksenini, dikey eksen ise f(x) eksenini temsil etmektedir. Fonksiyonun değerlerinin bilindiği ardışık noktalar x_{j-1}, x_j ve x_{j+1} ile gösterilmiştir. Mavi eğri, fonksiyonun gerçek eğrisini temsil ederken, kırmızı noktalar fonksiyonun bu noktalardaki değerlerini göstermektedir. Kırmızı noktalar, ileri farklar yöntemi ile hesaplanan yaklaşık türevi ifade ederken, siyah çizgi gerçek türev değerini göstermektedir.

Denklem 2'de, $f'(x_j)$, x_j noktasındaki türevin yaklaşık değerini temsil etmektedir. Grafikte kırmızı noktalar ile gösterilen yaklaşık eğimi ileri farklar yöntemi ile elde edilen türev değeridir. Gerçek eğim ise fonksiyonun gerçek türev değerini ifade eder ve siyah çizgi ile gösterilmiştir. İleri farklar yöntemi, x_j noktasındaki türevi hesaplarken, x_j ve x_{j+1} noktalarındaki fonksiyon değerlerini kullanır.

1.6.2. Geri Farklar Yöntemi

Geri farklar yöntemi, fonksiyonun türevini hesaplamak için bir diğer basit sayısal türev yöntemidir. Bu yöntemde, bir noktada x ve bir önceki noktada x-h konumunda bulunan iki noktadan elde edilen fonksiyon değerleri kullanılır. Geri farklar yönteminde, *h* adı verilen küçük bir azalma değeri kullanılır. Aşağıdaki formülde, f(x-h) fonksiyonun bir önceki noktadaki değerini temsil eder.

$$f'(x) \approx \frac{\{f(x) - f(x-h)\}}{\{h\}} \tag{3}$$



Şekil 2. Geri farklar hesaplaması ile yaklaşık eğim (Url-5, 2024)

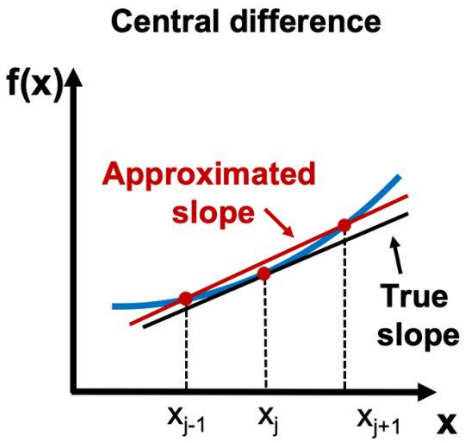
Şekil 2'deki grafikte, yatay eksen x eksenini, dikey eksen ise $f(x)$ eksenini temsil etmektedir. Fonksiyonun değerlerinin bilindiği ardışık noktalar x_{j-1}, x_j ve x_{j+1} ile gösterilmiştir. Mavi eğri, fonksiyonun gerçek eğrisini temsil ederken, kırmızı noktalar fonksiyonun bu noktalardaki değerlerini göstermektedir. Kırmızı noktalar, geri farklar yöntemi ile hesaplanan yaklaşık eğimi ifade ederken, siyah çizgi gerçek türev değerini göstermektedir.

Denklem 3'te, $f'(x_j)$, x_j noktasındaki türevin yaklaşık değerini temsil etmektedir. Grafikte kırmızı noktalar ile gösterilen yaklaşık eğimi geri farklar yöntemi ile elde edilen türev değeridir. Gerçek eğim ise fonksiyonun gerçek türev değerini ifade eder ve siyah çizgi ile gösterilmiştir. Geri farklar yöntemi, x_j noktasındaki türevi hesaplarken x_j ve x_{j-1} noktalarındaki fonksiyon değerlerini kullanır.

1.6.3. Merkezi Farklar Yöntemi

Merkezi farklar yöntemi, daha hassas bir türev tahmini sağlamak için kullanılan bir sayısal türev yöntemidir. Bu yöntemde, bir noktada x-h ve bir noktada x+h konumunda bulunan iki noktadan elde edilen fonksiyon değerleri kullanılır. Merkezi farklar yönteminde hem ileri hem de geri farklar yöntemlerinde olduğu gibi h adı verilen küçük bir artım değeri kullanılır. Aşağıdaki formülde, f(x+h) ve f(x-h) fonksiyonun bir sonraki ve bir önceki noktadaki değerlerini temsil eder.

$$f'(x) \approx \frac{\{f(x+h) - f(x-h)\}}{\{2h\}} \tag{4}$$



Şekil 3. Merkezi farklar hesaplaması ile yaklaşık eğim (Url-4, 2024)

Şekil 3'teki grafikte, yatay eksen x eksenini, dikey eksen ise $f(x)$ eksenini temsil etmektedir. Fonksiyonun değerlerinin bilindiği ardışık noktalar x_{j-1}, x_j ve x_{j+1} ile gösterilmiştir. Mavi eğri, fonksiyonun gerçek eğrisini temsil ederken, kırmızı noktalar fonksiyonun bu noktalardaki değerlerini göstermektedir. Kırmızı noktalar, merkezi farklar yöntemi ile hesaplanan yaklaşık türevi ifade ederken, siyah çizgi gerçek türev değerini yani eğimi göstermektedir.

1.7. Doğrusal Denklem Sistemleri

Doğrusal denklem sistemleri, birden fazla değişken içeren ve bu değişkenler arasında doğrusal ilişkiler kuran denklemlerden oluşur. Bu tür sistemler genellikle şu şekilde ifade edilir:

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$

...

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$

Burada x_1, x_2, ..., x_n bilinmeyenleri, a_{ij} katsayıları ve b_i sabit terimleri temsil eder. Sistemin genel formu matris notasyonu kullanılarak AX = B biçiminde yazılabilir.

Doğrusal denklem sistemlerini çözmek için çeşitli yöntemler kullanılabilir. Bu yöntemlerin kullanımı ile ilgili birçok kitap ve kaynak bulunmaktadır. Bu kaynaklarda öne çıkan başlıca yöntemler aşağıda verilmiştir.

- Gauss Eliminasyon Yöntemi: Gauss eliminasyonu, doğrusal denklem kümelerinin çözülmesinde kullanılan yaygın bir sayısal algoritmadır ve bilimsel hesaplama alanında temel bir öneme sahiptir. Lineer cebirde Gauss eliminasyon, bir lineer sistemin tüm çözümlerini bulma, determinant hesaplama ve bir matrisin derecesini belirleme gibi çeşitli amaçlar için kullanılır. Matris analizi ve hesaplamalarında da Gauss eliminasyonunun önemi vurgulanır (Higham, 2011). Ayrıca pratik uygulamalarda, Gauss eliminasyonunun matematiksel problemleri çözmedeki faydası gösterilmiştir (Johar, 2020). Gauss eliminasyon metodu günümüzde birçok alanda da kullanıldığı gibi en küçük kareler yöntemi ile uyum içerisinde çalışmaktadır (Gratton, 2007). Bu iki yöntem günümüzde özellikle zorlu matematik problemlerini içeren uygulamalar yapay zekâ, veri bilimi, makine öğrenmesi ve derin öğrenme gibi birçok alanda kullanılmaktadır. Gauss eliminasyon metodu problemin türüne göre bazı değişiklikler eklenerek zamanla değişime uğramıştır. Sonuç olarak farklı gauss metotları ortaya çıkmıştır.
- Gauss-Jordan Yöntemi: Gauss-Jordan yöntemi, Gauss eleme yönteminin bir modifikasyonudur. Gauss eleme işlemi, matristeki her pivot elementinin altını sıfırlarla doldurarak, üstten başlayıp aşağıya doğru ilerler. Buna karşın Gauss-Jordan eleme yöntemi, her pivot elementinin hem altını hem de üstünü sıfırlarla doldurarak bir adım daha ileri gider. Her matrisin bir satır indirgenmiş basamak formu vardır ve Gauss-Jordan yöntemiyle bu forma ulaşılması garanti edilir. (Mon vd.,2014). Bu yöntem, Gauss eleme yönteminden daha fazla hesaplama gerektirir, ancak matrisin tersi hesaplanmadan çözüm sunma avantajına sahiptir. Gauss eliminasyon yönteminin bir çeşidi olan Gauss-Jordan eliminasyonu, fotovoltaik sistemlerde güç akışı ve arıza analizinde uygulanmıştır (Zhang vd., 2023). Bu uygulamalar, Gauss eliminasyon yönteminin çeşitli alanlardaki çok yönlülüğünü ve önemini vurgulamakta ve yaygın kullanımını göstermektedir.
- Basit Gauss Eleme Yöntemi: $[A][X] = [C]$ formundaki lineer denklem sisteminin çözümü için geliştirilmiştir. İleriye doğru eleme ve geriye doğru yerine koyma gibi iki aşaması vardır. Bilinen gauss eliminasyon yöntemini ifade eder (Strang, 2022).

- Seçimli Eleme Yöntemi: Seçimli eliminasyon yöntemi ise basit gauss eliminasyon yöntemine benzer fakat gauss eliminasyon yönteminin ilk adımı olan genişletilmiş katsayı matrisi oluşturulduktan sonra her adımda, pivot eleman olarak en büyük mutlak değeri olan elemanı seçilir. Kalan işlemler aynıdır (Gohberg, 1995).
- Ters Matris Yöntemi: Eğer matrisin tersi alınabiliyorsa, lineer denklem sistemlerinin çözümü matrisin tersi ile katsayı matrisinin çarpımı olarak bulunabilir. Bu yöntem, matrisin tersi mevcut olduğu durumlarda kullanılabilir ve hesaplama açısından oldukça doğrudur. Gauss-Jordan eleme yöntemi, bir matrisin tersini bulmak için kullanılan güçlü bir tekniktir. Matematik, bilgisayar bilimleri ve mühendislik dahil olmak üzere çeşitli alanlarda kapsamlı bir şekilde çalışılmış ve uygulanmıştır (Ji, 2012).
- Cramer Kuralı: Cramer kuralı, doğrusal cebirde, sistemin benzersiz bir çözümü olduğu sürece bir doğrusal denklem sistemini çözmek için bir yöntem sağlayan temel bir kavramdır (Strang, 2022). Determinantlar yardımıyla denklem sisteminin tek bir çözümünün olup olmadığını belirler ve çözümleri hesaplar. Cramer kuralı n değişkenli bir doğrusal denklem sistemi için, katsayı matrisinin determinantının sıfırdan farklı olması durumunda sistemin tek bir çözümü olduğunu ve bu çözümün ilgili matrislerin determinantları kullanılarak ifade edilebileceğini belirtir. Cramer kuralı, matrislerin determinantlarının hesaplanmasını içerdiğinden, hesaplama karmaşıklığı nedeniyle özellikle küçük denklem sistemleri için kullanışlıdır.

Doğrusal denklem sistemleri, mühendislik, fizik, ekonomi ve bilgisayar bilimleri gibi birçok alanda yaygın olarak kullanılır. Örneğin:

- Elektrik devre analizlerinde, akım ve gerilim değerlerinin hesaplanmasında.
- Mekanik sistemlerde, kuvvet ve moment dengelerinin belirlenmesinde.
- Ekonomik modellemelerde, arz-talep dengelerinin analizinde.
- Veri analizi ve istatistikte, regresyon analizlerinde.

Doğrusal denklem sistemlerinin çözümleri, bu sistemlerin doğru modellenmesi ve uygun çözüm yöntemlerinin seçilmesiyle etkin ve hızlı bir şekilde bulunabilir. Bu, çok değişkenli ve çok denklemli sistemlerin analiz ve çözümünde büyük bir avantaj sağlar.

1.7.1. Gauss Eliminasyon Yöntemi

Gauss eliminasyon yönteminin hesaplama yöntemi adım adım açıklanmıştır. Bu yöntemlerin her bir adımı yazılım ile kodlanmıştır

- Matrisin Hazırlanması: Verilen katsayılar matrisi A ve sonuçlar vektörü b alınır. Bu iki yapı kullanılarak artırılmış matris [A | b] oluşturulur. Artırılmış matris, katsayılar matrisi ve sonuçlar vektörünün yan yana konulmuş halidir (Stoer vd,1980).
- Üst Üçgensel Forma Dönüştürme: Algoritma, matrisin en üst satırından başlar ve her kolon için pivot olarak adlandırılan öğeyi seçer. Pivot, genellikle o sütundaki en büyük mutlak değere sahip elemandır. Pivot altındaki tüm elemanlar 0 olacak şekilde işlemler yapılır. Bunun için, pivot satırı, alt satırlardan çıkarılır. Bu işlem, alt satırların pivot elemanı 0 olana kadar tekrarlanır. Her sütun için bu işlem uygulandığında, matris üst üçgensel forma diğer bir deyişle tüm alt kısmı sıfırlar olan bir matrise dönüşür (Stoer vd,1980).
- Geriye Doğru Yerine Koyma: Üst üçgensel matris formunda, en alt satırdan başlayarak her bir satır için, bilinmeyenler geriye doğru çözülür. Her satır, sadece bir bilinmeyen içerecek şekilde düzenlenir ve bu bilinmeyen, diğer tüm bilinenler cinsinden ifade edilir. Bu işlem, matrisin en üst satırına kadar devam eder (Stoer vd.,1980).
- Sonuçların Hesaplanması: Algoritmanın bu son adımında, tüm bilinmeyenler sırasıyla hesaplanmış olur. Eğer matriste sıfırdan farklı bir satır tamamen sıfır ise, bu durum sistemin çözümünün olmadığını gösterir. Eğer matriste sıfır olan bir satır tamamen sıfır ise, bu durum sistemin sonsuz sayıda çözüme sahip olduğunu gösterir (Stoer vd.,1980).

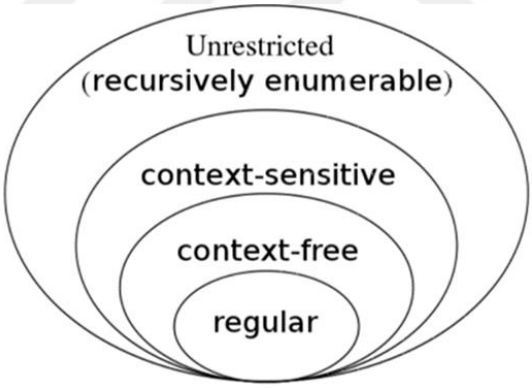
1.8. Biçimsel Gramerler

Biçimsel gramerler, dillerin sözdizimsel yapısını tanımlamak için kullanılan matematiksel sistemlerdir. Bu sistemler, bir dildeki tüm olası dizeleri oluşturmak için bir dizi kurallar sağlar ve bu kurallar, dilin kendisini tanımlayan temel yapı taşlarıdır. Biçimsel gramerler, dilbilim ve bilgisayar bilimi alanlarında, özellikle doğal dil işleme, programlama dillerinin tasarımı ve yapay zekâ gibi disiplinlerde geniş bir uygulama yelpazesine sahiptir.

Bu karmaşık süreç, gramerin derinlemesine anlaşılmasını ve çeşitli üretim yöntemlerinin kullanılmasını gerektirir. Gramerlerin bu denli önemli olmasının temel nedeni, dilin yapısal özelliklerini belirlemeleri ve bu yapıyı biçimsel olarak ifade etmeleridir. Örneğin, CFG (Bağlamdan Bağımsız Gramer), dilin sözdizimsel kurallarını belirlerken belirli bir bağlama ihtiyaç duymazlar ve bu sayede dilin yapısının daha genel ve esnek bir şekilde tanımlanmasını sağlarlar (Chomsky, 1959).

Biçimsel gramerlerin uygulanabilirliği, farklı dil yapılarının ve karmaşıklık seviyelerinin modellenmesine olanak tanır. Bu bağlamda, gramerlerin genelleştirilmesi, olasılıksal ve bulanık gramerler gibi çeşitli türlerin türetilmesine olanak tanır. Bu tür gramerler, dilin belirsizliklerini ve değişkenliklerini modelleyerek daha gerçekçi ve kullanışlı dil modelleri oluşturur (Mizumoto, Toyoda, & Tanaka, 1972).

Özellikle bilgisayar dilbilimi ve doğal dil işleme alanlarında, biçimsel gramerler, dilin yapı taşlarını belirlemek ve analiz etmek için kullanılır. Bu gramerler, bilgisayarların doğal dili anlamasını ve işlemesini sağlayarak, dil tabanlı uygulamalarda yüksek performans ve doğruluk sağlar (Johnson, 1994).



Şekil 4. Gramer hiyerarşisi (URL-18, 2024)

Şekil 4'te dil sınıflarının Chomsky hiyerarşisini göstermektedir. Chomsky hiyerarşisi, formal dillerin sınıflandırılmasında kullanılan bir modeldir ve dört ana dil sınıfını içerir. İç içe geçmiş bu dört ana sınıf, güç ve ifade yetenekleri açısından hiyerarşik bir yapı oluşturur.

1.8.1. Düzenli Gramer

Düzenli Gramer bilgisayar bilimlerinde genellikle düzenli gramer olarak bilinir ve otomat teorisi ve formal gramer teorisinin temel kavramlarından biridir (Yu, 1997).

Düzenli gramerler, basit kurallara dayanan ve sonlu otomatlar ya da düzenli ifadeler ile tanımlanabilen gramerdir. Bunlar, belirli bir gramerin yapı taşları olan karakterlerin belirli kurallar çerçevesinde birleştirilmesiyle oluşturulan karakter dizileri kümesidir. Örneğin, "ab*" düzenli ifadesi 'a' karakteri ile başlayan ve ardından sıfır veya daha fazla 'b' karakteri içeren tüm karakter dizilerini kapsar (örneğin, "a", "ab", "abb", "abbb" vs.). Düzenli diller, genellikle basit metin işleme, kod analizi ve bilgisayar bilimleri gibi alanlarda kullanılır.

1.8.2. Bağlamdan Bağımsız Gramer

Bilgisayar bilimlerinde önemli bir kavram olan CFG belirli kurallar kümesi ile tanımlanan gramer yapılarını ifade eder. Bu kurallar, bir gramerin sözdizimini oluşturan yapıların nasıl üretileceğini belirler ve dilbilgisi kuralları şeklinde ifade edilir.

CFG, biçimsel diller teorisinde bir gramer yapısıdır. Bu gramer türünün en önemli özelliği, üretim kurallarının herhangi bir sembol dizisinden oluşan bir sembol dizisi üretebilmesidir (Cremers,1975). Bu gramer türü, dil işleme, dönüşümü ve üretiminde sıklıkla kullanılır.

CFG, üretim kuralları veya çıkarım kuralları denilen yapılar kullanarak gramer üyelerini türetir. Bu kurallar genellikle "A --> B" şeklinde gösterilir. Burada "A", bir sonlu olmayan sembol ve "B", bu değişkenin yerine geçebilecek sembol veya semboller dizisidir.

Örnek olarak, basit bir CFG şu şekilde olabilir:

S ⟶ AB

A ⟶ a

B ⟶ b

Bu dilbilgisi, "ab" karakter dizisini üretir. "S" başlangıç sembolüdür ve CFG'deki kurallar kullanılarak A ve B'ye çözümlenir; A a' ya ve B b'ye dönüşür, böylece "ab" karakter dizisini oluşturur.

CFG'ler, daha karmaşık dillerin yapısını modellemek için kullanılır ve özellikle programlama dillerinin sözdizimini tanımlamak için yaygın olarak kullanılır. Ayrıca, bazı doğal dil işleme uygulamalarında da dilin sözdizimsel yapısını analiz etmek için CFG'lerden yararlanılır. CFG'lerin temel özelliği, bağlamdan bağımsız olmalarıdır; yani bir üretim kuralının uygulanabilirliği, kuralın uygulandığı çevresel sembollerden bağımsızdır.

1.8.3. Bağlama Duyarlı Gramer

CSG (Bağlama Duyarlı Gramer) teorisi ve formal dilbilgisi alanlarında kullanılan bir kavramdır ve Chomsky dil hiyerarşisinde CFG'lerin üzerinde yer alır (Hopcroft vd., 2001). CSG, dilbilgisi kurallarının uygulanabilmesi için belirli bir bağlamın gerektiği durumları tanımlamak için kullanılır. Bu dilbilgileri daha güçlüdür ve daha karmaşık dilleri tanımlayabilir, ancak bu güç artışı, genellikle daha büyük hesaplama maliyetiyle birlikte gelir.

1.8.4. Sınırsız Gramer

Sınırsız gramer, Chomsky dil hiyerarşisinde en genel ve kapsayıcı dil sınıfını temsil eder. Bu diller, sınırsız dilbilgisi veya Tip-0 dilbilgisi olarak da bilinen kurallarla tanımlanır. Sınırsız dilbilgileri, formal dil teorisinde en az kısıtlamaya sahip dilbilgileridir ve Turing makineleri tarafından tanınabilen dilleri ifade eder (Hopcroft vd., 2001). Bu, sınırsız dilbilgilerinin herhangi bir hesaplanabilir dilin temsil edilebileceği anlamına gelir.

1.9. Gramer Notasyonları

Gramer tanımlamak için kullanılan çeşitli notasyonlar vardır, bunlar genellikle dilbilimciler ve yazılım geliştiricileri tarafından kullanılır.

1.9.1. Backus-Naur Formu (BNF)

BNF (Backus-Naur Formu), bir dilin gramerini tanımlamak için kullanılan bir meta-dildir. Daha teknik olarak, BNF, bir dildeki tüm sentaks yapısını veren bir formalizasyondur (McCracken, 2003). John Backus ve Peter Naur tarafından geliştirilmiş olan bu form, bilgisayar bilimlerinde ve dilbilimde geniş bir kullanım alanına sahiptir. BNF, programlama dillerinin, veri biçimlerinin ve iletişim protokollerinin sözdizimini tanımlamak için yaygın olarak kullanılır. BNF, sözdizimsel analiz araçları ve derleyici tasarımı gibi alanlarda önemli bir rol oynar. Derleyiciler, BNF tanımlarını kullanarak kaynak kodunu analiz eder ve bu kodun geçerli olup olmadığını belirler. Bu analiz süreci, dilin kurallarına uygun olup olmadığını kontrol etmek için kritik öneme sahiptir.

BNF'nin temel bileşenleri şunlardır:

- Sonlu (Terminal) Semboller: Dilin en küçük yapı taşlarıdır ve daha fazla bölünemezler. Örneğin, anahtar kelimeler ve semboller.
- Sonlu olmayan (Non-terminal) Semboller: Diğer semboller aracılığıyla tanımlanan yapı taşlarıdır. Örneğin, ifadeler veya cümleler.
- Üretim Kuralları: Sonlu olmayan sembollerin, sonlu ve diğer sonlu olmayan semboller kullanılarak nasıl üretileceğini gösterir.

Tablo 1. BNF gramer tanımlama

```
<expression> ::= <term> | <expression> "+" <term> | <expression> "-
" <term>
<term> ::= <factor> | <term> "*" <factor> | <term> "/" <factor>
<factor> ::= <number> | "(" <expression> ")"
<number> ::= [0-9]+
```

Tablo 1'de belirtilen BNF gramer kuralları, aritmetik ifadelerin sözdizimini belirler. En üst düzeyde, bir <expression> (ifade) ya bir <term> (terim) ya da iki <term> arasında + veya - operatörleriyle oluşan bir yapı olabilir. Bir <term>, ya bir <factor> (faktör) ya da iki <factor> arasında * veya / operatörleriyle birleşen bir yapı olabilir. Bir <factor>, ya bir <number> (sayı) ya da parantez içinde bir <expression> olabilir. <number>, bir veya daha fazla [0-9] aralığındaki rakamları içeren bir dizidir. Bu kurallar, aritmetik ifadelerin doğru ve tutarlı bir şekilde yapılandırılmasını sağlar ve operatörlerin ve sayıların nasıl bir araya getirilmesi gerektiğini tanımlar.

1.9.2. Extended Backus-Naur Formu (EBNF)

EBNF (Extended Backus-Naur Form), BNF'nin genişletilmiş bir sürümüdür. EBNF, daha karmaşık dilbilgisi yapılarını tanımlamak için daha esnek bir yapı sunar. EBNF, BNF'ye ek olarak daha fazla sözdizim özelliği sunar (Aliyu, 2019). Örneğin tekrarlı yapıları, opsiyonel yapıları ve gruplamayı ifade etmek için daha fazla operatör içerir.

EBNF'nin temel özelliklerinden biri, üretim kurallarını daha ayrıntılı bir şekilde belirtmek için süslü parantezler, köşeli parantezler ve açıklama metinleri gibi ek sembollerin kullanılmasıdır. BNF, bilgisayar bilimcilerinin düzenli gramerleri tanımlamak

için kullandıkları bir notasyon tekniğidir. EBNF, BNF'yi genişleterek daha açıklayıcı ve güçlü bir gramer haline getirir.

EBNF'nin sunduğu ek özellikler arasında şunlar bulunur:

- Köşeli Parantezler ([]): Opsiyonel öğeleri belirtir.
- Küme Parantezler ({}): Sıfır veya daha fazla tekrar eden öğeleri belirtir.
- Parantezler (()): Gruplama yapmak için kullanılır.
- Alternatifler (|): Seçenekleri belirtir.

Tablo 2. EBNF gramer tanımlama

```
expression ::= term {("+" | "-") term}
term ::= factor {("*" | "/") factor}
factor ::= number | "(" expression ")"
number ::= digit {digit}
digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Tablo 2'de belirtilen gramer kuralları, aritmetik ifadelerin biçimlendirilmesini tanımlar. En üst düzeyde, bir expression (ifade) bir veya daha fazla term (terim) içerebilir ve bu terimler '+' veya '-'operatörleriyle ayrılabilir. Her term, bir veya daha fazla factor (faktör) içerebilir ve bu faktörler * veya / operatörleriyle ayrılabilir. Bir factor, bir number (sayı) veya parantez içinde bir expression olabilir. number bir veya daha fazla digit (basamak) içerir ve her digit 0'dan 9'a kadar bir rakam olabilir. Bu gramer kuralları, aritmetik ifadelerin yapı taşlarını ve bu yapı taşlarının birleşim kurallarını tanımlayarak, ifadelerin doğru ve tutarlı bir şekilde oluşturulmasını sağlar.

1.10. Ayrıştırıcılar

Simgesel hesaplama yapan bir araç geliştirirken ayrıştırıcı seçimi önemlidir. Çünkü seçilecek olan ayrıştırıcının işlevselliği, programın girdi verisinin sözdizimsel ve anlamsal olarak analiz edilmesini ve hesaplamaların yapılabilirliğini mümkün kılar. Sözdizim analizi için çeşitli araçlar geliştirilmiştir. Bunlar ANTLR (Parr & Quong, 1995), Lex/Yacc(Levine vd,1992), Bison(Levine vd, 2009), Flex(Levine vd, 2009), Ragel(Thurston,2003), Python Lex-Yacc (Beazley, 2003) ve Gold (Filannino & Di Bari, 2015) ve JavaCC(Java Compiler Compiler) (Kodaganallur, 2004)'dir. Bu araçlardan ayrıştırma işlemi yapacak olan araç ihtiyaca göre seçilebilir.

Ayrıştırıcı oluşturma araçları, biçimsel bir gramer tanımına dayanarak çalışır ve sözdizim analizi sonucunda AST (Soyut Sözdizim Ağacı) üretir. Bu ağaç, simgesel hesaplamalar için gerekli olan girdi verisini sağlar ve bu sayede simgesel hesaplama süreçlerinde kullanılan temel veri yapısını oluşturur. Bu sayede yeni bir gramer üretimi sağlanabilir.

Aşağıda çeşitli ayrıştırıcılar ve bu ayrıştırıcıların özelliklerini belirten bilgiler yer almaktadır.

JavaCC: JavaCC, Java programlama dilinden LL (Soldan Sağa Soldan Türetim) gramerleri kullanarak ayrıştırıcı üretmek için kullanılan bir araçtır. JavaCC kullanımı kolaydır ve güçlü bir ayrıştırıcı üretebilir. Ancak, JavaCC ile oluşturulan ayrıştırıcılar diğer ayrıştırıcılara kıyasla daha yavaş olabilir.

ANTLR**:** ANTLR, C++ ve Java programlama dillerinden ayrıştırıcı üretmek için kullanılan bir araçtır. ANTLR, JavaCC'ten daha güçlü ve esnek bir ayrıştırıcı üretebilir. Ancak, ANTLR kullanımı JavaCC'e kıyasla zordur.

Bison: Bison, C ve C++ programlama dillerinden ayrıştırıcısı üretmek için kullanılan bir araçtır. Bison, ANTLR'den daha hızlı ve daha verimli bir ayrıştırıcı üretebilir. Ancak, Bison kullanımı ANTLR'e kıyasla daha zordur.

1.10.1. Kelimesel Analiz

LL'de belirtilen ilk "L", girdi verisinin soldan sağa doğru bir şekilde birim sözcüklere ayrıştırılacağını ifade eder. Bu işlem, programlama dillerindeki kelimesel analiz tarafından gerçekleştirilir ve girdi metnini, dilin sözdizimsel yapılarına uygun anlamlı parçalar olan birim sözcüklere dönüştürme işlemidir. İkinci "L" ise, sözcüksel analiz sonucu elde edilen birim sözcük dizisinin, sözdizim kuralı çıkarımının yine soldan sağa doğru yapılacağını gösterir (Efendioğlu, 2017). Bu süreç genellikle ayrıştırıcı veya sözdizim analizcisi tarafından yürütülür ve birim sözcük dizilerini kullanarak dilin gramer kurallarına uygun ağaç yapıları oluşturur. Bu tür bir ayrıştırma, genellikle "yukarıdan aşağıya ayrıştırma" olarak bilinir ve belirli bir başlangıç sembolünden başlayarak girdi birim sözcüklerini kullanarak dilin üretim kurallarını takip eder. Bu metot, sözdizim ağacının kökünden yapraklarına doğru inşa edilmesini sağlar.

LR (Soldan Sağa Sağdan Türetim)'de geçen ilk "L", okunan girdi verisinin soldan sağa doğru birim sözcüklere ayrıştırılacağını ifade eder. Bu süreç genellikle sözcüksel

analiz tarafından gerçekleştirilir ve girdi verisi, dilin gramer yapısına uygun anlamlı birim sözcüklere dönüştürülür. "R" ise, sözcüksel analizden elde edilen birim sözcük dizisinden sözdizim kuralı türetiminin sağdan sola doğru yapılacağını gösterir (Efendioğlu, 2017). Bu, ayrıştırıcı veya sözdizim analizcisi tarafından yürütülen bir süreçtir ve "aşağıdan yukarıya ayrıştırma" olarak da bilinir. Bu yöntemde, birim sözcükler en düşük düzeydeki yapıdan başlayarak yukarı doğru, üst düzey yapıları oluşturacak şekilde birleştirilir. Sözdizim ağacı, yapraklardan köke doğru inşa edilir ve bu, girdi dizisinin sonundan başlayarak ilerler. Bu tür bir ayrıştırma yöntemi, genellikle daha karmaşık yapıların ve gramerlerin işlenmesinde kullanılır.

1.10.2. Sözdizim Analizi

Ayrıştırıcı tarafından ilk olarak işlenen ve metindeki temel sözcüksel birimleri temsil eden bir dilbilgisi birimidir. Bu birimler, kelimeler, noktalama işaretleri veya boşluklar gibi daha küçük anlamlı parçalardır. Ayrıştırıcı, metni ayrıştırırken birim sözcükleri tanır ve analiz eder ve bu da dilbilgisi kurallarına uygunluğunu belirlemesine ve metnin anlamını çıkarmasına olanak tanır.

Birim sözcükler, programlama dilleri ve doğal dil işleme gibi çeşitli alanlarda kullanılır. Programlama dillerinde, birim sözcükler genellikle anahtar kelimeler, tanımlayıcılar, sabitler ve operatörler gibi öğeleri temsil eder.

Birim sözcüklerin bazı önemli özellikleri şunlardır:

- Tip: Birim sözcüklerin değişik türleri vardır; kelime, noktalama işareti veya boşluk gibi.
- Değer: Bazı birim sözcüklerin bir değeri vardır, örneğin bir kelimenin metni veya bir sayının değeri.
- Konum: Her birim sözcüğün metinde bir konumu vardır.

Ayrıştırıcı, birim sözcükleri işlerken bunların türünü, değerini ve konumunu dikkate alır. Bu bilgiler, dilbilgisi kurallarına uygunluğunu belirlemek ve metnin anlamını çıkarmak için kullanılır. Birim sözcükler, dilbilgisi analizcileri ve diğer doğal dil işleme araçları için temel bir yapı taşıdır. Metnin yapısını ve anlamını analiz etmek için kullanılırlar ve dil ile ilgili çeşitli görevlerde önemli bir rol oynarlar.

Tablo 3. Birim sözcükleri içeren gramer kuralları

```
TOKEN: {
 < #DIGIT: ["0"-"9"] >
 < NUMBER: (<DIGIT>)+ >
 < #LETTER: ["a"-"z", "A"-"Z"] >
 < IDENTIFIER: (<LETTER>)+ >}
```

Tablo 3'te tanımlanan birim sözcükler şunları içerir:

- DIGIT: 0'dan 9'a kadar olan tek bir rakam.
- NUMBER: Bir veya daha fazla rakam içeren bir dizi.
- LETTER: Küçük veya büyük harflerden oluşan tek bir harf.
- IDENTIFIER: Bir veya daha fazla harf içeren bir dizi.

Her bir birim sözcük türü için uygun olan girişleri belirlemek, dilin kurallarına bağlıdır. Tablo 3'te yer alan birim sözcükleri JavaCC dosyasının içerisinde yer alan ve dili ayrıştırmamıza yardımcı olan fonksiyonlar sayesinde analiz edilecek dilin karmaşıklığı ve işlevselliği artırılabilir.

1.10.3. LL(k) Gramerleri: First Set ve Follow Set

Derleyici tasarımında kullanılan kritik bileşenlerden olan "First Set" ve "Follow Set" kavramları, sözdizimsel analiz aşamasında büyük önem taşır. Bu kavramlar, belirli bir dilin gramerini analiz ederken ve ayrıştırıcıları tasarlarken kullanılır. Aşağıda bu iki temel kavram detaylı bir şekilde açıklanmaktadır:

Bir dilin gramerindeki her sonlu olmayan simge için, First Set, bu simgeden türetilen çıktı dizilerinin başlangıç sonlu simgelerini içerir. Bu set, belirli bir sonlu olmayan sembollerin türetilebileceği ilk sonlu simgelerin belirlenmesinde kullanılır. Matematiksel olarak, bir sonlu olmayan X için First Set, X'ten başlayarak türetilen sonlu dizilerinin ilk simgelerini içerir. Eğer X boş bir üretim epsilon üretebiliyorsa, epsilonda First Set içinde yer alır.

Örnek gramer:

$A \rightarrow aB$A→aB

$B \rightarrow b$| Є B→b| Є

Bir dilin gramerinde her sonlu olmayan simge için, Follow Set, o sonlu olmayan simgenin gramer kuralları içindeki pozisyonlarından sonra gelen sonlu simgeleri içerir. Bu

set, özellikle LL ve LR gibi ayrıştırma yöntemlerinde, hangi ayrıştırma kuralının uygulanacağını belirlemede kullanılır.

Matematiksel olarak, bir sonlu olmayan X için Follow Set, X'in gramer kurallarındaki herhangi bir pozisyonundan sonra doğrudan gelen sonlu simgelerini içerir. Eğer X gramerin sonunda yer alıyorsa ve ardından başka bir simge gelmiyorsa, bu durumda girdi dizisinin sonunu belirten özel bir simge genellikle $ işareti Follow Set'e dahil edilir.

First ve Follow Set'in Kullanım Alanları:

- LL (1) Ayrıştırıcı: LL (1) ayrıştırıcılar, kaynak kodu soldan sağa ve tek bir sembole bakarak ayrıştırır. First Set ve Follow Set'ler, ayrıştırıcının bir sonraki sembolü tahmin ederek ayrıştırma işlemini kararlı bir şekilde yürütmesine yardımcı olur.
- Semantik Analiz**:** Semantik analizde, kaynak kodun anlamı incelenir. First ve Follow Set'ler, bu analiz sırasında değişkenlerin kullanımı ve değerlerinin atanması gibi işlemlerde kullanılır.

Tablo 4. First Set ve Follow Set üretim kuralları

```
A -> BA'
A' -> +B A'|Є
B -> C B'
B' -> *C B' | Є
C -> (A) | id

FIRST set

FIRST(A) = FIRST(B) = { ( , id }
FIRST(A') = { +, Є }FIRST(B) = FIRST(C) = { ( , id }
FIRST(B') = { *, Є }
FIRST(C) = { ( , id }

FOLLOW Set
FOLLOW(A) = { $ , ) } ')'
FOLLOW(A') = FOLLOW(A) = { $, ) }
FOLLOW(B) = { FIRST(A') – Є } U FOLLOW(A') U FOLLOW(A) = { + , $ , ) }
FOLLOW(B') = FOLLOW(B) = { + , $ , ) }
FOLLOW(C) = { FIRST(B') – Є } U FOLLOW(B') U FOLLOW(B) = { *, +, $, )
```

Tablo 4'te yer alan setlerin hesaplanması, ayrıştırıcıların sözdizim kurallarını doğru bir şekilde uygulamasını sağlar ve derleyicilerin dil yapılarını daha etkin bir şekilde

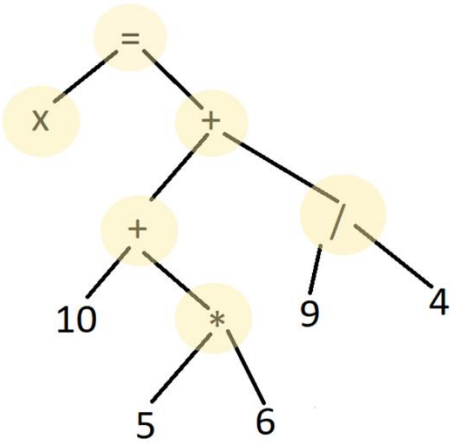
işlemesine olanak tanır. Özellikle karmaşık programlama dillerinin ayrıştırılmasında, First Set ve Follow Set'in doğru hesaplanması, dil işleme sürecinin temel taşlarından biridir.

1.10.4. Soyut Sözdizim Ağacı

AST dilin sözdizimsel yapısını temsil eden ve semantik açıdan önemli bilgileri içeren bir soyut ağaç yapısıdır. Bu ağaç, programın temel bileşenlerini (ifadeler, deyimler, işleçler vb.) içeren bir hiyerarşi oluşturur ve her düğümde belirli bir dil öğesini (ifade, deyim, işleç vb.) temsil eder. AST, programın içerdiği operatörlerin ve bu işlemlerin birbiriyle etkileşimde bulunduğu öğelerin (operand) ilişkilerini gösterir. Önemli bir özelliği, dil bağımsız bir yapıya sahip olmasıdır; yani, aynı AST yapısı farklı bir programlama dilinde yazılmış kodun soyut yapısını da doğru bir şekilde yansıtabilir.

Bu veri yapısı genellikle derleyiciler ve yorumlayıcılar tarafından kullanılır ve kaynak kodun analiz aşamasında oluşturulur. AST, programın anlaşılması, optimize edilmesi ve yürütülmesi gibi süreçlerde önemli bir rol oynar. Ayrıca, yazılım geliştirme araçlarında, kod analizi, dönüştürme ve yeniden kod düzenleme gibi işlemlerde de etkili bir şekilde kullanılabilir.

Sözdizimi ağacı ise kaynak verilerin ağaç veri yapısı şeklinde temsil edildiği bir yapıdır. Sözdizim sınıflarından türetilen nesnelerin birbirine bağlanmasıyla oluşturulan bu ağaç, sözdizimindeki öğeleri doğru bir şekilde temsil ederek ayrı bir hiyerarşi oluşturur. Ağacın ara düğümleri, işlem bileşenlerini, yaprakları ise veri bileşenlerini temsil eder. Bu sayede, her düğüm ilgili kod bileşeninin türüne bağlı olarak farklı bir sözdizimi sınıfından bir nesneyi içerebilir. Sözdizimi ağaçları, kaynak verilerin tip kontrolü ve kod yorumlama gibi önemli işlemlerini gerçekleştirmek için kullanılır. Bu yapılar, genellikle sözdizimi analizi sırasında kullanılan JavaCC gibi araçlarla üretilebilir. Ayrıştırıcı metotlarına özel kod blokları eklenerek, ağacın ilgili düğüm verilerini üretecek Java ifadeleri düzenlenir ve böylece sözdizimi ağacı oluşturulur. Bu ağaçlar, yazılım geliştirme süreçlerinde kodun anlamlandırılması, dönüştürülmesi ve optimize edilmesi gibi birçok önemli görevde etkin bir şekilde kullanılır.



Şekil 5. AST’nin yapısı

Örneğin int x = 10 + 5 * 6 + 9 / 4; ifadesinin AST'sini oluşturmaya dair basit bir örnek Şekil 5’te yer almaktadır. Bir matematiksel ifadenin AST’de nasıl temsil edildiği göstermektedir. Bu ağaçta, operatörler belirli sınıflar ile temsil edilmektedir; toplama işlemleri için Plus, çıkarma işlemleri için Minus, çarpma işlemleri için Times ve bölme işlemleri için Divide sınıfları bulunmaktadır. Her bir operatör sınıfı Exp adı verilen soyut bir sınıftan türetilmiştir. Bu yapı, farklı matematiksel işlemleri temsil etmek ve işlemek için kullanılan nesne yönelimli programlama prensiplerine dayanmaktadır.

1.10.5. Soyut Sözdizim Sınıfları

ASC (Soyut Sözdizim Sınıfı), derleyiciler ve programlama dili tasarımı bağlamında temel bir kavramdır. Bu sınıflar, kaynak verinin bilgisayar programları tarafından daha kolay işlenebilmesini sağlayacak temsili bir yapısal verinin oluşturulabilmesine imkân verir.

Soyut sözdizimi sınıfları, AST'nin yapı taşlarını oluşturur. Her bir sınıf, dildeki belirli bir yapı türünü temsil eder. Kaynak bileşenler için de Assignment ve Expression gibi soyut sözdizimi sınıfları tanımlanmıştır. Bu sınıflar, her bir yapının temel özelliklerini ve davranışlarını tanımlar. Soyut sözdizimi sınıfları genellikle kalıtım ilkesini kullanarak hiyerarşik bir yapı oluşturur. Bu sayede, ilgili yapıların ortak özelliklerini tek bir üst sınıfta tanımlamak ve alt sınıflarda bu özellikleri özelleştirmek mümkün olur. Örneğin, Divide, Times ve Plus gibi farklı deyim türleri, genel bir işlem sınıfından türeyebilir. Bu sayede, her iki deyim türü de ortak özelliklere sahip olurken, kendi türlerine özgü davranışları da tanımlayabilir.

Soyut sözdizimi sınıflarının temel amacı, derleyiciler ve yorumlayıcılar tarafından kaynak kodunu analiz etmeyi ve işlemeyi kolaylaştırmaktır. Örneğin, bir derleyici ayrıştırılmış koddan AST'yi oluşturmak için soyut sözdizimi sınıflarını kullanır. Daha sonra, AST semantik analiz, optimizasyon ve kod üretimi gibi aşamalarda kullanılır. Uygulamada, soyut sözdizimi sınıfları genellikle yüksek seviyeli bir programlama dilinde tanımlanır ve derleyicinin çeşitli bileşenleri tarafından kullanılır. Ayrıştırıcı, kaynak kodunu ayrıştırarak AST'yi oluşturur. Daha sonra, AST'yi işleyen diğer bileşenler (örneğin, semantik analizci, kod optimizasyonu ve kod üreticisi) bu sınıfların sunduğu arayüzleri ve işlevleri kullanarak görevlerini yerine getirir. Ayrıca dilin sözdizimi kurallarına uymayan kodları tespit edip hataları raporlayarak programcılara hata düzeltme ve kod yazma konusunda yol gösterir. Soyut sözdizimi sınıfı, dilin gramer kurallarını içeren ve programcıların dilin doğru kullanımını sağlamada önemli rol oynayan bir yapıdır (Pehlivan, 2018).

Sonuç olarak soyut sözdizimi sınıfları, derleyicilerin ve yorumlayıcıların temel bir yapı taşıdır. Bu sınıflar, kaynak kodun ayrıntılarını daha kolay analiz ve manipülasyon için yapılandırılmış bir formata soyutlamada yardımcı olur. Bu sayede, derleyiciler ve yorumlayıcılar kaynak kodu daha verimli ve hatasız bir şekilde işleyebilir.

Tablo 5. Soyut sözdizim sınıfları

class Exp{}
class num_Exp extends Exp{ int num;
public num_Exp(int a1){ num=a1;
}
public String toString(){
return "num_Exp("+ num + ")"; } }
	class plus_Exp extends Exp{
	public plus_Exp(){}
	public String toString(){return "plus_Exp";
	} }class minus_Exp extends Exp{ public
	minus_Exp(){ }
	public String toString(){
	return "minus_Exp()";}}

Tablo 5'te yer alan ASC kullanıma ait örnek tanımlamalar gösterilmiştir. ASC programlama dilindeki ifadelerin ve komutların sınıflar halinde tanımlanması ve bu sınıfların türetilmesi için kullanılır. Bu yapı dilin sözdizimsel yapısını tanımlamak ve sınıflar aracılığıyla bu yapının öğelerini modellemek için kullanılır.

1.10.6. Ziyaretçi Tasarım Deseni

Günümüzde tasarım kalıpları, yazılım geliştirme ve uygulamada temel bir rol oynamakta ve tekrar eden problemler için geniş bir yelpazede tasarım çözümleri

sunmaktadır (Hussein & Mehanna, 2020). Tasarım kalıpları, yazılım geliştirmede sıkça karşılaşılan sorunları çözmek için yaygın olarak kullanılan tekrarlanabilir çözümler sunar. Bu kalıplar, yazılımın daha modüler, esnek ve bakımı kolay hale gelmesini sağlar. Üç ana kategoriye ayrılır: yaratıcı, yapısal ve davranışsal tasarım kalıpları. Yaratıcı kalıplar nesne oluşturma mekanizmalarını iyileştirir, yapısal kalıplar nesneler ve sınıflar arasındaki ilişkileri tanımlar ve davranışsal kalıplar nesneler arasındaki iletişim ve sorumlulukları belirler.

Davranışsal kalıp olan Ziyaretçi (Visitor) tasarım deseni, bir nesne hiyerarşisindeki çeşitli nesne türleri üzerinde farklı işlemleri gerçekleştirmek için kullanılan bir davranışsal tasarım desenidir. Bu desende, her bir nesne türü için ayrı bir ziyaretçi sınıfı tanımlanır ve her ziyaretçi sınıfı, her nesne türü için bir ziyaret yöntemi içerir. Bu yöntemler, ziyaret edilen nesne üzerinde istenilen işlemi gerçekleştirir. Ziyaretçi tasarım deseninin özellikleri aşağıda açıklanmıştır.

- Kod Tekrarını Azaltır: Ziyaretçi deseni, kod tekrarını azaltmak için idealdir. Birden fazla işlem için aynı kodu tekrar yazmak yerine, her işlem için ayrı bir ziyaretçi sınıfı tanımlayabilirsiniz. Bu sayede, kodunuz daha kolay okunabilir ve yönetilebilir hale gelir.
- Esneklik Sağlar: Ziyaretçi deseni, yeni işlemler eklemeyi kolaylaştırır. Mevcut bir ziyaretçi sınıfını değiştirmenize veya yeni bir ziyaretçi sınıfı oluşturmanıza gerek kalmadan yeni işlemler ekleyebilirsiniz.
- AST oluşturulduktan sonra, AST üzerindeki işlemler ziyaretçi tasarım deseni kullanılarak gerçekleştirilir. Bu desen, AST'nin her düğümünde farklı işlemler yapmayı kolaylaştırır.

Ziyaretçi tasarım deseni, AST düğümleri üzerinde yapılan işlemleri tanımlamak ve uygulamak için kullanılır. AST düğümleri genellikle çeşitli türlerde olabilir (örneğin, ifadeler, deyimler, fonksiyon tanımları), ve her düğüm türü için farklı işlemler gerekebilir. Ziyaretçi deseni, bu işlemleri düğüm sınıflarından soyutlayarak, kodun esnekliğini ve modülerliğini artırır.

Ziyaretçi tasarım deseni, yazılım programlarında, özellikle ayrıştırıcı sonrası işlemler için büyük faydalar sağlar. Ayrıştırıcı tarafından gerçekleştirilen ayrıştırma işlemi bittikten sonra, elde edilen veri yapısı üzerinde çeşitli işlemler gerçekleştirmek için ideal bir araçtır. Ziyaretçi deseninin kullanımı, ayrıştırıcı sınıfının temel yapısına dokunmadan esnek ve çok

yönlü işlem seçenekleri sunar. Bu sayede, mevcut kodun üzerine yeni işlevler kolayca eklenir ve mevcut yapının genişletilmesi sırasında karmaşıklık azalır.

Ziyaretçi tasarım deseni özellikle AST sınıf yapısına sahip sistemler için başarılı bir şekilde kullanılır. Örneğin yazılan bir gramer üzerinde anlamsal analiz, kod analizi ve sonuç üretimi yapabilmek farklı Ziyaretçi sınıflar tanımlanabilmektedir (Gamma vd,1995). Bu tip faydalar, yazılım geliştirme sürecini daha verimli, nesnel ve yönetilebilir hale getirir. Bu yaklaşım, yazılımın esnekliğini ve genişletilebilirliğini artırırken, aynı zamanda kodun bakımını kolaylaştırır, böylece geliştirme süreci daha verimli ve etkili hale gelir.

Tablo 6. Ziyaretçi tasarım deseni

```
// Ziyaretçi arayüzü tanımlama
 interface ExpVisitor { String visitNumExp(num_Exp exp);
 String visitPlusExp(plus_Exp exp);
 String visitMinusExp(minus_Exp exp);} }
// Temel Exp sınıfı
 abstract class Exp { public abstract String accept(ExpVisitor visitor);}//
 class num_Exp extends Exp { num_Exp sınıfı
 int num;
 public num_Exp(int a1) { num = a1; }

 @Override
 public String accept(ExpVisitor visitor) {
 return visitor.visitNumExp(this); }
public String toString() { return "num_Exp(" + num + ")"; }}
 public String toString() { return "plus_Exp"; }}

// minus_Exp sınıfı
class minus_Exp extends Exp {
 public minus_Exp() { } @Override
 public String accept(ExpVisitor visitor) {
 return visitor.visitMinusExp(this); }
public String toString() {
 return "minus_Exp()"; }}// Bir ziyaretçi uygulaması
class ConcreteVisitor implements ExpVisitor { @Override
 public String visitNumExp(num_Exp exp) {
 return exp.toString(); }
@Override
 public String visitPlusExp(plus_Exp exp)
{ return exp.toString(); }
 @Override
 public String visitMinusExp(minus_Exp exp) { return exp.toString();
}}
```

Tablo 6'da gösterilen kod, AST ve ziyaretçi desenini kullanarak matematiksel ifadelerin nasıl işleneceğini tanımlamaktadır. ExpVisitor arayüzü, ziyaretçi deseninin

temelini oluşturur ve num_Exp, plus_Exp ve minus_Exp gibi farklı ifade türlerini ziyaret eden metotları içerir. Exp sınıfı, tüm ifadelerin türetildiği soyut bir sınıftır ve her ifade sınıfının, belirli bir ziyaretçiyi kabul etmesi gerektiğini belirtir. Bu yapı, ifadelere eklenen yeni işlemlerin kolayca genişletilmesini sağlar, çünkü yeni bir ziyaretçi eklemek sadece ExpVisitor arayüzüne ve ziyaretçi sınıfına yeni metotlar eklemeyi gerektirir.

Kodun devamında, num_Exp, plus_Exp ve minus_Exp sınıfları, Exp sınıfından türetilmiş ve her biri belirli bir matematiksel ifadeyi temsil etmektedir. num_Exp sınıfı, bir sayısal ifadeyi temsil ederken, plus_Exp ve minus_Exp sınıfları sırasıyla toplama ve çıkarma işlemlerini temsil eder. Bu sınıflar, kendi kabul (accept) metotlarını uygulayarak, kendilerini ziyaret eden ziyaretçiyi kabul ederler. ConcreteVisitor sınıfı, ExpVisitor arayüzünü uygular ve her bir ifade türünü işlemek için visitNumExp, visitPlusExp ve visitMinusExp metotlarını içerir. Bu ziyaretçi deseni, ifadelerin türüne bağlı olarak farklı işlemlerin gerçekleştirilmesini sağlar ve kodun genişletilebilirliğini artırır. Bu yapı, özellikle sayısal türev hesaplamalarında kullanılan matematiksel ifadelerin işlenmesi ve değerlendirilmesi açısından esneklik ve modülerlik sağlar.

1.11. Javada Matematik Kütüphaneleri

Java’da birçok matematiksel işlem yapabilen kütüphane aracı mevcuttur. Bu araçlar her biri çeşitli zorlukların üstesinden gelebilmek için üretilmiştir. Java’yı kullanan mühendisler, yazılımcılar ve programcılar çalışmalarının bazı yerlerinde belirli matematiksel problemleri hızlı bir şekilde ele alan ve çözüm üretebilen böyle araçlara ihtiyaç duymaktadır. Java’da kod yazan birçok kullanıcı kendi ihtiyaçları doğrultusunda bu araçları projelerinde kullanmışlardır.

Java’da yer alan kütüphanelerin her birinin amacı farklılık göstermektedir. Java kütüphanelerinin bu farklılıkları aşağıda ifade edilmiştir.

- Apache Commons Math: Commons Math kütüphanesi, Java programlama dilinde veya Commons Lang'de olmayan yaygın sorunları ele alan, matematik ve istatistik bileşenlerinden oluşan bir kütüphanedir (URL-12, 2024). Ayrıca rastgele sayı üretimi ve olasılık dağılımları ve sayısal çözümleme ile ilgili yardımcı araçtır.
- JScience: Fizik, matematik ve astronomi alanlarında ihtiyaç olan matematiksel fonksiyonları içeren kütüphanedir (URL-13, 2024). Birçok vektör ve matris yapıları barındırır. Uygulamaların çoğunluğundan mühendisler tarafından tercih edilirler.

- Colt: Yüksek seviyede lineer cebir işlemleri içeren problemlerin çözümünde kullanılırlar. Ayrıca istatiksel veri analizi uygulamalarında bu kütüphane tercih edilebilir (URL-14, 2024).
- Algib: Lineer denklem sistemlerinin çözümünde sayısal yaklaşım hesaplamalarında tek boyutlu veya çok boyutlu sayısal optimizasyonda kullanılır. Ayrıca sayısal türev, sayısal integral hesaplamada ve diferansiyel denklemleri çözebilen fonksiyonları içermektedir (URL-15, 2024).
- ND4J: Derin öğrenme problemlerinin ele alınması sürecinde gerekli olan matematik fonksiyonları içeren araçtır. Lineer cebir, matris işlemleri ve çok boyutlu dizilerde hesaplama yapabilen fonksiyonları barındırmaktadır (URL-16, 2024).
- JAMA: Özvektör ve özdeğer hesaplayabilen, matris oluşturabilen, matrisi manipüle edebilen, matrisin tersini alabilen ve matris çarpımı yapabilen araçtır. Lineer cebir problemleri için kolaylık sağlamaktadır (URL-17, 2024).

2. YAPILAN ÇALIŞMALAR

Bu çalışmada simgesel hesaplama yaklaşımları yardımıyla sayısal türev problemlerine yönelik bir yazılım uygulamasının geliştirilmesine odaklanılmıştır. Bu yeni yöntem, bağlamdan bağımsız dilbilgisi yapısı ile tasarlanmış ve sayısal türev ifadelerinin üretimini mümkün kılmakta olup, aşağıdaki özellikleri barındırmaktadır.

• Yüksek Sayısal Hassasiyet: Sayısal hassasiyete özen göstermeli ve hataları en aza indirmelidir.

• Verimlilik: Sayısal türev problemlerini çözmek için optimize edilmelidir. Bu amaçla, verimli sayısal algoritmalar ve veri yapıları kullanılmalıdır.

• Geniş İşlevsellik: Sayısal türev problemlerinde ihtiyaç duyulan tüm noktasal yönlü işlevleri sunmalıdır.

• Kullanıcı Dostluğu: Kullanıcı dostu bir arayüze ve ayrıntılı bir dokümantasyona sahip olmalıdır.

Uygulamanın girdi verisinin yapısı seçilecek ayrıştırıcı türünü etkileyen önemli bir faktördür. Geniş ve karmaşık sözdizim bileşenlerine sahip kaynak veri, daha esnek ve güçlü ayrıştırıcı mekanizmalarının kullanımını gerektirir. Bu bağlamda, otomatik araçlar kullanarak CFG'ye benzer notasyonlarla tanımlanabilen ayrıştırıcılar daha yüksek çalışma performansı gösterebilmektedir. Örneğin, JavaCC, EBNF notasyonuna dayalı bir ayrıştırıcı üretecidir. EBNF gramerleri kullanılarak tanımlanan sözdizim bileşenlerini Java koduna dönüştürerek ayrıştırma metotlarını düzenler. Özellikle, EBNF gramerlerinin doğrudan Java koduna dönüştürülmesi, ayrıştırma işleminin hızlı ve etkili bir şekilde gerçekleştirilmesini sağlar.

2.1. Sayısal Türevin Programlanması

Matematik ve bilgisayar bilimlerinde, analitik yöntemlerle bir fonksiyonun türevini hesaplamak oldukça karmaşık ve zaman alıcı olabilmektedir. Özellikle fonksiyonların terim sayıları ve dereceleri, analitik yöntemlerle yapılacak türev hesaplamasının maliyetini artırabilmektedir. Fonksiyonun sadece belirli noktalardaki değerlerinin bilindiği durumlarda türev hesaplamaları daha da karmaşık hale gelebilmektedir. Bu tür durumlarda, standart türev alma yöntemleri pratik veya yeterince doğru olmayabilir.

Java, geniş kullanım alanı olan bir programlama dili olmasına rağmen, sayısal türev hesaplamaları için yeterli düzeyde destek sunmamaktadır. Bu durum, türev hesaplamaları yapmak isteyen geliştiriciler ve araştırmacılar için önemli bir engel teşkil etmektedir. Mevcut matematiksel kütüphanelerin çoğu, noktalar kümesine bağlı olarak türev hesaplaması yapma yeteneğine sahip değildir. Türev hesaplaması yapabilen kütüphaneler ise, bu hesaplamaların nasıl gerçekleştirildiğine dair yeterli açıklama ve kılavuz sunmamaktadır. Bu eksiklikler, Java çevrelerinde sayısal türev programlamalarına destek verecek yeni araçlara olan gereksinimi açıkça ortaya koymaktadır.

Bu bağlamda, Java'da sayısal türev hesaplamaları yapabilen, kolay kullanılabilir ve adımları iyi bir şekilde raporlayabilen bir kütüphane veya araç geliştirilmesi, araştırmacılar ve geliştiriciler için önemli bir katkı sağlayacaktır. Bu araç, sonlu fark yöntemlerini kullanarak türev hesaplamalarını gerçekleştirebilir ve nokta sayısına bağlı olarak doğruluğu artırma olanağına sahip olabilir. Böylece, analitik türev almanın zorluklarından kaçınılarak, pratik ve doğru sonuçlar elde edilebilir.

Uygulama girdi verisinin sözdizimi basit ve anlaşılır olup, aynı zamanda esnek yapıları destekleyecek şekilde genişletilebilir. Bu sayede, mühendisler, araştırmacılar ve öğrenciler, sayısal türev hesaplamalarının yöntemleri hakkında derinlemesine bilgi edinebilir. Ayrıca, uygulama sonucu elde edilen türev değerlerini kendi çalışmalarında kullanma imkânına sahip olacaklardır. Tezde, EBNF notasyonu kullanılarak bir biçimsel gramerin tanımlanması ile AST veri yapısının üretimine kadar tüm programlama sürecinin işlem adımları açıklanmıştır. AST üzerinden yapılacak hesaplamalarda Ziyaretçi (Visitor) tasarım deseni kullanılmıştır. Hesaplama aşamalarının belgelenmesi ve görselleştirilmesi için LaTeX metin formatlayıcısı kullanılarak kapsamlı bir belgelendirme yapılmıştır. Bu şekilde hem teorik hem de uygulamalı olarak tüm hesaplama adımlarının matematiksel bir betimlemesi sağlanmıştır.

2.2. Sayısal Türev Formülleri

Tablo 7'de formül sütununda yer alan $y[d,b,f]$ ifadesi, sayısal türev hesaplamasının girdi parametreleri temsil edilmiştir.; burada d, türev derecesini, b, geriden alınacak nokta sayısını ve f, ileriye doğru alınacak nokta sayısını ifade etmektedir.

Tablo 7. 3 noktalı türev formülleri

Türev Türü	**Formül $y\,[d, b, f]$**	**Açıklama**
1. Türev	$y[1,2,0] = \frac{y_2 - 4y_1 + 3y_0}{2h} + O(h^2)$	İleri fark formülü: İleri yönlü 2 nokta
1. Türev	$y[1,1,1] = \frac{-y_{-1} + y_1}{2h} + O(h^2)$	Merkez fark formülü: İleri ve geri yönlü 1'er nokta
1. Türev	$y[1,0,2] = \frac{-3y_0 + 4y_1 - y_2}{2h} + O(h^2)$	Geri fark formülü: Geriden 2 nokta
2. Türev	$y[2,2,0] = \frac{y_2 - 2y_1 + y_0}{h^2} + O(h)$	İleri fark formülü: İleriden 2 nokta
2. Türev	$y[2,1,1] = \frac{y_{-1} - 2y_0 + y_1}{h^2} + O(h)$	Merkez fark formülü: Geriden ve ileriden 1'er nokta.
2. Türev	$y[2,0,2] = \frac{y_0 - 2y_1 + y_2}{h^2} + O(h)$	Geri fark formülü: Geriden 2 nokta
3. Türev	y [3,0,3]= Formül yok	3 nokta kullanarak üçüncü türev hesaplanamaz.

Tablo 7'de 3 noktalı örnek türev formülleri listelenmiştir.

Tablo 8. 4 noktalı türev formülleri

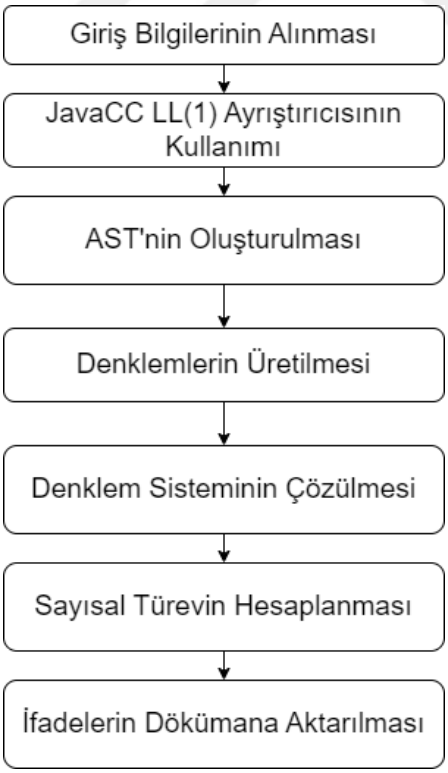
Türev Türü	Formül $y\,[d, b, f]$	Açıklama
1. Türev	$y[1,3,0] = \frac{-2y_3 + 9y_2 - 18y_1 + 11y_0}{6h} + O(h^3)$	İleri fark formülü: İleriden 3 nokta
1. Türev	$y[1,2,1] = \frac{y_2 - 6y_1 + 3y_0 + 2y_1}{6h} + O(h^3)$	Merkez fark formülü: Geriden 2 nokta ve ileriden 1 nokta
1. Türev	$y[1,1,2] = \frac{-2y_1 - 3y_0 + 6y_1 - y_2}{6h} + O(h^3)$	Merkez fark formülü: Geriden 1 nokta ve ileriden 2 nokta
1. Türev	$y[1,0,3] = \frac{-11y_0 + 18y_1 - 9y_2 + 2y_3}{6h} + O(h^3)$	Geri fark formülü: Geriden 3 nokta
2. Türev	$y[2,3,0] = \frac{-y_3 + 4y_2 - 5y_1 + 2y_0}{h^2} + O(h^2)$	İleri fark formülü: İleriden 3 nokta
2. Türev	$y[2,2,1] = \frac{y_1 - 2y_0 + y_1}{h^2} + O(h^2)$	Merkez fark formülü: Geriden 1 nokta ve ileriden 2 nokta
2. Türev	y [2,1,2] = formül yok	Merkezi fark formülü: Hesaplanamadı.
2. Türev	$y[2,0,3] = \frac{2y_0 - 5y_1 + 4y_2 - y_3}{h^2} + O(h^2)$	Geri fark formülü: Geriden 3 nokta
3. Türev	$y[3,3,0] = \frac{-y_3 + 3y_2 - 3y_1 + y_0}{h^3} + O(h)$	İleri fark formülü: İleriden 3 nokta
3. Türev	$y[3,2,1] = \frac{-y_2 + 3y_1 - 3y_0 + y_1}{h^3} + O(h)$	Merkez fark formülü: Geriden 2 nokta ve ileriden 1 nokta
3. Türev	$y[3,1,2] = \frac{-y_1 + 3y_0 - 3y_1 + y_2}{h^3} + O(h)$	Merkez fark: Geriden 1 nokta ve ileriden 2 nokta
3. Türev	$y[3,0,3] = \frac{-y_0 + 3y_1 - 3y_2 + y_3}{h^3} + O(h)$	Geri fark formülü: 3 nokta geriye

Tablo 8'de belirtilen örnek noktalara göre hesaplama yapılabilmektedir. Örneğin, b değeri 2 ise, geriden alınacak noktalar -2 ve -1 olacaktır. Benzer şekilde, f değeri 3 ise,

ileriye doğru alınacak noktalar 1, 2 ve 3 olacaktır. Bu durumda, y [1, 2, 3] formülü, geriden -2 ve -1, ileriden ise 1, 2 ve 3 noktalarını kullanarak birinci türev hesaplaması yapılacağını belirtmektedir. Yani, bu formülde kullanılan noktalar -2, -1, 1, 2 ve 3 olup, bu noktalar kullanılarak merkezi fark yöntemi ile 5. türeve kadar hesaplamalar yapılabilir. Eğer b değeri sıfır ve f değeri 2 veya daha büyük pozitif tam sayı ise, bu formül ileri yönlü farkları temsil etmektedir. Benzer şekilde, b değeri 2 veya daha büyük bir tam sayı ise, bu durum geri farklar formülünü temsil etmektedir. b ve f değerlerinin en az 2 veya daha büyük olmasının nedeni, türev hesabı yapabilmek için en az iki noktanın gerekli olmasıdır.

2.3. Uygulamanın Akış Şeması

Şekil 6'da yer alan akış şeması, sayısal türev uygulamasının genel adımlarını özetlemektedir. İşlem, gerekli veri noktalarının toplanmasıyla başlar. Bu veri noktaları, sonraki hesaplamalarda kullanılacaktır ve çeşitli kaynaklardan elde edilebilir; örneğin deneysel ölçümler, önceden tanımlanmış veri setleri veya kullanıcı girdileri. Giriş verileri toplandıktan sonra, JavaCC ile oluşturulan bir LL (1) ayrıştırıcı kullanılır.



Şekil 6. İşlem basamakları

Giriş verileri ayrıştırıldıktan sonra, AST oluşturulur ve formül hesaplama için gereken denklemler üretilir. Bu denklemler, daha sonra çözülmek üzere bir denklem sistemine dönüştürülür. Denklem sistemi çözüldükten sonra, sayısal türev hesaplamaları yapılır. Son olarak, tüm ifadeler dokümana aktarılır ve süreç tamamlanır. Bu şekilde, giriş verilerinden başlayarak nihai dokümana kadar tüm adımlar sistematik bir şekilde gerçekleştirilmiştir.

2.4. Türev Formülü Hesaplama Adımları

Sayısal türev formüllerini hesaplama adımları aşağıda özetlenmiştir.

1. Adım: Giriş Parametreleri

Bu adımda, sayısal türev hesaplamaları için gerekli olan giriş parametreleri belirlenir. Bu parametreler şunlardır:

h: Adım büyüklüğü

d: Türevin derecesi

$m_1, m_2, \ldots, m_k$: Nokta yönleri (geri ve/veya ileri)

$y_0, y_1, y_2, \ldots, y_k$: Fonksiyon noktaları

Giriş: h, d, m_k ve y_k değerleri

2. Adım: Taylor Serisi Açılımı

Taylor serisi açılımı, bir fonksiyonun belirli bir noktadaki değerini ve türevlerini kullanarak yaklaşık bir değer hesaplamak için kullanılır.

FUNCTION TaylorSeries(h):

3. Adım 3: Örnek Fonksiyon Noktalarının Üretimi: $x_k=x_0+m_k*h$

Bu adımda, Taylor serisi açılımı kullanılarak örnek noktalar üretilir. Her bir örnek nokta, $x_k=x_0+m_k*h$ formülü kullanılarak hesaplanır.

y_1 = TaylorSeries(m_1*h)

y_2 = TaylorSeries(m_2*h)

…….

y_k = TaylorSeries(m_k*h)

4. Adım: Katsayı ile Çarpım

Bu adımda, örnek noktalar katsayılarla çarpılır.

$ty_1 = t_1 * y_1$

$ty_2 = t_2 * y_2$

……

$ty_k = t_k * y_k$

5. Adım: Denklemlerin Birleştirilmesi

Bu adımda, katsayılarla çarpılan denklemler birleştirilir.

combined_equation ($ty_1 + ty_2 + \ldots + ty_k$)

6. Adım: Katsayı İfadelerinin Denklemden Çıkarılması

Bu adımda, katsayı ifadeleri denklemlerden çıkarılır.

$Exp_0 = t_1 + t_2 + \ldots + t_k$

$Exp_1 = t_1 + 2 * t_2 + \ldots + k* t_k$

$Exp_2 = t_1 + 4 * t_2 + \ldots + k^2 * t_k$

…….

$Exp_k = t_1 + 2^k * t_2 + \ldots + k^k * t_k$

7. Adım: Denklem Sisteminin Çözümü

Bu adımda, denklem sistemi çözümlenir. Matris formunda katsayılar aşağıdaki gibi gösterilir. Ancak Exp_0 ve girişten alınan d değişkeninin değerine göre ilgili Exp_d ifadesi hariç, geri kalan Exp ifadelerinin katsayıları matrise atılır.

$| t_1\ t_2 \ldots t_k | = | 1\ 2 \ldots k |$

$| t_1\ t_2 \ldots t_k | = | 1\ 4 \ldots k^2 |$

……..

$| t_1\ t_2 \ldots t_k | = | 1\ 2^k \ldots k^k |$

8. Adım: Gauss Eliminasyon Adımları

Bu adımda, Gauss eliminasyon yöntemi kullanılarak katsayılar hesaplanır.

SolveCoefficients(matrix)

9. Adım: Katsayılar için Tamsayı Değerlerini Hesaplama

Bu adımda, katsayıların en küçük ortak katı hesaplanarak tam sayı değerleri bulunur; k=4 için aşağıdaki değerler hesaplanır.

t4 = LCM(1, 1, 3) (En küçük ortak kat)

t1 = -16 * t4, t2 = 12 * t4, t3 = -16 / 3 * t4

10. Adım: Katsayılara Değer Atama

Bu adımda, katsayılara hesaplanan değerler atanır.

y0 = t1 + t2 + t3 + t4

y1 = t1, y2 = t2, y3 = t3, y4 = t4

11. Adım: Türevi Hesapla

Son adımda, türev hesaplanır.

y_d = (-25 * y0 + 48 * y1 - 36 * y2 + 16 * y3 - 3 * y4) / (12 * h)

2.5. Girdi Bilgilerinin Alınması

Bu çalışmanın ilk aşamasında tasarlanan gramer, uygulamanın başlangıç aşamasını oluşturur. Gramer tanımlama sürecinde, girdi verileri için belirlenen sözdizimine uygun kurallar düzenlenir.

Tablo 9. Girdi ifadelerini içeren txt dosyası

```
n_diff=1;
h_size=0.001;
m_list={1,2,3}
y_vals={0.7071,0.7078,0.7085,0.7092 };
```

Tablo 9'da yer alan girdi dosyası, TXT formatında olup, sayısal türev için gerekli parametreleri tedarik eder. Bu dosyadan elde edilen veriler, 'm_list' adı verilen bir dizide saklanır; bu dizi, fonksiyonun nokta sayısını temsil eder ve hesaplamaların ileri farklar, geri farklar veya merkezi farklar yöntemiyle yapılmasını sağlar. Örneğin, 0'dan küçük sıralı değerler içeren bir 'm_list' dizi, geri farklar yönteminin kullanılacağını gösterirken; 0'dan büyük sıralı değerler içeren bir 'm_list' dizi, ileri farklar yöntemini işaret eder. Eğer 'm_list'dizisinde her iki aralıktan sıralı değerler varsa, bu durum merkezi farklar yöntemini belirtir. Dosyadaki 'y_vals' dizisi ise, bu noktalara karşılık düşen fonksiyon değerlerini barındırır. 'n_diff' değişkeni fonksiyonun türev derecesini, 'h_size' değişkeni ise hesaplama sırasında kullanılacak adım boyutunu ifade eder.

Girdi dosyasındaki 'y_vals' dizisinin uzunluğu, 'm_list' dizisinin uzunluğundan daima bir fazla olmalıdır. Bu kural, tüm sonlu fark yöntemleri için geçerlidir. 'y_vals' dizisinin ilk değeri, fonksiyonun hesaplanmak istenen noktadaki değerini temsil eder. Örneğin, nokta a ise, fonksiyonun değeri f(a) olarak belirtilir. 'y_vals' dizisinin diğer değerleri ise, 'm_list' dizisindeki noktalara karşılık gelen fonksiyon değerlerini ifade eder; yani f(a+m_list[i]·h_size) şeklindedir. Böylece, 'y_vals' dizisi, 'm_list' dizisinden bir eleman fazla olur.

Programın işleyişi sırasında, girdi dosyasındaki veriler sözdizimi ve anlamsal analizlerle incelenir. Herhangi bir uyumsuzluk tespit edildiğinde, program hatanın sebebini

ve yerini kullanıcıya bildirir. Ayrıca, 'm_list' ve 'y_vals' dizilerinin uygun uzunlukta olup olmadığı kontrol edilir ve herhangi bir hata durumunda kullanıcıya detaylı geri bildirim sağlanır.

Girdi dosyasındaki verilerin türleri çeşitlilik gösterebilir; 'm_list' ve 'y_vals' dizileri double, float, integer ve rasyonel sayı türünde ifadeler içerebilir. Özellikle integer, double ve float sayı türlerindeki işlemlerde ortaya çıkabilecek yuvarlama hatalarını önlemek için kullanıcılar rasyonel ifadeler tercih edebilir. Rasyonel ifadeler, işlemlerin sonucunda oluşabilecek hassasiyet kayıplarını minimize eder ve matematiksel işlemlerde daha doğru sonuçlar alınmasını sağlar. Adım boyutunun küçük tutulması, fonksiyonun sayısal yaklaşımla elde edilen sonucunun gerçek değere daha yakın olmasını sağlar; küçük adım boyutları daha hassas sonuçlar elde etmeyi mümkün kılar. Sayısal türev işlemleri, verilen fonksiyonun daha detaylı analiz edilmesini ve sonuçlarının elde edilmesini sağlar, bu yöntemler matematiksel hesaplamalarda ve çeşitli bilimsel alanlarda önemlidir.

2.6. JavaCC'de Kural Bildirimi

Gramer kurallarının JavaCC belirtim formatına göre tanımlandığı aşamadır. Bu aşamada bir ifadeyi ayrıştırmak ve işlemek için JavaCC kodları yazılmıştır. Kod, öncelikle JavaCC'nin sözdizimsel analiz yeteneklerini kullanarak girdi ifadeleri ayrıştırabilecek metotlardan oluşmaktadır. Bu ayrıştırma işlemi, tanımlanan birim sözcükler ve türetim kuralları aracılığıyla gerçekleştirilmektedir. Girdi ifadelerindeki sayılar, operatörler ve diğer sözdizimsel öğeler belirlenerek karşılığı olan birim sözcükler üretilir. Ayrıştırma sürecinde, birim sözcüklere dayalı işlem türleri belirtilir.

2.6.1. Üretim Kurallarının Oluşturulması

JavaCC'de 'options' bölümü, ayrıştırıcının özelliklerini belirlemek için kullanılır. Bu bölümde yapılan ayarlar, JavaCC tarafından oluşturulan kodun nasıl çalışacağının bildirimini yapar.

Tablo 10. Static ve lookahead tanımlamaları

```
options { STATIC = true; // Ayrıştırıcı metotlarını statik yapar
 LOOKAHEAD = 1; // Lookahead değerini 1 olarak ayarlar}
```

Tablo 10’da yer alan STATIC değişkeni, ayrıştırıcı metotlarının statik (sınıf metotları) ya da dinamik (nesne metotları) olarak tanımlanmasını sağlar. Örneğin, STATIC = true; ayarı, ayrıştırıcı metotlarının statik olarak oluşturulacağını ve böylece sınıfın bir örneği olmadan doğrudan sınıf adı üzerinden çağrılabileceği anlamına gelir.

LOOKAHEAD seçeneği ise ayrıştırıcının ayrıştırma sırasında ne kadar ileriye bakacağını belirler. Örneğin, LOOKAHEAD = 1; ayarı, ayrıştırıcının her adımda yalnızca bir birim sözcük ileriye bakacağını gösterir. LOOKAHEAD = 2; seçeneği ise iki adım ileri bakılacağını belirtilir. LOOKAHEAD’in 1 veya 2 olması basit durumlar için yeterlidir, ancak daha karmaşık gramer yapıları için daha büyük bir LOOKAHEAD değeri gerekebilir.

JavaCC'de tanımlanan ayrıştırıcı sınıfı, ayrıştırma işleminin merkezinde yer alan bir Java sınıfıdır. Ayrıştırıcı sınıfı ile metin dosyalarının yönetimi sağlanır. Bu sınıf üretim kurallarını uygular ve karşılaşılabilen hatalı durumlarla ilgilenebilir. Girdi verisi, belirtilen dil bilgisi kurallarına uygun değilse, hata mekanizması devreye girer ve kullanıcıya anlaşılır hata mesajları sağlanır. Örneğin hata mesajı hatanın satır numarasını, hatanın kaynağını, beklenen birim sözcük tipini ve kullanıcının girdiği birim sözcük tipini gösterebilmektedir. Bu ayrıştırma mekanizması sayesinde, kullanıcı hatasını hızlı bir şekilde düzeltebilir.

Tablo 11. Gramer kuralları

```
void Command() :
{}{ ( <M_LIST> "=" "{" e1=Elements() "}" ";"
 | <Y_LIST> "=" "{" e2=Elements2() "}" ";"
 | <N_DIFF> "=" e3=Integer() ";"
 | <H_SIZE> "=" e4=Rational() ";" )*
 <EOF>
}
void Elements() : {} { Integer() ( "," Integer() ) * }
void Elements2() : {}{ Rational()( "," Rational() ) *}
void Integer() :{}{ t=<INTEGER>{}}
void Rational() :{}{ <INTEGER>
(<SLASH> <INTEGER>)?
| <FLOAT_LITERAL> }
```

Tablo 11'de yer alan gramer kuralları, nokta sayısına bağlı olarak sayısal türev hesaplaması yapabilmek amacıyla tanımlanmıştır. Bu gramer kuralları, belirli bir dilin komutlarının ve elemanlarının sözdizimsel yapısını temsil etmek için kullanılır. Command

ismindeki üretim kuralı, dört farklı komut biçimini içermektedir: M_LIST, Y_LIST, N_DIFF ve H_SIZE. M_LIST ve Y_LIST komutları, sırasıyla Elements ve Elements2 gramer kurallarına uygun elemanlardan oluşan listeler içerirken, N_DIFF bir tam sayı ve H_SIZE ise bir double türünde sayı ifadeleri içerir. Elements kuralı, bir veya daha fazla tam sayı değerinden oluşan bir listeyi tanımlarken, Elements2 kuralı, bir veya daha fazla double türünde sayı değerinden oluşan bir listeyi tanımlar. Bu kurallar, dosyadan alınan nokta sayısı, türev değerleri ve adım büyüklüklerine bağlı olarak sayısal türevlerin doğru bir şekilde hesaplanması için dilin sözdizimini belirlemek ve yapılandırılmış ifadeler oluşturmak amacıyla kullanılmaktadır.

2.6.2. Birim Sözcük Tanımlamaları

Ayrıştırıcı tarafından kaynak verinin temel bileşenlerine ayrılabilmesi için tanımlanan ifadeler bütünüdür. Bu temel bileşenler dilin kabul edeceği birim sözcüklerden oluşmaktadır. Tablo 12'de, EBNF notasyonunda birim sözcük tanımlamasını gösteren bir kod örneği görülmektedir. Bu sınıfa ait birim sözcük ifadeleri türev hesaplamasının boyutuna göre arttırılıp azaltılabilmektedir.

Tablo 12. Birim sözcük tanımlamaları

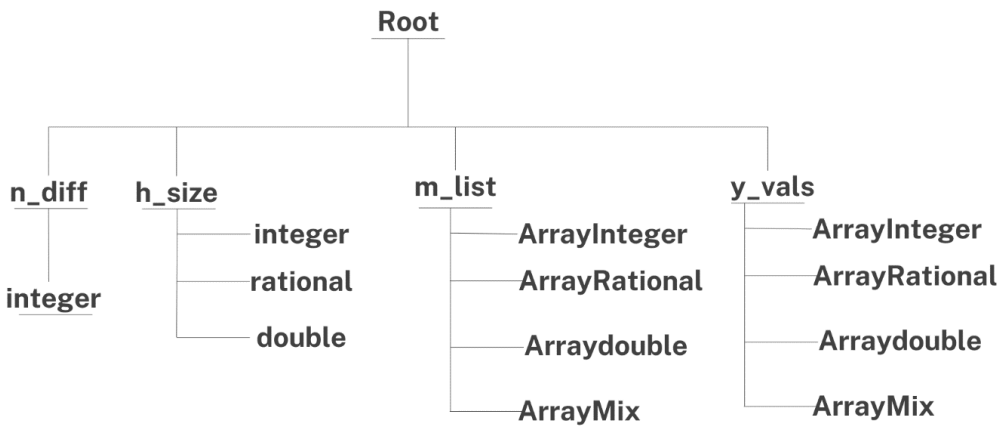
```
TOKEN:
{  <FLOAT_LITERAL: (<DIGIT>)+ "."
<DIGIT>)+>}
TOKEN :
{   <INTEGER : ("-")?(["0"-"9"])+ >
| <DIGIT : ("-")?(["0"-"9"])+ >
| < SLASH: "/" >
| <M_LIST : "m_list" >
| <Y_LIST : "y_vals" >
| <N_DIFF : "n_diff" >
| <H_SIZE : "h_size" >}
```

Tablo 12'de yer alan sözcük tanımlamaları, bir dilin sözdizimsel birimlerini temsil etmek amacıyla kullanılan birim sözcük çeşitlerini içerir. Bu tanımlamalar, dilin kaynak kodunu belirli anlamlı birimlere ayırmak için kullanılır. Gramerde, FLOAT_LITERAL ondalık sayıları, INTEGER ve DIGIT tam sayıları, SLASH eğik çizgi karakterini, M_LIST nokta sayılarını belirten anahtar kelimeyi, Y_LIST noktalara karşılık gelen değerlerin listesini, N_DIFF türev derecesini ve H_SIZE adım büyüklüğünü tanımlar. Bu birim

sözcük tanımları, girdi dosyasını ayrıştırarak her bir bileşeni doğru bir şekilde tanımlayıp işlemenizi sağlar, böylece türev hesaplamaları için gerekli olan veriler programa aktarılır.

2.7. Soyut Sözdizim Ağacı

JavaCC kullanılarak, metin içerisinde yer alan verilerin önceden belirlenmiş gramer kuralları çerçevesinde birim sözcüklere ayrılmasını ve bu birim sözcüklerin bir ağaç yapısında organize edilmesiyle soyut sözdizim ağaçları inşa edilir. Ayrıştırılan her bir birim sözcük, JavaCC tarafından oluşturulan soyut sözdizimi ağacı'nın bir düğümü olarak atanır. Örneğin n_diff, h_size, m_list ve y_vals gibi değişkenler ve bunların değerleri, belirlenen gramer kuralları ile tanımlanır ve her biri için uygun bir AST düğümü oluşturulur.



Şekil 7. Ağaç yapısı

Şekil 7’de yer alan AST’nin alabileceği düğümler ve her düğümün alabileceği değişken tiplerine yer verilmiştir.

Bir AST oluşturmanın ardındaki sözdizim sınıflarının işlevselliğini ve önemini anlamak önemlidir. AST, programlama dilindeki ifadelerin ayrıştırılmasından sonra elde edilen, dilin gramer yapısına uygun hiyerarşik bir ağaç yapısıdır. Bu yapının oluşturulması, sözdizim sınıfı nesnelerinin bağlı listeler biçimine ilişkilendirilmesini gerektirir. Sözdizim sınıfları, dilin gramer kurallarına dayalı olarak analiz edilen giriş metni bileşenlerini temsil eder ve bu birim sözcükleri mantıksal bir yapıda düğümlere dönüştürür.

Kök düğümü, AST'nin ana düğümüdür ve diğer tüm düğümleri içerir. n_diff, h_size, m_list, ve y_vals düğümleri, bu kök düğüm altında uygun bir hiyerarşiyle sıralanır.

Uygulamada, Tablo 9'da belirtilen girdi parametrelerine göre sözdizim sınıflarının ilişkilendirilme süreci:

- n_diff düğümü, içerisinde 1 olan türev değerini barındıran bir Num sınıfı ile ilişkilendirilir.
- h_size düğümü, 0.001 adım büyüklük değerini barındıran başka bir double sınıfı içerir.
- m_list içerisinde Num, RNum ve dNum ile ilişkilendirilir.
- y_vals, içerisinde Num, RNum ve dNum ile ilişkilendirilir.

Tablo 13. Uygulamanın sözdizim sınıfı

```
abstract class Exp {
 public abstract Object accept(Visitor  v);}
class Taylor extends Exp{
Exp e1=null,e2=null,e3=null,e4=null;
public Taylor(Exp a, Exp b, Exp c,Exp d)
{ e1=a; e2=b; e3=c; e4=d;
 }
 public String toString()
{
return "Taylor(" +e1+ " " +e2+" "+e3+" "
+e4+")"; }
 public Object accept(Ziyaretçi v) {
 return v.visit(this); }}
```

```
class Yeni_Taylor extends Exp{ Object[][]
e1;
public Yeni_Taylor(Object[][] a){
 e1 = a; }
public String toString(){
 return "Yeni_Taylor("+e1+")";
} public Object accept(Visitor  v) {
 return v.visit(this); }}
class Num extends Exp{
 int sayi;
 public Num(int a){ sayi=a; }public String
toString(){ return "Num(" +sayi+")"; }
public Object accept(Visitor  v) { return
v.visit(this); }}
```

Tablo 14. Ziyaretçi yapısı

```
interface Visitor {
public Object visit(Exp e);
public Object visit(Taylor e);
public Object visit(m_List e);
public Object visit(y_List e);
public Object visit(Num e);
public Object visit(RNum e);
public Object visit(DRNum e);
public Object visit(Yeni_Taylor e);
public Object visit(Denklem e);
public Object visit(Gauss e);}
```

Tablo 13 ve 14'te uygulamanın bazı Java tanımlamaları ve arayüzü yer almaktadır. Bu kodlar ziyaretçi tasarım deseni kullanarak türev işlemlerinin temsil edilmesini sağlar. Exp soyut sınıfı, tüm ifadeler için temel sınıf olarak tanımlanır ve kabul yöntemiyle bir

ziyaretçiyi kabul eder. Taylor sınıfı, dört alt ifadeyi (e1, e2, e3, e4) temsil eder ve bu ifadeleri birleştirerek Taylor serisini oluşturur. Yeni_Taylor sınıfı ise, iki boyutlu bir nesne dizisi (Object [][] e1) ile çalışır ve bu diziyi temsil eder. Her iki sınıf da toString yöntemini bastırarak kendi içeriğini metin formatında döndürür. Num sınıfı, bir tam sayı değeri içerir ve bu değeri temsil eder. Tüm bu sınıflar, kabul yöntemiyle bir ziyaretçi (visitor) nesnesini kabul ederek, ziyaretçinin ilgili sınıfta tanımlı olan ziyaretçi yöntemini çağırmasını sağlar. Bu yapı, çeşitli ifade türlerinin aynı hiyerarşide işlenmesine olanak tanır ve genişletilebilir bir yapı sunar.

2.8. Sonlu Fark Şemaları ile Denklemlerin Üretilmesi

Sayısal türev hesaplamalarının doğru bir şekilde gerçekleştirilmesi için öncelikle ilgili denklemlerin üretilmesi gerekmektedir. Bu süreçte Taylor serisi genişletmelerinden faydalanılarak sonlu fark denklemleri oluşturulmuş ve bu denklemler kullanılarak türev formülleri hesaplanmıştır. Taylor serisi genişletmeleri, bir fonksiyonun belirli bir noktadaki davranışını çevresindeki noktalar cinsinden ifade eder ve böylece türevler ve integraller gibi diferansiyel işlemleri hesaplamak için kullanışlı hale gelir. Sonlu fark denklemleri ise bu genişletmelerin sonlu bir dizi terimle yaklaşık olarak temsil edilmesiyle elde edilir.

Taylor serisi genişletmeleri kullanılarak elde edilen sonlu fark denklemleri, türev hesaplamalarında yüksek doğruluk sağlamaktadır (Burden & Faires, 2010). Bu genişletmeler, fonksiyonun daha yüksek dereceden türevlerini içerir ve bu sayede daha kesin sonuçlar elde edilmesini sağlar.

Sonlu fark denklemlerinin doğruluğunu artıran bir diğer önemli faktör ise hata terimlerinin (O(h)) doğru bir şekilde ele alınmasıdır. Taylor serisi genişletmeleri, fonksiyonun türevlerini içeren terimlerin yanı sıra, bu terimlerin ihmal edilmesi durumunda ortaya çıkan hata terimlerini de içerir. Bu hata terimlerinin küçültülmesi, türev hesaplamalarının doğruluğunu önemli ölçüde artırır. Yüksek doğruluk gerektiren uygulamalarda, daha fazla terim kullanılır.

2.8.1. Noktalara Göre Denklemlerin Üretilmesi

Tezde yer alan uygulamada m_list içerisinde belirtilen noktalarda Taylor serisi açılımlarının yapılması, çok değişkenli denklemlerin oluşturulması için önemlidir. Bu denklemler sayesinde problemin konusu nümerik alana taşınabilmiştir. Bu çalışmada örnek

olarak, y_1, y_2 ve y_3 olarak adlandırılan üç farklı denklem, sırasıyla h, 2h ve 3h adım büyüklükleri ile x_0 noktasından itibaren hesaplanmıştır.

Tablo 15. Denklemlerin üretilmesi

$$y_1 = f(x_0) + h * f'(x_0) + \frac{h^2}{2!} * f''(x_0) + \frac{h^3}{3!} * f'''(x_0)$$
$$y_2 = f(x_0) + 2 * h * f'(x_0) + 2^2 * \frac{h^2}{2!} * f''(x_0) + 2^3 * \frac{h^3}{3!} * f'''(x_0)$$
$$y_3 = f(x_0) + 3 * h * f'(x_0) + 3^2 * \frac{h^2}{2!} * f''(x_0) + 3^3 * \frac{h^3}{3!} * f'''(x_0)$$

Tablo 15'te verilen denklemler, Taylor serisi genişlemesi kullanılarak farklı noktalardaki fonksiyonun denklemi üretilmiştir. Her y_i değeri, x_0 noktasında hesaplanan fonksiyon değerine ve türevlerine bağlıdır. Genel olarak, y_i terimi, x_0 noktasındaki fonksiyon değeri $f(x_0)$ ve ardından gelen türevler ($f'(x_0)$, $f''(x_0)$, $f'''(x_0)$, vb.) kullanılarak ifade edilir. Her terim, h ile çarpılan ve artan derecelerdeki türevlerin katsayıları ile kullanılarak hesaplanır. Katsayılar, türevin derecesine ve i sayısına bağlı olarak belirlenir. Bu genişlemeler, x_0 noktasına olan mesafenin h birimi ile çarpılmasıyla, x_0 noktasının etrafındaki fonksiyon değerlerini tahmin etmek için kullanılır. Özetle, y_i terimleri x_0 noktasındaki fonksiyon ve türev değerlerinin ağırlıklı toplamı olarak ifade edilmiştir.

2.8.2. Denklemlerin Katsayı Değişkenleri ile Çarpımı

Değişkenlerin uygun katsayılarla çarpılması ve denklemlerin genel bir formata dönüştürülmesi, matematiksel işlemleri kolaylaştırmak ve denklemler ile lineer cebir ortamlarında ilgilenebilmeyi sağlar.

Tablo 16. Katsayı ile çarpılmış denklemler

$$t_1 * y_1 = t_1 * f(x_0) + t_1 * h * f'(x_0) + t_1 * \frac{h^2}{2!} * f''(x_0) + t_1 * \frac{h^3}{3!} * f'''(x_0)$$
$$t_2 * y_2 = t_2 * f(x_0) + t_2 * 2 * h * f'(x_0) + t_2 * 2^2 * \frac{h^2}{2!} * f''(x_0) + t_2 * 2^3 * \frac{h^3}{3!} * f'''(x_0)$$
$$t_3 * y_3 = t_3 * f(x_0) + t_3 * 3 * h * f'(x_0) + t_3 * 3^2 * \frac{h^2}{2!} * f''(x_0) + t_3 * 3^3 * \frac{h^3}{3!} * f'''(x_0)$$

Tablo 16'da verilen denklemler, her y_i teriminin bir ağırlık faktörü t_i ile çarpılmasını ifade etmektedir. Bu çarpımlar, Taylor serisi genişlemesinin her bir bileşenine uygulanır. Her $t_i * y_i$ terimi, $f(x_0)$, $f'(x_0)$, $f''(x_0)$, vb. gibi x_0 noktasındaki fonksiyon ve türev değerlerinin ağırlıklı toplamı olarak ifade edilir. Bu genişlemeler, x_0 noktasına belirli bir mesafede yer alan fonksiyon değerlerinin, bu mesafenin farklı ağırlıklarla çarpılmasıyla hesaplanmasını sağlar.

2.8.3. Denklemlerin Birleştirilmesi

Denklemlerin sol ve sağ taraflarının toplanması adımı, farklı denklemlerden oluşan bir sistemden tek bir denklem elde etmeyi amaçlar. Bu yöntem, aynı derecede ve aynı noktada olan türev ifadesine ait katsayıları bir araya toplar.

Tablo 17.Denklemin toplanması

$$
\begin{aligned}
t_1 * y_1 + t_2 * y_2 + t_3 * y_3 & \\
&= (t_1 + t_2 + t_3) * f(x_0) \\
&+ (t_1 + t_2 * 2 + t_3 * 3) * h * f'(x_0) \\
&+ (t_1 + t_2 * 2^2 + t_3 * 3^2) * \frac{h^2}{2!} * f''(x_0) \\
&+ (t_1 + t_2 * 2^3 + t_3 * 3^3) * \frac{h^3}{3!} * f'''(x_0)
\end{aligned}
$$

Tablo 17'de yer alan denklem, farklı t_i değişkenleri ile çarpılan denklemlerin terimlerinin toplamını göstermektedir. Sol tarafta, her y_i terimi kendi t_i katsayısı ile çarpılır ve bu terimlerin toplamı alınır. Sağ tarafta ise, x_0 noktasındaki fonksiyon değeri ve farklı derecedeki türevleriyle çarpılan katsayıların toplamı bulunur.

2.8.4. Denklemden Katsayı İfadelerinin Çıkarımı

Birleştirilen denklemlerden katsayılı ifadelerine dayalı bir denklem sisteminin oluşturulduğu aşamadır. Bu aşamada, Tablo 17'de belirtilen denklemlerden türev ifadeleri, h değerleri ve paydadaki ifadelerin atılarak t katsayılarından oluşan denklemler elde edilir. Her bir denklem Exp_i şeklinde adlandırılan ifadeyi oluşturur. Bu şekilde, denklem sistemine nümerik alan işlemlerini uygulamak mümkün olur.

Tablo 18. Katsayı ifadelerine dayalı denklemler

$Exp_0: t_1 + t_2 + t_3$
$Exp_1: t_1 + t_2 * 2 + t_3 * 3$
$Exp_2: t_1 + t_2 * 2^2 + t_3 * 3^2$
$Exp_3: t_1 + t_2 * 2^3 + t_3 * 3^3$

Tablo 18'de verilen ifadeler, her bir t_i katsayısının belirli kuvvetlerinin alınmasıyla oluşan toplamları ifade eder ve fonksiyonun x_0 noktasındaki belirli türevlerini hesaplamada kullanılır. Örneğin, Exp_0: $t_1+t_2+t_3$ ifadesi, tüm ağırlık faktörlerinin toplamını ifade eder ve f(x_0) katsayılı değişkenlerin toplamıdır. Exp_1: $t_1+t_2*2+t_3*3$ ifadesi, her bir ağırlık faktörünün sırasıyla 1, 2 ve 3 ile çarpılmasıyla elde edilir ve bu, f'(x_0) katsayılı değişkenlerin toplamıdır.

2.8.5. Yeni Denklem Sisteminin Oluşturulması

Bu aşamada üretilen denklemler, çeşitli denklem sistemi çözme tekniklerini uygulamak amacıyla sıfıra eşitlenir. Bu süreç, elde edilen denklemlerin, sayısal yöntemlerle çözülebilir hale getirilmesi için zorunludur.

Tablo 19. Denklemlerin sıfıra eşitlenmesi

$t_1 + t_2 * 2^2 + t_3 * 3^2 = 0$
$t_1 + t_2 * 2^3 + t_3 * 3^3 = 0$

Tablo 18'de verilen denklem kümesinde istenilen türev derecesinin hesaplamasını yapabilmek için bazı ifadeler çıkartılarak Tablo 19'da verilen denklem sistemi oluşturulmuştur. Belirli bir türev ifadesini bulabilmek için, bu türevin derecesini içeren katsayı ifadesi ile fonksiyonun x_0 noktasındaki değerini içeren katsayı ifadesi dışında kalan katsayı ifadelerinin sıfırlanmasını sağlamak içindir. Birinci türevi hesaplamak istediğimizde, denklem serisinde yer alan ifadelerden Exp_0 ve Exp_1 ifadelerini çıkartırız. Bu işlem sonucunda geriye Exp_2 ve Exp_3 ifadeleri kalır ve birinci türevin hesaplanması bu denklemler üzerinden gerçekleştirilir. Benzer şekilde, ikinci türevi hesaplamak istediğimizde, denklem serisindeki Exp_0 ve Exp_2 ifadelerini çıkartırız. Bu durumda geriye kalan ifadeler Exp_1 ve Exp3 olur ve ikinci türevin hesaplanması bu denklemler üzerinden

yapılır. Bu şekilde, türev hesaplamalarında hangi ifadelerin çıkarılması gerektiğini ve hangi ifadelerin kullanılacağını belirleyerek işlem yapabiliriz.

$$\begin{bmatrix} t_1 & t_2 & t_3 \\ 1 & 2^2 & 3^2 \\ 1 & 2^3 & 3^3 \end{bmatrix}$$

Şekil 8. Genel katsayısı matrisi

Gauss eliminasyon yöntemi, çalışmada üretilen denklem sistemini çözmek için kullanılmaktadır. Denklemler, bu yöntem uygulanmadan önce katsayıları bir matrise yerleştirilir, böylece lineer denklem sistemlerinin çözümüne olanak tanınır.

$$\left[\begin{array}{cc|c} t_1 & t_2 & t_3 \\ 1 & 4 & -9 \\ 1 & 8 & -27 \end{array}\right]$$

Şekil 9. Yeni katsayı matrisi

Şekil 8'de yer alan Katsayı matrisi oluşturulduktan sonra Gauss eliminasyon işlemi için bağımsız değişken sağ tarafa çekilerek Şekil 9'da görüldüğü gibi yeni matris oluşturulur. Böylece matris genişletilmiş katsayılar matrisi formuna dönüştürülür ve Gauss eliminasyon yöntemini uygulanmak üzere hazırlanır.

2.8.6. Doğrusal Denklem Sisteminin Çözümü

Önceki bölümde anlatılan denklem sistemini çözebilmek için katsayılar matrise atılmıştır. Bu kısımda ise denklem sisteminin nümerik alanda çözümü ele alınacaktır. Bu süreç, katsayı matrisi üzerinde Gauss eliminasyon yöntemini uygulayarak, sistemin çözümünü adım adım ilerletir. Gauss eliminasyonu, matrisi üst üçgensel, alt üçgensel veya birim matris formuna dönüştürür ve ardından yerine koyma gibi işlem adımlarıyla sistemdeki bilinmeyenlerin belirlenmesini sağlar. Bu adım, matematiksel ifadeleri daha basit bir formda ifade ederek, lineer cebir teknikleriyle sistemin çözümünü elde etmeyi

amaçlar. Şekil 10'da birim matris dönüşümüyle denklem sisteminin çözümü gösterilmektedir.

$$\left[\begin{array}{cc|c} t_1 & t_2 & t_3 \\ 1 & 0 & 9 \\ 0 & 1 & \frac{-9}{2} \end{array}\right]$$

Şekil 10. Denklem sisteminin çözümü

Şekil 10'da yer alan matristen de anlaşılacağı üzere değişkenlerin değerleri, t_3 bağımsız değişken yapılarak hesaplanır.

$t_1 = 9 * t_3$

$t_2 = -9/2 * t_3$

2.8.7. Tamsayı Değerleri İçin Katsayı Değerlerinin Hesaplanması

Bölüm 2.8.6'da bulunan t_1, t_2 ve t_3 değişkenlerine en uygun tamsayı değerinin atanabilmesi için EKOK hesaplama yönteminden faydalanılmıştır. Bu sayede değişken değerleri tamsayı cinsinden ifade edilmiştir. EKOK hesaplamak için hızlı olması hasebiyle oklid algoritması tercih edilmiştir.

Tamsayı değeri elde edebilmek için t_1 ve t_2 değişkenlerinin paydalarının EKOK'u hesaplanmıştır. Bulunan EKOK değeri t_3 değişkenine atanmıştır; böylelikle tüm değişkenler tamsayı türünden ifade edilmiştir.

$t_3 = EKOK\ (1, 2) = 2$

EKOK değeri görüldüğü üzere 2 olarak bulunmuştur. Diğer hesaplamalar ise aşağıdaki gibidir.

$t_1 = 9 * 2 = 18,$

$t_2 = -9/2 * 2 = -9$

$t_3 = 2$

2.9. İfadenin Hesaplanması ve Düzenlenmesi

İfadenin düzenlenmesi ve hesaplanması aşamasında bulunan değişkenlerin değerleri Bölüm 2.8.3'te yer alan birleştirilmiş denklem sisteminde yerine koyulur ve denklemler yeniden düzenlenir.

Tablo 20. Sayısal hesaplama işlemleri

$$y^{(1)} = \frac{-11 * y_0 + 18 * y_1 - 9 * y_2 + 2 * y_3}{6 * h} + O(h^3)$$

$$y^{(1)} = \frac{-11 * 0.7071 + 18 * 0.7078 - 9 * 0.7085 + 2 * 0.7092}{6 * 0.001} + O(h^3)$$

$$y^{(1)} = \frac{0.04242616}{0.06} = 0.7071$$

$$y^{(1)} = 0.7071 + O(h^3)$$

Elde edilen katsayılar denklemde yerine konulmuştur ve Tablo 20'de görüldüğü gibi bir örnek fonksiyonun türev ifadesi elde edilmiştir.

2.10. İşlem Adımlarının Dokümana Aktarılması

Türev ifadelerinin üretimleri LaTeX dilinde formatlanarak PDF dokümanı olarak kaydedilmektedir. LaTeX, bilimsel ve matematiksel belgelerin estetik bir biçimde düzenlenmesi için yaygın olarak kullanılan bir yazım sistemidir. LaTeX dilindeki yazım, matematiksel semboller, ifadeler ve denklemlerin doğru ve etkili bir şekilde biçimlendirilmesini sağlar. Özellikle matematiksel içeriklerin sunumunda, LaTeX dilinin sunduğu olanakların güçlü ve esnek olduğu unutulmamalıdır.

Tablo 21. Latex kaynak kodu

Latex	f(x) = f(x0) + f'(x0)\times(x-x0) + f''(x0)\times \frac{(x-x0)^2}{2!}+ f'''(x0)\times \frac{(x-x0)^3}{3!}+ f''''(x0)\times \frac{(x-x0)^4}{4!}
Matematiksel Notasyon	$f(x) = f(x0) + f'(x0) \times (x - x0) + f''(x0) \times \frac{(x - x0)^2}{2!} + f'''(x0) \times \frac{(x - x0)^3}{3!} + f''''(x0) \times \frac{(x - x0)^4}{4!}$

Tablo 21'deki matematiksel ifadelerin LaTeX diline dönüşüm işlemi, her adımda sistemli bir şekilde gerçekleştirilmiştir ve bu süreç, bir PDF belgesi oluşturmak amacıyla kullanılacak olan tex uzantılı dosyanın içeriğine entegre edilmiştir. İlk etapta, AST ve ziyaretçi tasarım deseni kullanılarak matematiksel ifadelerin temsil edildiği bir yapı oluşturulmuştur. Daha sonra bu ağaç üzerinde gezinilerek Taylor serisi açılımı hesaplanmış ve sonrasında elde edilen ifadeler LaTeX diline çevrilmiştir. Taylor serisinin açılımından yararlanarak lineer denklem sistemi oluşturulmuş ve bu sistem Gauss eliminasyon yöntemi ile çözülmüştür.

Her bir adım, LaTeX dokümanının ilgili bölümlerine eklenerek, adım adım açıklamalı bir LaTeX dokümanı oluşturulmuştur. Bu tex dosyası, bir LaTeX derleyicisi kullanılarak PDF formatına dönüştürülür. Böylece, matematiksel notasyon dönüşüm sürecini içeren kapsamlı bir çıktı elde edilir.

3. BULGULAR

Yazılımın üretim sürecinde türev ifadelerinin belirlenmesinde kullanılan matematiksel hesaplamalar, AST ve ziyaretçi tasarım deseni yer almaktadır. Bu hesaplamalar, Taylor serisinin genişletilmesi, lineer denklem sistemlerinin oluşturulması, Gauss eliminasyon yöntemi ile çözülmesi ve türev ifadelerinin üretilip değerlendirilmesini içermektedir.

Yazılım ile herhangi bir fonksiyonun farklı mertebeden türevlerinin başarıyla hesaplanabildiği gözlemlenmiştir. Türev hesaplamaları, ileri yönlü, geri yönlü ve merkezi fark yöntemleri kullanılarak ayrı ayrı gerçekleştirilebilmektedir. Bu yöntemler, fonksiyon türevlerinin doğruluğunu ve hesaplama esnekliğini artırarak, çeşitli analizler için güçlü bir Java aracı olarak sunulmuştur.

Sayısal hesaplamalar sırasında, özellikle rasyonel aritmetik işlemleri için büyük sayılarla çalışmak gerektiğinden, standart veri tiplerinin bu sayıları doğru bir şekilde temsil etmekte yetersiz kaldığı gözlemlenmiştir. Bu yetersizlikler, sayısal kayıplara yol açmıştır. Bu tür kayıpları azaltmak amacıyla, BigInteger ve BigDecimal gibi uygun veri tipleri kullanılmış ve ara sonuçlar yüksek hassasiyetle saklanmıştır. Bu yaklaşım, hesaplamalarda oluşabilecek hataları minimize etmiş ve sonuçların doğruluğunu önemli ölçüde artırmıştır. Bu sayede geliştirilen Java aracı diğer uygulamalara güçlü bir alternatif olmuştur.

Çalışmada fonksiyonların farklı mertebeden türevlerini içeren örneklemeler sunulmaktadır. Hesaplanan bu sonuçlar, Matlab ortamında elde edilen değerlerle karşılaştırılmış ve bu karşılaştırmalar sonucunda iki hesaplama arasında yakınlık gözlemlenmiştir. Bu uyum, geliştirilen yazılımın türev hesaplamalarını başarılı bir şekilde gerçekleştirebildiğini ve doğru sonuçlar üretebildiğini teyit etmektedir.

Tablo 22. Türev sonuçları

Fonksiyon	**Türev**	**Matlab**	**Uygulama**	**Gerçek Değer**
sin(x)	1.Türev	0.7071	0.7071	0.7071
exp(-x)*sin(x)	2.Türev	-1.0260	−1.0260	-1.0260

Tablo 22'de uygulamanın elde ettiği sonuç, Matlab aracının elde ettiği sonuç ve gerçek değer yer almaktadır. Geliştirilen uygulamanın hem türevi kolay alınan fonksiyonlar için hem de türevi zor olan fonksiyonlar için başarılı bir şekilde türev hesapladığı ve bu işlemleri herhangi bir performans kaybı yaşamadan gerçekleştirdiği görülmüştür.

Çalışmada ele alınan bazı fonksiyon örnekleri ve noktasal örnekleme kümeleri üzerinden yapılan türev hesaplamalarının doğruluğu değerlendirilmiştir. Özellikle, nokta sayısının artmasıyla elde edilen türev değerlerinin gerçek değere daha yaklaştığı gözlemlenmiştir. Bu değerlendirme, sayısal türev kullanılan matematiksel hesaplamaların doğruluğunu belirlemede önemli bir adımdır. Noktasal örnekleme kümelerinin artması, hesaplamalardaki hata payını azaltabilir ve türev hesaplamalarının daha kesin sonuçlar vermesini sağlar. Bu adım, sayısal türevlerin gerçek dünya problemlerindeki doğruluğu ve güvenilirliğini değerlendiren önemli bir araştırma aşamasını temsil etmektedir.

Tablo 23. 6 Noktalı Örnek fonksiyonların türev sonuçları

Fonksiyon$(f(x))$	**Adım boyutu (h) ve x değeri**	**2. Türevi$(f''(x))$**	**3. Türevi$(f'''(x))$**
$f(x) = x^3$	x=2, h=1	12.0	6.0
$f(x) = Sin(x)$	x=pi/4, h=0.001	-0.7071	-0.7071
$f(x) = (e)^{(-x)} * Sin(x)$	x=pi/4, h=0.001	-0.6447	1.2895
$f(x) = \frac{1}{1+(x)^2}$	x=pi/4, h=0.001	0.4024	1.0567
$f(x) = 2 * x^{50} + 3 * x^{25} + 5$	x = pi/4,h= 0.001	7.0009	206.1569
$f(x) = sin(x^2) + cos(3x) + e^{(2x)}$	x =pi/4, h=0.001	25.8099	48.9623
$f(x) = (x^2 + 1)^3$	x =pi/4, h=0.001	0.4024	1. 114.685

Tablo 23'te, bazı fonksiyonların belirtilen adım ve nokta değerlerine göre elde edilen türev değerleri yer almaktadır. Bu fonksiyonların türev hesaplamaları oldukça zordur. Ancak uygulama bu hesaplamaları kolaylıkla yapabilmektedir.

3.1. Uygulamanın Performansı

Çalışmanın dikkat çeken bir diğer noktası ise, herhangi bir fonksiyon için verilen nokta sayısına göre türev hesaplama işlemlerinde, nokta sayısı ve değeri aynı olduğunda Matlab gradient ve Python derivative ve gradient türev hesaplama araçlarına göre daha gerçekçi sonuçlar elde edilebilmesidir.

Tablo 24. Uygulamanın Diğer Uygulamalar ile Karşılaştırılması

Fonksiyon	x Değeri	h Değeri	Maple, Python Matlab Sonucu	Uygulama Sonucu	Gerçek Değer
$\mathbf{f(x) = (e)^{(-x)} * Sin(x)}$	0	π/4	0.9003	0.95637	1
$\mathbf{f(x) = \frac{1}{1 + (x)^2}}$	π/4	1	-0.3000	-0.48825	-0.5
$\mathbf{f(x) = \frac{1}{1 + (x)^2}}$	0	π/20	1.8130	1.9946	2

Matlab, Python ve Maple bünyesindeki sayısal türev alma işlemleri, benzer süreç içerdikleri için aynı sonuç elde edilmiştir. Tablo 24'te üç farklı fonksiyonun belirli x ve h değerleri için söz konusu üç araç kullanılarak hesaplanan türev sonuçları ile uygulama sonuçlarını ve gerçek değerleri karşılaştırmaktadır. İlk fonksiyon (f(x) = e^{-x} sin(x)) için x = 0 ve h = π/4 değerlerinde Python ve Matlab sonucu 0.9003 iken, uygulama sonucu 0.95637 ve gerçek değer 1'dir. Bu, uygulama sonucunun gerçek değere daha yakın olduğunu göstermektedir. İkinci fonksiyon (f(x) = 1 / (1 + x^2)) için x = π/4 ve h = 1 değerlerinde Python ve Matlab sonucu -0.3000, uygulama sonucu -0.48825 ve gerçek değer -0.5'dir. Burada da uygulama sonucu gerçek değere daha yakındır. Üçüncü fonksiyon (f(x) = sin(x^2) + cos(3x) + e^{2x}) için x = 0 ve h = π/20 değerlerinde Python ve Matlab

sonucu 1.8130, uygulama sonucu 1.9946 ve gerçek değer 2'dir. Yine, uygulama sonucu gerçek değere daha yakın bir doğruluk sergilemektedir. Bu genel tablo, uygulamanın aynı adım boyutunda Python ve Matlab'a göre daha gerçekçi ve doğru sonuçlar elde ettiğini göstermektedir.

3.2. Uygulamanın Kısıtlamaları

Uygulama, belirli nokta sayıları ve adım boyutları için türev hesaplamaları gerçekleştirirken bazı kısıtlamalara tabi olmaktadır. Bu kısıtlamalar, artan nokta sayısı ve denklemlerin boyutlarının büyümesi, sayısal değerlerin büyüklüğü, hesaplama hataları ve bazı noktaların hesaplanamaması gibi konuları kapsamaktadır.

- Artan Nokta Sayısı ve Denklem Boyutları: Nokta sayısının artması, çözülecek denklemlerin boyutunu da artırmaktadır. Bu durum, çözüm yönteminin hesaplama süresini uzatmakta ve verimliliği düşürmektedir. Artan nokta sayısıyla birlikte, hesaplamaların karmaşıklığı artmakta ve daha fazla bellek kullanımı gerekmektedir. Bu da özellikle büyük veri setlerinde işlem süresinin uzamasına ve verimliliğin azalmasına yol açmaktadır.
- Sayısal Değerlerin Büyüklüğü: Sayısal değerlerin çarpımı sonucu çok büyük ifadeler elde edilebilmektedir. Bu durum, kullanılan veri tiplerinin kapasitesini aşabilir ve bellek taşmalarına neden olabilir. Sonuç olarak, türev hesaplamalarının doğruluğunu olumsuz yönde etkileyebilir. Özellikle büyük sayılarla yapılan işlemlerde, kullanılan veri tiplerinin sınırlamaları ve bellek yönetimi önemli hale gelmektedir.
- Noktaların Hesaplanamaması: Uygulamada 4, 6, 8 ve 10 noktalı merkezi fark yöntemleri kullanılarak yapılan 2. türev hesaplamalarında, aynı katsayı değişkenine ait değerlerin farklı sonuç üretmesi nedeniyle bazı noktalar hesaplanamamıştır. Bu durum, metodolojinin bazı özel durumlarda yetersiz kalabileceğini ve sonuçların doğruluğunu olumsuz etkileyebileceğini göstermektedir.

Çalışmada, Tablo 25'te yer alan noktaların hesaplama yapamadığı tespit edilmiş ve bu durum detaylı olarak rapor edilmiştir. Bu olumsuz durum sadece belirli şartlar meydana geldiğinde ortaya çıkmaktadır. Bu şartlardan ilki hesaplama yöntemi olarak merkezi fark yönteminin seçilmesi ikincisi şart ise 2. türevin hesaplanmasının talep edilmesi ve son şart

noktaların Tablo 25'te gösterildiği gibi seçilmesi durumunda ortaya çıkar ve hesaplama seçilen noktalara göre yapılamayabilir.

Tablo 25. Hesaplanamayan noktalar

Formül Türü	Formül	Durum	Açıklama (Merkez fark formülü)
4 noktalı	y [2,1,2]	Hesaplanamadı.	y0'da ikinci türev, 1 nokta geri ve 2 nokta ileriye bakılarak hesaplanır.
6 noktalı	y [2,2,3]	Hesaplanamadı	y0'da ikinci türev, 2 nokta geri ve 3 nokta ileriye bakılarak hesaplanır.
6 noktalı	y [2,2,3]	Hesaplanamadı	y0'da ikinci türev, 2 nokta geri ve 4 nokta ileriye bakılarak hesaplanır.
8 noktalı	y [2,3,4]	Hesaplanamadı	y0'da ikinci türev, 3 nokta geri ve 4 nokta ileriye bakılarak hesaplanır.
8 noktalı	y [2,1,6]	Hesaplanamadı	y0'da ikinci türev, 1 nokta geri ve 6 nokta ileriye bakılarak hesaplanır.
10 noktalı	y [2,4,5]	Hesaplanamadı	y0'da ikinci türev, 4 nokta geri ve 5 nokta ileriye bakılarak hesaplanır.
10 noktalı	y [2,1,8]	Hesaplanamadı	y0'da ikinci türev, 1 nokta geri ve 8 nokta ileriye bakılarak hesaplanır.

Tablo 25'te merkezi fark formüllerinin çeşitli nokta sayılarıyla ikinci türev hesaplama yöntemlerini ve bu yöntemlerin hesaplanıp hesaplanamadığını göstermektedir. Her formülde, y_0 noktasındaki ikinci türev, belirli sayıda geriye ve ileriye bakılarak hesaplanmaya çalışılmış, ancak belirtilen formüller için hesaplama yapılamamaktadır.

3.3. Uygulamanın Arayüzü

OpenAPI, Java ile yazılmış bir uygulamanın uzaktan erişimle kullanılmasını ve diğer kullanıcıların taleplerine yanıt vermesini sağlayan bir standarttır. Bu standart, uygulamaların web üzerinden etkileşimini kolaylaştırarak, çeşitli istemcilerin (web uygulamaları, mobil uygulamalar, diğer sunucular) bu hizmetlere erişimini mümkün kılar. Geliştirdiğimiz sayısal türev hesaplama aracı, OpenAPI ile entegre edilmiştir. Bu

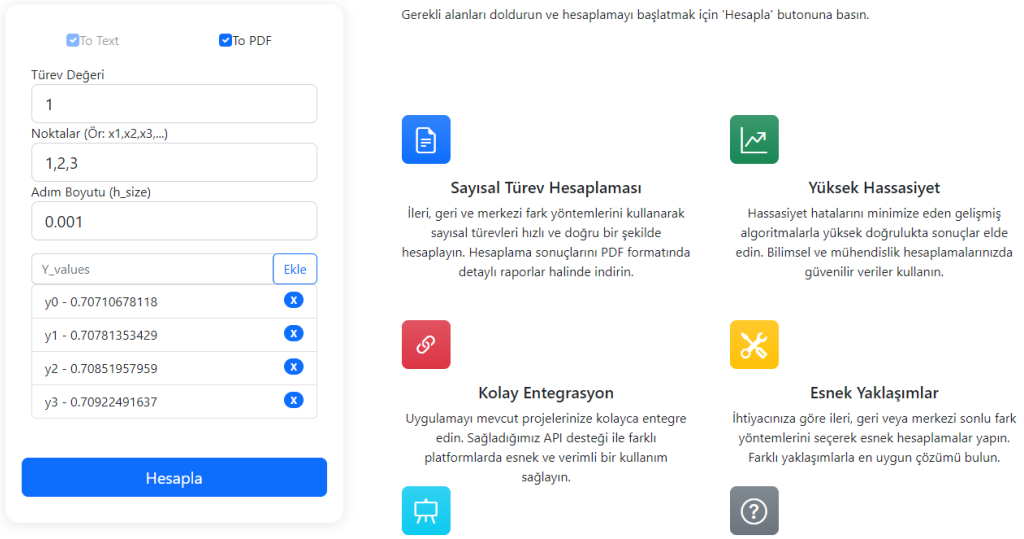
entegrasyon sayesinde, uygulama uzaktan erişimle kullanılabilir hale gelmiş ve diğer kullanıcıların taleplerine yanıt verebilir olmuştur.

OpenAPI'nin sunduğu en önemli faydalardan biri, uygulamaların geniş bir kullanıcı kitlesine ulaşmasını sağlamasıdır. Kullanıcılar, API dokümantasyonu sayesinde nasıl istek göndereceklerini ve hangi formatlarda yanıt alacaklarını kolayca öğrenebilirler. Bu, geliştiricilerin ve kullanıcıların entegrasyon süreçlerini hızlandırır ve kolaylaştırır. Ayrıca, bu standart sayesinde uygulamanın bakımı ve güncellenmesi de daha kolay hale gelir.

Uygulamanın arayüzü sade, basit ve anlaşılır olacak şekilde tasarlanmıştır. Kullanıcı dostu bu arayüz sayesinde, kullanıcılar kolayca sayısal türev hesaplamalarını gerçekleştirebilir ve sonuçları hızlıca elde edebilirler. Arayüzün kullanıcı dostu olması hem teknik bilgiye sahip olan hem de olmayan kullanıcıların rahatça uygulamayı kullanabilmesini sağlar.

Bu API sayesinde, mühendisler doğrudan istek göndererek ve sonuç alarak uygulamayı kullanabilirler. Aynı zamanda, kullanıcılar arayüzü kullanarak hizmet alabilirler. Kullanıcılar, istedikleri takdirde sonuçları Tex, LaTeX veya PDF formatında alabilirler, bu da sonuçların akademik belgeler veya teknik raporlar için uygun olmasını sağlar. Bu esneklik, uygulamanın geniş bir yelpazede kullanılmasına olanak tanır. Hem mühendisler hem de öğrenciler, uygulamayı kendi projelerinde kullanarak, sayısal türev hesaplamalarını kolay ve hızlı bir şekilde gerçekleştirebilirler. Uygulamanın esnek yapısı, kullanıcıların çeşitli formatlarda sonuçlar elde etmesine ve bu sonuçları farklı disiplinlerde ve kullanım senaryolarında uygulamalarına olanak tanır. Bu da sayısal türev hesaplama aracımızın eğitimden araştırmaya, endüstriyel uygulamalardan akademik çalışmalara kadar geniş alanda bir araç olmasına olanak tanır.

1. Türev Değeri: Kullanıcının hesaplamak istediği türevin derecesini belirttiği alandır.
2. Noktalar: Kullanıcının, hesaplama için kullanacağı noktaları virgülle ayırarak girdiği alandır (örneğin: x_1, x_2, x_3, ...).
3. Adım Boyutu (h_size): Hesaplamalarda kullanılacak adım boyutunun girildiği alandır.
4. Fonksiyon Değer Sayısı: Hesaplama için gerekli olan fonksiyon değerlerinin sayısını belirten alandır.
5. Fonksiyon Değerleri (y_0, y_1, y_2, y_3, ...): Kullanıcının belirttiği noktalardaki fonksiyon değerlerini girdiği alanlardır.



Şekil 11. Uygulama Arayüzü

Arayüzün üst kısmında, sonuçların hangi formatta alınmak istendiğini seçmek için iki seçenek bulunmaktadır; ToText ve ToPDF. Kullanıcılar, bu kutucukları işaretleyerek sonuçları metin, TeX/LaTeX veya PDF formatında almayı tercih edebilirler. Bu seçenekler, kullanıcıların farklı ihtiyaçlarına uygun şekilde sonuçları kullanmalarını sağlar.

Tüm gerekli veriler girildikten sonra, kullanıcılar “Hesapla” düğmesine tıklayarak hesaplamaları başlatabilirler. Bu düğme, kullanıcıların işlemlerini hızlı ve kolay bir şekilde gerçekleştirmelerine olanak tanır. Arayüzün kullanıcı dostu olması hem teknik bilgiye sahip olan hem de olmayan kullanıcıların rahatça uygulamayı kullanabilmesini sağlar. Bu basit ve tasarım, kullanıcıların sayısal türev hesaplamalarını kolayca yapmalarını ve sonuçları istedikleri formatta alabilmelerini mümkün kılar.

4. ÖNERİLER

Java'da kodlanan sayısal türev aracının kullanım ve hizmet düzeylerini geliştirmek amacıyla, işlevselliği genişletecek ve kullanıcı deneyimini zenginleştirecek çeşitli öneriler sunulmaktadır. İlk olarak, yazılımınıza çeşitli matematiksel hesaplama yöntemleri ve sayısal türev tekniklerinin eklenmesi, aracın uygulama alanlarını genişletecek ve daha geniş bir kullanıcı kitlesine hitap etmesini sağlayacaktır. Bu tür bir genişleme, yazılımın akademik ve endüstriyel kullanımlar için daha çekici hale gelmesine katkıda bulunur.

Kullanıcı arayüzünün daha etkileşimli ve sezgisel hale getirilmesi, yazılımın kullanım kolaylığını artırabilir. Özellikle karmaşık matematiksel işlemleri gerçekleştiren bir yazılım için, kullanıcı arayüzünün sade ve anlaşılır olması, kullanıcılara işlemlerini daha hızlı ve hatasız bir şekilde yapma fırsatı sunar. Bu, matematik ve mühendislik disiplinlerinde çalışan profesyoneller için büyük bir avantaj sağlayabilir.

Simgesel ifadelerin daha net ve kolay anlaşılır şekilde gösterilmesi de önemlidir. Bu, kullanıcıların işlemleri daha etkin bir şekilde yapmalarını sağlar ve yazılımın genel verimliliğini artırır. Böylece yazılım, daha geniş bir kullanıcı tabanı tarafından tercih edilir ve pazardaki rekabet gücü artabilir.

Yazılım aracının performansını iyileştirmek için, algoritma daha hızlı ve daha az kaynak tüketen sürümleriyle değiştirmek, performansı artırabilir. Çoklu iş parçacıklı programlama, özellikle çok çekirdekli işlemcilerde, yazılımın eş zamanlı işlemleri daha verimli bir şekilde gerçekleştirmesini sağlayabilir. Ayrıca, veri yapılarının doğru seçimi ve kullanımı, veri erişim hızını önemli ölçüde artırabilir. Java'da gereksiz kütüphane bağımlılıklarından kaçınmak, yazılımın yüklenme süresini ve hafıza kullanımını minimize edebilir. Bu stratejilerin bütünleşik uygulanması, yazılımınızın genel performansını önemli ölçüde iyileştirecek ve daha etkin bir kullanım sunacaktır.

Her bir öneri, yazılımın kapasitesini arttırmaya ve kullanıcı ihtiyaçlarına daha iyi yanıt vermeye odaklanmıştır. Bu önerilerin uygulanması, yazılımın sürekli gelişimine katkıda bulunacak ve kullanıcı memnuniyetini artıracaktır.

5. SONUÇLAR

Bu tezde bir matematiksel fonksiyona ait noktaları kümesi yardımıyla değişik mertebelerden türev hesaplayabilen bir uygulama geliştirilmiştir. Uygulama, Taylor serisi ve Gauss eliminasyonu gibi matematiksel yöntemleri kullanarak sayısal türev hesaplamaları yapabilmektedir. Bu yöntemler, belirli noktalardaki değerler üzerinden fonksiyonun türevini yaklaşık olarak hesaplamak için kullanılmaktadır. Fonksiyonların tanımlaması yapılırken Taylor serisi açılımlarından yararlanılmış ve bir doğrusal denklem sistemi ele edilmiştir. Bu denklemlerin çözümünde Gauss eliminasyon yöntemi kullanılmıştır.

Uygulamanın önemli bir özelliği, nokta sayısının artmasıyla birlikte hesaplanan türev değerlerinin gerçek değere daha yakın olmasıdır. Bu, diğer yaklaşık hesaplama yöntemlerinin doğasından kaynaklanmaktadır. Daha fazla nokta, fonksiyonun davranışını daha iyi yakalamamızı ve dolayısıyla türev hesaplamasında daha doğru sonuçlar elde etmemizi sağlamaktadır. Böyle bir uygulamanın matematik, mühendislik ve bilimsel hesaplamalar gibi alanlarda faydalı olabileceği beklenmektedir. Özellikle karmaşık fonksiyonların türevlerini analitik olarak hesaplamak zor olduğundan, bu tür sayısal yöntemler önem arz etmektedir.

Geliştirilen yazılım, sayısal türev ifadelerini üretme, değerlendirme ve belgelendirme konularında etkin bir araç olarak kullanılabilir. Özellikle farklı yönlere sahip sonlu fark yöntemlerini kullanarak çeşitli mertebeden türev ifadelerini elde etme esnekliği sunmaktadır. Türev ifadeleri, matematiksel analiz ve bilimsel hesaplamalarda çeşitli uygulamalara yönelik olarak kullanılabilir.

PDF üretimi, nümerik analiz derslerinde öğrencilere sayısal türev yöntemlerini ayrıntılı bir şekilde öğretmek amacıyla tasarlanmıştır. Bu süreç, fonksiyonun her bir noktasında Taylor serisine açılmasıyla başlamaktadır. İlk adımda, fonksiyonun her bir noktasındaki türevinin hesaplanması için gerekli olan temel teorik altyapı oluşturulur. Ardından Taylor serisi kullanılarak açılan denklemler, değişkenlerle çarpılarak yeni denklem serilerine ulaşılmaktadır. Bu adımlar, fonksiyonun türev hesaplamasına uygun hale getirilmesini sağlamak amacıyla titizlikle ele alınır. Her bir adım öğrencilere, sayısal türevin nasıl hesaplandığı konusunda derinlemesine bir anlayış kazandırmayı amaçlar.

PDF dosyası, ileri, geri ve merkezi yöntemlerle sayısal türev hesaplamasını detaylı bir şekilde açıklamaktadır. Bu yöntemlerin her birinin adım adım uygulanması, öğrenciler için fonksiyonun belirli noktalardaki türetilmesini anlama konusunda kapsamlı bir perspektif sunmaktadır. Özellikle, teorik temellerin yanı sıra uygulamalara odaklanarak, sayısal türev hesaplaması için iyi bir altyapı sağlanmaktadır. PDF dosyası, öğrencilere kendi çözümlerini üretebilme becerisi kazandırır. Fonksiyonun türetilmesi adım adım anlatılarak, öğrenciler kendi çözümlerini Pdf'nin ürettiği çözümle karşılaştırabilir ve bu sayede hatalarını kolaylıkla bulup düzeltebilir. Bu interaktif öğrenme yöntemi, öğrencilere sayısal türev hesaplamasının hem teorik temellerini hem de pratik uygulamalarını anlama konusunda güçlü bir temel sağlar.

Sonuç olarak, bu çalışma fonksiyon türevlerinin hesaplanmasında yeni ve kullanışlı bir yöntem sunmaktadır. Bu yöntem, matematik, mühendislik ve bilimsel hesaplamalar gibi alanlarda çalışan kişiler için faydalı olabilir. Nümerik analiz derslerinde öğrencilere fonksiyonun türev alma yöntemlerini öğretmek için PDF üretiminin kullanılması, öğrencilerin bu kavramları daha iyi anlamalarına yardımcı olabilir. Gelecekteki çalışmalar, bu yöntemin daha da geliştirilmesi ve genişletilmesi üzerinde odaklanarak hizmet seviyesini yükseltecektir.

6. KAYNAKLAR

Aliyu, E. O., Adewale, O. S., Adetunmbi, A. O., & Ojokoh, B. A. (2019). Requirement Formalization for Model Checking Using Extended Backus Naur Form. i-Manager's Journal on Software Engineering, 13(3), 1.

Anand, R., Aggarwal, D., & Kumar, V. (2017). A comparative analysis of optimization solvers. Journal of Statistics and Management Systems, 20(4), 623-635.

Aydoğdu, M. C., Aydoğdu, Ö., & Pehlivan, H. (2024). MxPL: A Programming Language for Matrix-Related Operations. Symmetry, 16(2), 181.

Aydoğdu, M.(2016)Belirsiz Limit İfadelerinin Otomatik Üretimi Ve Adım Adım Çözümü. K.T.Ü.,Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Trabzon.

Ba, B. (2008). JScience, a general scientific API in Java.

Bai, J., & Zhou, S. (2023). Evaluation, analysis, and the state-of-applications of Taylor expansion. Theoretical and Natural Science. https://doi.org/10.54254/2753-8818/10/20230348.

Barton, R. R. (1992). Computing forward difference derivatives in engineering optimization. Engineering optimization, 20(3), 205-224.

Belu, R., & Belu, A. (2009, June). Teaching physics with computer algebra systems. In 2009 Annual Conference & Exposition (pp. 14-1147).

Bert, C., & Malik, M. (1996). Differential Quadrature Method in Computational Mechanics: A Review. Applied Mechanics Reviews, 49, 1-28. https://doi.org/10.1115/1.3101882.

Bliek, C., Bonami, P., & Lodi, A. (2014, October). Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report. In Proceedings of the twenty-sixth RAMP symposium (pp. 16-17).

Burden, R. L. (2011). Numerical analysis. Brooks/Cole Cengage Learning.

Burden, R. L., & Faires, J. D. (2010). Numerical Analysis. Cengage Learning.

Chapra, S. C. (2010). Numerical methods for engineers. Mcgraw-hill.

Cheng, J., Jia, X. Z., & Wang, Y. B. (2007). Numerical differentiation and its applications. Inverse Problems in Science and Engineering, 15(4), 339-357.

Chomsky, N. (1959). On certain formal properties of grammars. Information and control, 2(2), 137-167.

Collatz, L. (2012). The numerical treatment of differential equations (Vol. 60). Springer Science & Business Media.,

Console, P., & Hairer, E. (2014). Reducing round-off errors in symmetric multistep methods. J. Comput. Appl. Math., 262, 217-222. https://doi.org/10.1016/j.cam.2013.07.025.

Crawley, M. J. (2012). The R book. John Wiley & Sons.

Cremers, A., & Ginsburg, S. (1975). Context-free grammar forms. Journal of Computer and System Sciences, 11(1), 86-117.

Dadashi, A. and Sari, C. (2022). Doğal potansiyel verilerinin ikinci ve dördüncü mertebeden türev analizi kullanilarak değerlendirilmesi. Deu Muhendislik Fakultesi Fen Ve Muhendislik, 24(70), 329-340. https://doi.org/10.21205/deufmd.2022247029

Daw, C. S., Finney, C. E. A., & Tracy, E. R. (2003). A review of symbolic analysis of experimental data. Review of Scientific instruments, 74(2), 915-930.

Deuflhard, P., & Hohmann, A. (2003). Numerical analysis in modern scientific computing: an introduction (Vol. 43). New York: Springer.

Dil, D. (2007). Writing parsers and compilers with PLY. PyCon'07.

Du, Q., Gunzburger, M., Lehoucq, R. B., & Zhou, K. (2012). Analysis and approximation of nonlocal diffusion problems with volume constraints. SIAM review, 54(4), 667-696.

E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of Reusable

Efendioğlu, S. (2017). Polinomlar için bir simgesel hesaplama çatısının tasarımı ve gerçeklenmesi (Master's thesis, Fen Bilimleri Enstitüsü).

Ehiwario, J. C., & Aghamie, S. O. (2014). Comparative study of bisection, Newton-Raphson and secant methods of root-finding problems. IOSR Journal of Engineering, 4(4), 01-07.

Epperson, J. F. (2013). An introduction to numerical methods and analysis.

Filannino, M., & Di Bari, M. (2015). Gold standard vs. silver standard: the case of dependency parsing for Italian. CLiC it, 141

Fornberg, B. (1996). A practical guide to pseudospectral methods. Cambridge University Press.

Fourer, R., Gay, D. M., & Kernighan, B. W. (1990). AMPL: A mathematical programming language. Management Science, 36(5), 519-554.

Foy, W. H. (1976). Position-location solutions by Taylor-series estimation. IEEE transactions on aerospace and electronic systems, (2), 187-194.

Gander, W. (2007). Teaching computational science with computer algebra. Pamm, 7(1), 1170103-1170104. https://doi.org/10.1002/pamm.200701004

Gohberg, I., Kailath, T., & Olshevsky, V. (1995). Fast Gohberg, I., Kailath, T., & Olshevsky, V. (1995). Fast Gaussian elimination with partial pivoting for matrices with displacement structure. Mathematics of computation, 64(212), 1557-1576.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Gorenflo, R., & Vessella, S. (1991). Abel integral equations (Vol. 1461, pp. viii+-215). Berlin: Springer.

Gratton, S., Lawless, A. S., & Nichols, N. K. (2007). Approximate Gauss–Newton methods for nonlinear least squares problems. SIAM Journal on Optimization, 18(1), 106-132.

Grcar, J. F. (2011). Mathematicians of Gaussian elimination. Notices of the AMS, 58(6), 782-792.

Halstead Jr, R. H. (1985). Multilisp: A language for concurrent symbolic computation. ACM Transactions on Programming Languages and Systems (TOPLAS), 7(4), 501-538.

Hanke, M., & Scherzer, O. (2001). Inverse problems light: numerical differentiation. The American Mathematical Monthly, 108(6), 512-521.

He, D. (2023). Analysis of applications for Taylor series expansion: Evidence from machine learning, mathematics and engineering. Theoretical and Natural Science. https://doi.org/10.54254/2753-8818/10/20230346.

Higham, N. J. (2011). Gaussian elimination. Wiley Interdisciplinary Reviews: Computational Statistics, 3(3), 230-238.

Hildebrand, F. B. (1987). Introduction to numerical analysis. Courier Corporation.

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. Acm Sigact News, 32(1), 60-65.

Hussain, K., Adeyeye, O., & Ahmad, N. (2023). Numerical solution of second order fuzzy ordinary differential equations using two-step block method with third and fourth derivatives. Journal of the Nigerian Society of Physical Sciences, 1087. https://doi.org/10.46481/jnsps.2023.1087

Hussein, B., Mehanna, A., & Rabih, Y. (2020). Ziyaretçi design pattern using reflection mechanism. International Journal of Software Innovation (IJSI), 8(1), 92-107.

Ji, J. (2012). Gauss–jordan elimination methods for the moore–penrose inverse of a matrix. Linear Algebra and Its Applications, 437(7), 1835-1844.

Johar, D. (2020). Application of the concept of linear equation systems in balancing chemical reaction equations. International Journal of Global Operations Research, 1(4), 130-135.

John Wiley & Sons.Thota, S., & Srivastav, V. K. (2021). An algorithm to compute real root of transcendental equations using hyperbolic tangent function. Int. J. Open Problems Compt. Math, 14(2), 1-14.

Johnson, M. (1994). Two ways of formalizing grammars. Linguistics and Philosophy, 221-248.

Josuttis, N. M. (2012). The C++ standard library: a tutorial and reference.

Kagiwada, H., Kalaba, R., Rasakhoo, N., & Spingarn, K. (2013). Numerical derivatives and nonlinear analysis (Vol. 31). Springer Science & Business Media.

Kalantari, B. (2008). Polynomial root-finding and polynomiography. World Scientific.

Khan, I. R., & Ohba, R. (2000). New finite difference formulas for numerical differentiation. Journal of Computational and Applied Mathematics, 126(1-2), 269-276.

Khang, N., Dat, D., & Tuan, N. (2019). Taylor expansion for matrix functions of vector variable using the Kronecker product. Vietnam Journal of Mechanics. https://doi.org/10.15625/0866-7136/14196.

Kodaganallur, V. (2004). Incorporating language processing into java applications: A JavaCC tutorial. IEEE software, 21(4), 70-77.

Kotan, H. (2018). Mekanik alaşımlama ile üretilen nanokristal yapılı östenitik paslanmaz çelik alaşımlarında y ve nano - y2o3 ilavelerinin tane büyümesi ve sertliğe etkisi. Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi, 2018(18-2). https://doi.org/10.17341/gazimmfd.460523

Kramarski, B., & Hirsch, C. (2003). Using computer algebra systems in mathematical classrooms. Journal of Computer Assisted Learning, 19(1), 35-45.

Kutsenko, A. A. (2021). Classification of Integrodifferential C∗-Algebras. Symmetry, 13(10), 1900.symposium on Symbolic and Algebraic Manipulation. ACM, pp. 59-75, 1971.

Laitinen, E., & Laitinen, T. (2000). Bankruptcy prediction: Application of the Taylor's expansion in logistic regression. International Review of Financial Analysis, 9, 327-349. https://doi.org/10.1016/S1057-5219(00)00039-9.

Langlois, P. (2001). Automatic Linear Correction of Rounding Errors. BIT Numerical Mathematics, 41, 515-539. https://doi.org/10.1023/A:1021919329342.

Levine, J. (2009). Flex & Bison: Text Processing Tools. " O'Reilly Media, Inc.".

Levine, J. R., Mason, T., & Brown, D. (1992). Lex & yacc. " O'Reilly Media, Inc.".

Liu, H., Ji, Y., & Wang, X. (2004). Image analysis by analogy with Taylor expansion. , 209-211. https://doi.org/10.1145/1037210.1037243.

López, J., & Temme, N. (2002). Multi-point Taylor Expansions of Analytic Functions. Transactions of the American Mathematical Society, 356, 4323-4342. https://doi.org/10.1090/S0002-9947-04-03619-0.

Lutz, M. (2001). Programming python. " O'Reilly Media, Inc.".

Lukyanenko, D., Shinkarev, V., & Yagola, A. (2022). Accounting for Round-Off Errors When Using Gradient Minimization Methods. Algorithms, 15, 324. https://doi.org/10.3390/a15090324.

Martins, J. R. R. A., Alonso, J. J., & Reuther, J. J. (2003). High-precision finite-difference methods for nonlinear problems. Journal of Computational Physics, 183(2), 273-291.

McCracken, D. D., & Reilly, E. D. (2003). Backus-naur form (bnf). In Encyclopedia of Computer Science (pp. 129-131).

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., ... &

Mizumoto, M., Toyoda, J. I., & Tanaka, K. (1975). Various kinds of automata with weights. Journal of Computer and System Sciences, 10(2), 219-236.

Mon, Y., & Kyi, L. L. W. (2014). Performance Comparison of Gauss Elimination and Gauss Jordan Elimination. International Journal of Computer & Communication Engineering Research (IJCCER), 2(2).

Moreno, P. J., Raj, B., & Stern, R. M. (1996, May). A vector Taylor series approach for environment-independent speech recognition. In 1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings (Vol. 2, pp. 733-736). IEEE.

Object-Oriented Software", Addison-Wesley Reading, MA, 1995.

Pandey, A. K., & Singh, S. (2021). Derivatives: A Comprehensive Study of Rate of Change. Journal of Applied Science and Education (JASE), 1(1), 1-4.

Parr, T. J., & Quong, R. W. (1995). ANTLR: A predicated-LL (k) parser generator. Software: Practice and Experience, 25(7), 789-810.

Parsopoulos, K. E., & Vrahatis, M. N. (2003, December). Investigating the existence of function roots using particle swarm optimization. In The 2003 Congress on Evolutionary Computation, 2003. CEC'03. (Vol. 2, pp. 1448-1455). IEEE.

Pehlivan, H. (2019). Designing and interpreting a mathematical programming language. Sakarya University Journal of Science, 23(6), 1027-1041.

Pehlivan, H. (2019). Sayısal Çözümleme Yöntemlerinin Programlanması ve Yorumlanması. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 7(1), 872-894.

Pooseh, S., Almeida, R., & Torres, D. (2012). Numerical approximations of fractional derivatives with applications. Asian Journal of Control, 15(3), 698-712. https://doi.org/10.1002/asjc.617

Potra, F. A. (1981). An application of the induction method of V. Pták to the study of

Qureshi, S., Rangaig, N., & Băleanu, D. (2019). New numerical aspects of caputo-fabrizio fractional derivative operator. Mathematics, 7(4), 374. https://doi.org/10.3390/math7040374

Rosenthal, R. E. (2007). A gams tutorial. GAMS-A User's Guide, 5(26), 649.

Saad, K., Khader, M., Gómez-Aguilar, J., & Baleanu, D. (2019). Numerical solutions of the fractional Fisher's type equations with Atangana-Baleanu fractional derivative by using spectral collocation methods. Chaos, 29 2, 023116 . https://doi.org/10.1063/1.5086771.

Schrage, L. E. (2006). Optimization modeling with LINGO. Lindo System.

Scopatz, A. (2017). SymPy: symbolic computing in Python. PeerJ Computer Science, 3, e103.

Secmen, M. and Turhan-Sayan, G. (2008). The theory and application of an electromagnetic target recognition method based on natural-resonance for multi-targets. https://doi.org/10.1109/siu.2008.4632599

Skala, V. (2021). Efficient Taylor expansion computation of multidimensional vector functions on GPU. Annales Mathematicae et Informaticae. https://doi.org/10.33039/AMI.2021.03.004.

Stewart, J. (2007). Essential calculus: Early transcendentals. Brooks/Cole, a part of the Thomson Corporation.

Stoer, J., Bulirsch, R., Bartels, R., Gautschi, W., & Witzgall, C. (1980). Introduction to numerical analysis (Vol. 1993). New York: Springer.

Strang, G. (2022). Introduction to linear algebra. Wellesley-Cambridge Press.

Tang, Y., Valko, M., & Munos, R. (2020). Taylor Expansion Policy Optimization. ArXiv, abs/2003.06259.

Thurston, A. (2003). Ragel state machine compiler. URL http://www. colm. net/open-source/ragel.

Trofimov, S., & Trofimova, O. (2018, May). Visualization of graphs of complex functions and exact geometric method of finding complex roots of a polynomial. In 2018 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT) (pp. 152-155). IEEE.

URL-1.(24, Ekim 2023). Sonlu fark:https : //tr.wikipedia.org/wiki/Sonlu_fark adresinden alınmıştır.

URL-2.(27 Ekim2023). Taylor Serisi: https://tr.wikipedia.org/wiki/Taylor_serisi adresinden alınmıştır.

URL-3.(27, Ekim 2023). Taylor Series : https://edumatth.weebly.com/uploads/1/3/1/9/13198236/taylor_series.pdf adresinden alınmıştır.

URL-4.(5, Mayıs 2024). Finite Diffrence: https://pythonnumericalmethods.studentorg.berkeley.edu/ adresinden alınmıştır.

URL-5.(5, Mayıs 2024). Numerical Differentiation: https://csw.uobaghdad.edu.iq/wp-content/uploads/sites/30/2019/09/Chapter1-2-merged.pdf adresinden alınmıştır.

URL-6.(9, Mayıs 2024). Bilgisayar Cebiri https://en.wikipedia.org/wiki/Computer_algebra adresinden alınmıştır.

URL-7.(9, Mayıs 2024).Maple: https://en.wikipedia.org/wiki/Maple_(software) adresinden alınmıştır.

URL-8.(9, Mayıs 2024). Mathematica: https://en.wikipedia.org/wiki/Wolfram_Mathematica adresinden alınmıştır.

URL-9.(9, Mayıs 2024). MATLAB: https://en.wikipedia.org/wiki/MATLAB adresinden alınmıştır.

URL-10.(9, Mayıs 2024). Maxima: https://en.wikipedia.org/wiki/Maxima_(software) adresinden alınmıştır.

URL-11.(9, Mayıs 2024). Sonlu_farklar_yöntemi: https://tr.wikipedia.org/wiki/Sonlu_farklar_y%C3%B6ntemi adresinden alınmıştır.

URL-12.(24, Mayıs 2024). Common Math: https://commons.apache.org/proper/commons-math/#:~:text=Commons%20Math%20is%20a%20library,programming%20language%20or%20Commons%20Lang. adresinden alınmıştır.

URL-13.(25, Mayıs 2024). Jscience https://jscience.org/ adresinden alınmıştır.

URL-14.(25, Mayıs 2024). Colt (libraries) https://en.wikipedia.org/wiki/Colt_(libraries) adresinden alınmıştır.

URL-15.(25, Mayıs 2024). ALGLIBhttps://www.alglib.net/adresinden alınmıştır.

URL-16.(25, Mayıs 2024).ND4J https://deeplearning4j.konduit.ai/nd4j/tutorials/quickstart adresinden alınmıştır.

URL-17.(25, Mayıs 2024). JAMA:https://en.wikipedia.org/wiki/JAMA adresinden alınmıştır.

URL-18.(1, Haziran 2024). Biçimsel Gramerler: https://medium.com/@yuanwhy/bnf-and-context-free-grammar-add2b9a63f19 adresinden alınmıştır.

Vatansever, F. A. H. R. İ. (2018). Sayısal Hesaplama ve Programlama.

Velychko, V., Stopkin, A., & Fedorenko, E. (2019). Use of computer algebra system maxima in the process of teaching future mathematics teachers. Information Technologies and Learning Tools, 69(1), 112. https://doi.org/10.33407/itlt.v69i1.2284

Yılmaz, C. (2019) Belirsiz İntegral Problemlerinin Çözümü İçin Genel Bir Yorumlayıcının Simgesel Hesaplama Yaklaşımları Kullanılarak Tasarımı Ve Gerçeklenmesi. K.T.Ü.,Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Trabzon.

Yu, S. (1997). Regular languages (pp. 41-110). Springer Berlin Heidelberg.

Xie, H., Zhao, H., & Fu, Q. (2009). Taylor expansion and its application in missile-borne SAR imaging. 2009 2nd Asian-Pacific Conference on Synthetic Aperture Radar, 426-430. https://doi.org/10.1109/APSAR.2009.5374291.

ÖZGEÇMİŞ

Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümünden mezun olmuştur. 2022 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünde Bilgisayar Bilimleri Anabilim dalında yüksek lisans eğitimine başlamıştır.

Yayınlar

1. Yılmaz, İ. U., & Pehlivan, H. (2024). Sonlu fark yaklaşımlarıyla sayısal türev ifadelerinin üretilmesi, değerlendirilmesi ve dokümantasyonu. In Proceedings of the 7th International Conference on Engineering Sciences (pp. 151-160). Ankara, Turkey.