T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

VAN DER POL DENKLEMİNİN OLASILIKSAL EVRİM YÖNTEMİ İLE ÇÖZÜMÜ İÇİN VERİ TABANI OLUŞTURMA

Yüksek Lisans Tezi

Tezi Hazırlayan
Mustafa ÖZTÜRK

İstanbul, 2024

T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

VAN DER POL DENKLEMİNİN OLASILIKSAL EVRİM YÖNTEMİ İLE ÇÖZÜMÜ İÇİN VERİ TABANI OLUŞTURMA

Yüksek Lisans Tezi

Tezi Hazırlayan
Mustafa ÖZTÜRK

Öğrenci No
2120003081

ORCID ID
0009-0008-2450-922X

Danışman
Dr. Öğr. Üyesi Coşar GÖZÜKIRMIZI

İstanbul, 2024

YEMİN METNİ

Yüksek Lisans Tezi Olarak Sunduğum **‘‘Van Der Pol Denkleminin Olasılıksal Evrim Yöntemi İle Çözümü İçin Veri Tabanı Oluşturma’’** başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını, patent ve telif haklarını ihlal edici bir davranışımın olmadığını belirtir ve bunu onurumla doğrularım. 04/10/2024

Mustafa ÖZTÜRK

T.C.
İSTANBUL BEYKENT ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ
TEZLİ YÜKSEK LİSANS SINAV TUTANAĞI

04/10/2024

Enstitümüz ***Bilgisayar Mühendisliği*** Anabilim Dalı ***Bilgisayar Mühendisliği*** Programı yüksek lisans öğrencilerinden ***2120003081*** numaralı ***Mustafa ÖZTÜRK'ün*** *"İstanbul Beykent Üniversitesi Lisansüstü Eğitim – Öğretim Yönetmeliği"*nin ilgili maddesine göre hazırlayarak, Enstitümüze teslim ettiği ***"Van Der Pol Denkleminin Olasılıksal Evrim Yöntemi ile Çözümü için Veri Tabanı Oluşturma"*** konulu tezini, Yönetim Kurulumuzun 14/05/2024 tarih ve 2024/22 sayılı toplantısında seçilen ve Taksim yerleşkesinde toplanan biz jüri üyeleri huzurunda, İstanbul Beykent Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 29. maddesinin 3. fıkrası gereğince aday tarafından savunulmuş ve sonuçta adayın tezi hakkında ***"OYBİRLİĞİ"*** ile ***"KABUL"*** kararı verilmiştir.

İşbu tutanak, 2 nüsha olarak hazırlanmış ve Enstitü Müdürlüğü'ne sunulmak üzere tarafımızdan düzenlenmiştir.

DANIŞMAN
Dr. Öğr. Üyesi Co*** GÖ***
(İstanbul Beykent Üniversitesi)

ÜYE
Dr. Öğr. Üyesi Ke*** Gök*** NA***
(İstanbul Beykent Üniversitesi)

ÜYE
Dr. Öğr. Üyesi Su*** ÜR***
(Bahçeşehir Üniversitesi)

Adı ve Soyadı : Mustafa ÖZTÜRK

Danışmanı : Dr. Öğr. Üyesi Coşar GÖZÜKIRMIZI

Derecesi ve Tarihi : Yüksek Lisans (Tezli), 2024

Alanı : Bilgisayar Mühendisliği

Anahtar Kelimeler : Van der Pol Osilatörü, Olasılıksal Evrim Yöntemleri, Veri Tabanı Yönetimi, Node.js

ÖZ

VAN DER POL DENKLEMİNİN OLASILIKSAL EVRİM YÖNTEMİ İLE ÇÖZÜMÜ İÇİN VERİ TABANI OLUŞTURMA

Bu tez çalışmasında, Van der Pol denkleminin olasılıksal evrim yöntemi kullanılarak çözümü için bir veri tabanı oluşturulmuştur. Van der Pol osilatörü, biyolojik, elektriksel ve mekanik sistemlerde yaygın olarak kullanılan doğrusal olmayan bir diferansiyel denklemdir. Van der Pol denkleminin türevli denklemlerin sayısal çözümlerinin sınanmasında sıklıkla kullanıldığı ve buradaki çalışmanın da sağ yanı multinom olan bütün denklem takımları için kolaylıkla genelleştirilebileceği vurgulanabilir. Bu çalışma, Van der Pol denkleminin sayısal çözümlerini elde etmek ve bu çözümleri saklamak amacıyla gerçekleştirilmiştir. Kullanıcılar, web tabanlı bir arayüz üzerinden parametreleri girer ve bu parametreler Node.js tabanlı bir sunucu tarafından işlenir. Sunucu, ilgili C++ kodunu çalıştırarak çözümleri hesaplar.

Elde edilen çözümler, JSON formatında işlenir ve MySQL veri tabanına kaydedilir. Bu yapı, verilerin düzenli bir şekilde saklanmasını ve gerektiğinde hızlıca erişilmesini sağlar. Çalışma kapsamında, verilerin kullanıcı dostu bir arayüz aracılığıyla sunulması sağlanmıştır. Bu bağlamda, çözüm sürecinin tamamı, kullanıcı etkileşiminden sonuçların sunumuna kadar ayrıntılı bir şekilde ele alınmıştır. Çalışmanın bulguları, Van der Pol denkleminin çözümlerinin verimli bir şekilde saklanması ve analiz edilmesi açısından önemlidir.

Name and Surname : Mustafa ÖZTÜRK

Supervisor : Asst. Prof. Dr. Coşar GÖZÜKIRMIZI

Degree and Date : Master's (Thesis), 2024

Major : Computer Engineering

Keywords : Database Management, Probabilistic Evolution Methods, Node.js, Van der Pol Oscillator

ABSTRACT

CREATING A DATABASE FOR THE SOLUTION OF VAN DER POL EQUATION WITH PROBABILISTIC EVOLUTION METHOD

In this thesis, a database was created for solving the Van der Pol equation using the probabilistic evolution method. The Van der Pol oscillator is a nonlinear differential equation commonly used in biological, electrical, and mechanical systems. It can be emphasized that the Van der Pol equation is frequently used in testing numerical solutions of differential equations and that this study can be easily generalized for all equation systems with multinomial right-hand sides. This study was carried out to obtain numerical solutions of the Van der Pol equation and to store these solutions effectively. Users enter parameters through a web-based interface, and these parameters are processed by a Node.js-based server. The server runs the relevant C++ code to compute the solutions.

The obtained solutions are processed in JSON format and stored in a MySQL database. This structure ensures that the data is stored in an organized manner and can be quickly accessed when needed. The study also provides a user-friendly interface for presenting the data. In this context, the entire solution process, from user interaction to the presentation of results, is addressed in detail. The findings of the study are significant for the efficient storage and analysis of the solutions to the Van der Pol equation.

İÇİNDEKİLER

Sayfa No.

TABLOLAR LİSTESİ

Sayfa No.

ŞEKİLLER LİSTESİ

Sayfa No.

KISALTMALAR

API : Application Programming Interface (Uygulama Programlama Arayüzü)

CORS : Cross-Origin Resource Sharing (Kaynaklar Arası Paylaşım)

CSS : Cascading Style Sheets (Basamaklı Stil Sayfaları)

GPU : Graphics Processing Unit (Grafik İşlem Birimi)

HTML : Hypertext Markup Language (Hipermetin İşaretleme Dili)

HTTP : Hypertext Transfer Protocol (Hipermetin Transfer Protokolü)

HTTPS : Hypertext Transfer Protocol Secure (Güvenli Hipermetin Transfer Protokolü)

I/O : Input/Output (Giriş/Çıkış)

ID : Identifier (Tanımlayıcı)

IP : Internet Protocol (İnternet Protokolü)

JSON : JavaScript Object Notation (JavaScript Nesne Gösterimi)

NoSQL : Not Only SQL (Sadece SQL Değil)

NPM : Node Package Manager (Düğüm Paket Yöneticisi)

ODE : Ordinary Differential Equation (Sıradan Türevli Denklem)

RAM : Random Access Memory (Rastgele Erişimli Bellek)

RDBMS : Relational Database Management System (İlişkisel Veri Tabanı Yönetim Sistemi)

REST : Representational State Transfer (Temsili Durum Transferi)

SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)

UI : User Interface (Kullanıcı Arayüzü)

URL : Uniform Resource Locator (Tek Biçimli Kaynak Konumlandırıcı)

XML : eXtensible Markup Language (Genişletilebilir İşaretleme Dili)

SÖZLÜK

API Endpoint: Bir API'nin belirli bir işlevi gerçekleştirmek üzere tasarlanmış ve erişilebilir olan özel adresi veya URL'si.

Asenkron İşlem: Bir işlemin tamamlanmasını beklemeden diğer işlemlerin devam etmesine olanak tanıyan programlama tekniği.

Async/Await: JavaScript'te asenkron işlemleri daha okunabilir ve yönetilebilir hale getiren bir sözdizimi yapısı.

Backend: Bir web uygulamasının sunucu tarafında çalışan, veri işleme ve depolama gibi işlevleri yerine getiren bölümü.

Beam Search: Olası çözümlerin sınırlı bir alt kümesini araştırarak en iyi çözümü bulmayı amaçlayan bir arama algoritması.

Bootstrap: Duyarlı ve mobil öncelikli web siteleri geliştirmek için kullanılan açık kaynaklı bir CSS çerçevesi.

C++: Yüksek performanslı hesaplamalar için kullanılan genel amaçlı bir programlama dili.

Callback Fonksiyonu: Başka bir fonksiyona argüman olarak geçirilen ve belirli bir olay veya koşul gerçekleştiğinde çağrılan fonksiyon.

Çocuk İşlem (Child Process): Bir ana işlem tarafından başlatılan ve kontrol edilen bağımsız bir işlem.

DataTables: Büyük veri setlerini tablo halinde göstermek ve yönetmek için kullanılan bir jQuery eklentisi.

Derleme (Compilation): Kaynak kodun makine diline çevrilmesi işlemi.

Diferansiyel Denklem: Bir veya daha fazla bağımsız değişkene göre bir fonksiyonun türevlerini içeren denklem.

Doğrusallık (Linearity): Bir sistemin veya denklemin, bileşenlerinin toplamı veya çarpımı altında aynı şekilde davranma özelliği. Doğrusal sistemler ve denklemler, süperpozisyon (girdilerin etkilerinin toplamı) ve homojenlik (girdinin bir sabitle çarpılması) ilkelerine uyar.

Erişim Kontrolü: Kullanıcıların sistem kaynaklarına erişimini düzenleyen ve yöneten güvenlik mekanizması.

Express.js: Node.js için web ve mobil uygulamalar geliştirmek için kullanılan minimal ve esnek bir web uygulama çerçevesi.

Foreign Key: Bir veri tabanı tablosundaki bir alanın, başka bir tablodaki birincil anahtara referans verdiği ilişkisel veri tabanı kavramı.

Frontend: Bir web uygulamasının kullanıcı tarafında çalışan, kullanıcı arayüzünü oluşturan ve kullanıcı etkileşimlerini yöneten bölümü.

Git (Versiyon Kontrol Sistemi): Yazılım geliştirme sürecinde kod değişikliklerini izlemek ve yönetmek için kullanılan dağıtık versiyon kontrol sistemi.

Hata Ayıklama (Debugging): Yazılımdaki hataları tespit etme ve düzeltme süreci.

HTTP Metotları (GET, POST, PUT, DELETE): Web sunucuları ile iletişim kurmak için kullanılan standart istek yöntemleri.

İlişkisel Veri Tabanı: Veriler arasında ilişkiler kurularak organize edilen ve yönetilen veri tabanı türü.

İndeksleme (Veri Tabanı): Veri tabanı sorgularının performansını artırmak için kullanılan veri yapısı.

JSON Format: Veri alışverişi için kullanılan, insanlar tarafından okunabilir ve yazılabilir bir veri formatı.

Kimlik Doğrulama: Bir kullanıcının veya sistemin, belirtilen kimliğe sahip olduğunu doğrulama süreci.

Kullanıcı Arayüzü (UI): Kullanıcıların bir sistemle etkileşime girmesini sağlayan görsel ve işlevsel bileşenler bütünü.

Limit Döngüsü (Limit Cycle): Nonlineer dinamik sistemlerde görülen, kapalı ve izole edilmiş periyodik yörünge.

Makine Öğrenmesi: Bilgisayar sistemlerinin verilerden öğrenerek performanslarını artırma yeteneği.

Middleware: Yazılım uygulamalarında, işletim sistemi ve uygulama yazılımı arasında aracılık yapan yazılım katmanı.

MySQL: Açık kaynaklı bir ilişkisel veri tabanı yönetim sistemi.

Node.js: Sunucu tarafı JavaScript çalıştırmak için kullanılan açık kaynaklı, platformlar arası bir çalışma zamanı ortamı.

Nonlineer Dinamik Sistemler: Zaman içinde doğrusal olmayan davranışlar gösteren karmaşık sistemler.

Nonlineer Terim: Bir denklem veya sistemde doğrusal olmayan ilişkiyi ifade eden terim.

NoSQL Veri Tabanı: İlişkisel modelin dışında kalan, genellikle büyük veri setleri için kullanılan veri tabanı türü.

Olasılıksal Evrim Yöntemi: Sağ yanı multinom olan diferansiyel denklem takımlarının seri çözümünü üreten bir yöntem.

Olay Tabanlı Programlama: Programın akışının kullanıcı eylemleri veya sistem olayları tarafından yönlendirildiği programlama paradigması.

Ölçeklenebilirlik: Bir sistemin artan iş yüküyle başa çıkabilme ve büyüyebilme yeteneği.

RESTful API: Web hizmetleri arasında veri alışverişi için kullanılan bir mimari stil.

Runge-Kutta Yöntemi: Diferansiyel denklemlerin sayısal çözümü için kullanılan bir yöntem.

Shell Script: Unix ve Linux sistemlerinde komut satırı üzerinden otomatik işlemler yapmak için kullanılan betik dili.

SQL Sorgusu: İlişkisel veri tabanlarında veri çekmek, güncellemek veya silmek için kullanılan yapılandırılmış sorgu dili ifadesi.

Şifreleme: Bilgiyi yetkisiz erişime karşı korumak için okunamaz hale getirme işlemi.

Uzay Genişletme Tekniği: Diferansiyel denklemlerin çözümünde kullanılan, denklem sistemini daha çözülebilir bir formata dönüştürme yöntemi.

Van der Pol Osilatörü: Nonlineer bir diferansiyel denklem sistemi olup, biyolojik ve elektronik sistemlerde osilasyonları modellemek için kullanılır.

Veri Analizi: Ham verilerden anlamlı bilgiler çıkarmak için kullanılan teknikler ve süreçler bütünü.

Veri Bütünlüğü: Verilerin doğruluğunu, tutarlılığını ve güvenilirliğini koruma özelliği.

Veri Görselleştirme: Verileri grafikler, haritalar veya diğer görsel formatlarda temsil etme süreci.

Veri Madenciliği: Büyük veri setlerinden anlamlı örüntüler ve ilişkiler çıkarma süreci.

Veri Normalizasyonu: Veri tabanı tasarımında veri tekrarını azaltmak ve veri bütünlüğünü artırmak için kullanılan bir süreç.

Veri Önbelleğe Alma: Sık kullanılan verileri hızlı erişim için geçici olarak saklama tekniği.

Veri Tabanı Şeması: Bir veri tabanının yapısını, tablolarını ve aralarındaki ilişkileri tanımlayan plan.

Veri Tabanı Yönetim Sistemi: Verilerin organize edilmesini, depolanmasını ve yönetilmesini sağlayan yazılım sistemi.

Web Sunucusu: İnternet üzerinden web sayfalarını ve diğer içerikleri kullanıcılara sunan yazılım veya donanım.

Yetkilendirme: Kimliği doğrulanmış bir kullanıcıya belirli kaynaklara erişim izni verme süreci.

Yük Dengeleme: Ağ trafiğini veya işlem yükünü birden fazla sunucu veya kaynak arasında dağıtma işlemi.

GİRİŞ

Modern mühendislik ve fizik problemlerinin çözümünde kullanılan sayısal yöntemler, özellikle karmaşık sistemlerin modellemesinde kritik bir role sahiptir. Van der Pol osilatörünün dinamiklerini modellemek ve bu dinamikleri olasılıksal evrim yöntemleri kullanarak çözmek, bu çözümlerin performansının Node.js tabanlı bir web platformu aracılığıyla nasıl işlenebileceğini ve MySQL veri tabanında nasıl organize edilebileceğini ele almak, bu tezin odak noktasını oluşturmaktadır.

Tezin temel amacı, Van der Pol denkleminin çözümlerini farklı başlangıç değerleri ve parametreler altında hızlı ve etkin bir şekilde üretmek, bu çözümleri analiz etmek ve sonuçları sistematik bir şekilde veri tabanında saklamaktır. Bu amaç doğrultusunda, bir dizi sayısal ve analitik yöntem kullanılarak denklemin çözümleri elde edilirken, bir web uygulaması üzerinden kullanıcı dostu bir arayüz sağlanmaktadır. Bu arayüz, kullanıcıların denklemin parametrelerini kolayca değiştirerek çözümleri gerçek zamanlı olarak gözlemlemelerine olanak tanır.

Tezin yapısı, okuyucuları adım adım araştırmanın her aşamasına yönlendirecek şekilde tasarlanmıştır. İlk olarak, Van der Pol osilatörü ve olasılıksal evrim yöntemlerinin teorik temelleri anlatılacaktır. Daha sonra, kullanılan yazılım ve donanım gereksinimlerinden, sistem tasarımına, veri toplama ve işleme yöntemlerine kadar detaylı bilgiler sunulacaktır. Sonuç bölümünde, elde edilen bulgular değerlendirilirken, çalışmanın sınırları ve potansiyel iyileştirmeler için öneriler de dile getirilmektedir. Tezin sonunda ise, elde edilen bilgiler ışığında gelecekteki çalışmalar için yön gösterici olacak önerilere yer verilmektedir.

Araştırma Sorusu ve Amaçları

Bu çalışmanın temel araştırma sorusu, Van der Pol osilatörünün olasılıksal evrim yöntemleri kullanılarak çözümünün nasıl elde edilebileceği ve bu çözümlerin sayısal olarak nasıl işlenebileceği üzerinedir. Araştırmanın ana amacı, bu denklemin sayısal çözümlerinin çeşitli yöntemlerle üretilmesi, bu çözümlerin etkin bir şekilde işlenmesi ve bu çözümleri veri tabanında düzenli bir şekilde saklamaktır.

Elde edilen verilerin, kullanıcıların erişimine açık bir şekilde MySQL veri tabanında düzenli ve verimli bir biçimde saklanması ve veri tabanı tasarımının, veri toplama ve analiz süreçlerini destekleyecek şekilde optimize edilmesi de araştırmanın önemli bir amacıdır. Başlangıç değer problemi için daha önce çözüm elde edilmiş ise, çözümün tekrarlanmadan yalnızca veri tabanından çekilmesi vurgulanabilir. Kullanıcıların, denklem parametrelerini kolayca değiştirebilecekleri ve çözümleri gerçek zamanlı olarak gözlemleyebilecekleri interaktif bir web arayüzü sunulması, araştırmanın kullanıcı dostu bir yapıya sahip olmasını sağlamaktadır.

Bu amaçlara ulaşmak, Van der Pol denklemlerinin sayısal çözümlerinin daha iyi anlaşılmasını ve bu denklemlerin ürettiği sonuçların veri tabanında saklanmasını sağlarken, bilimsel ve teknik alanlarda uygulanabilirliğini artıracaktır.

Tezin Önemi

Van der Pol denkleminin çözümü, matematiksel modelleme ve mühendislik uygulamalarında önemli bir yere sahiptir. Bu denklem, biyoloji, elektrik mühendisliği, mekanik ve fizik alanlarında karşılaşılan nonlineer dinamiklerin anlaşılmasında kullanılmaktadır. Özellikle, biyolojik osilasyonların ve elektrik devrelerindeki röle osilasyonlarının modellenmesinde temel bir araç olarak ön plana çıkar. Van der Pol osilatörünün bu geniş uygulama alanı, denklemin çözümlerinin daha derinlemesine incelenmesini gerektirir, çünkü bu çözümler sistemlerin davranışlarını anlamada kritik rol oynar.

Bu tez, Van der Pol denkleminin çözümlerini olasılıksal evrim yöntemiyle üreterek ve bu çözümleri bir veri tabanında organize ederek, bu alanda mevcut bilgi birikimine katkılarda bulunmayı hedeflemektedir. Ayrıca, geliştirilen web tabanlı uygulama, kullanıcılara interaktif bir ortamda deneyler yapma imkânı sunarak, eğitim ve araştırma faaliyetlerine katkıda bulunmayı hedeflemektedir.

Sonuç olarak, bu tez, Van der Pol denklemlerinin analizi ve uygulamaları konusunda alana katkı sağlamayı ve bu tür bilgilerin daha geniş bir kitle tarafından kullanımını teşvik etmeyi amaçlamaktadır. Bu çalışmanın sonuçları, öğretim ve pratik uygulamalar için yeni perspektifler sunmakta ve doğrusal olmayan sistemlerin daha

iyi anlaşılmasına bir web arayüz ile değişik parametrelerde hızlı sonuçlar üretmesi açısından katkı sağlamayı amaçlamaktadır.

Tezin Yapısı

Bu tez, "Van Der Pol Denkleminin Olasılıksal Evrim Yöntemi ile Çözümü İçin Veri Tabanı Oluşturma" başlığı altında, Van der Pol osilatörünün çözümlerini olasılıksal evrim yöntemi kullanarak elde etmeyi ve bu çözümleri bir veri tabanında organize etmeyi detaylı bir şekilde ele alır. Tez, problemi çözme sürecinde izlenen metodolojiyi, elde edilen sonuçları ve bu sonuçların analizini kapsamlı bir şekilde sunmayı amaçlamaktadır. Doküman, aşağıdaki bölümlerden oluşmaktadır:

Giriş:

Araştırma sorusunu, tezin amacını ve önemini tanıtır. Ayrıca, çalışmanın genel yapısını ve alana sağlayacağı katkıları özetler.

Teorik Çerçeve:

Van der Pol osilatörü, olasılıksal evrim yöntemleri ve veri tabanı sistemlerinin teorik temelleri bu bölümde işlenir. Burada, kullanılan matematiksel modeller ve algoritmalar hakkında detaylı bilgiler verilir.

Sistem Tasarımı ve Geliştirme:

Çalışmada kullanılan yazılım ve donanım gereksinimleri, sistem mimarisi ve modellenen denklemlerin detayları bu bölümde açıklanır. Ayrıca, veri tabanı tasarımı ve sistem bileşenlerinin entegrasyon süreçleri ele alınır.

Uygulama ve Veri Toplama:

Van der Pol denkleminin çözümlerinin nasıl elde edildiği, bu verilerin nasıl toplandığı ve işlendiği, bu bölümde sunulur.

Verilerin İşlenmesi ve Analizi:

Elde edilen verilerin analizi, veri işleme teknikleri ve bu verilerin veri tabanına nasıl entegre edildiği konuları bu kısımda işlenir.

Web Uygulaması ve Kullanıcı Arayüzü:

Kullanıcıların denklemin parametrelerini değiştirebilecekleri, çözümleri gerçek zamanlı olarak inceleyebilecekleri web tabanlı uygulama ve arayüz tasarımı bu bölümde detaylandırılır.

Sonuçlar ve Değerlendirme:

Araştırma sonuçlarının sunumu, elde edilen bulguların değerlendirilmesi ve çalışmanın öne çıkan yönleri bu bölümde ele alınır. Ayrıca, sonuçların akademik ve pratik etkileri tartışılır.

Öneriler ve Gelecek Çalışmalar:

Çalışmanın potansiyel geliştirme alanları, öneriler ve gelecekte yapılacak araştırmalar için önerilen yönlendirmeler bu sonuç bölümünde sunulur.

1. TEORİK ÇERÇEVE (KURAMSAL ÇERÇEVE)

Bu bölümde çalışmanın uzbilimsel (matematiksel) temellerini oluşturan bilimsel yazındaki (literatürdeki) bazı makale ve bildirilere değinilecektir. Türevli denklemler ve bu denklemlerin sayısal çözümleri hakkında birçok çalışma bulunmaktadır. Bu çalışmaları sınıflandırmak ve değerlendirmek oldukça kapsamlı bir iş olmakla birlikte, bu tezin ilgi alanı dışındadır. Burada yalnızca türevli denklem kavramına değinilecek, yazından bazı örnekler verilecek ve doğrudan olasılıksal evrim kuramına taban oluşturan veya ilintili olan makale ve bildirilere değinilecektir. Yazından örnekler verilirken de çok ön planda olan yayınlar yerine, önemli olmasına karşın daha kenarda kalmış yayınlara da değinilecektir. Bunun nedeni türevli denklem çözümü üzerine birçok derleme (İng. survey) makalesi bulunmaktadır. Bu makalelerdeki kavramların burada yinelenmesine gerek görülmemektedir.

Yazın özetine girmeden önce türevli denklem kavramı hakkında bazı temel olguları olabildiğince yalın biçimde gündeme getirmek önem kazanmaktadır. Bir bilgisayar mühendisi gözünden, işleç kuramı (İng. operator theory) ve doğrusal cebir (İng. linear algebra) kavramlarının ayrıntılarına girmeden, türevli denklem olgusunu anlatmak önem kazanmaktadır. Buradaki anlatım matematiksel bir anlatım değil, işin özünü anlamaya yönelik bir anlatım olacaktır.

Daha sonraki alt bölümlerde ise sıradan türevli denklem çözümü bağlamındaki çalışmalara, uzay genişletimine ve olasılıksal evrim kuramına ayrı ayrı değinilecektir.

Devingen dizgeler (dinamik sistemler), türev olgusunun tek uygulama alanı değildir, ama en öne çıkan uygulama alanlarından biridir. Bir işlevin (fonksiyonun) zamana bağlı olduğu öngörülmekte ve belirli bir anda hangi değeri alacağı gündeme getirilmektedir. Eğer işlevin ilgilendiğimiz bütün anlarda değerini biliyorsak, o işlev hakkında istediğimiz her şey elimizde zaten bulunmaktadır. Birçok durumda ise daha karışık yapılar söz konusudur. Zamana göre türev olgusu, bir işlevin ne kadar çabuk değer değiştirdiğinin ölçüsüdür. Bir boyutta devinen (hareket eden) bir parçacık düşünülürse, bu parçacığın herhangi bir anda nerede olacağının bilinmesi önem kazanır. Bazen elimizde bu bilgi yoktur ama parçacığın yerdeğişiminin, yönlü hızının (İng. velocity) ve ivmesinin arasındaki ilişki bilinmektedir. Buradan parçacığın istenilen andaki yerinin belirlenmesi için türevli denklem çözümü gereklidir.

Yerdeğişiminin ne kadar çabuk değiştiğini yönlü hız gösterir. Yönlü hızın ne kadar çabuk değiştiğini ise ivme gösterir.

Türevli denklem çözümü denildiğinde, aslında bahsedilen olgu başlangıç değer sorunu çözümüdür. Başlangıç değer sorunu türevli denklem takımından ve karşılık gelen başlangıç değerlerinden oluşur. Başlangıç değerinin anlamı zamanın 0 olarak alındığı andaki işlev değeridir. Belirtik (açık, İng. explicit) denklemlerde en yüksek mertebe türev, eşitliğin sol yanında bulunur, dolayısıyla en yüksek mertebe türevi ayrıştırmak için bir denklem çözümü gerekmez. Bir toplamsal terimdeki türevin mertebesi (İng. order) o terimde kaç kere türev alındığını gösterir. Bir denklemin mertebesi ise o denklemde herhangi bir toplamsal terimde alınan en yüksek mertebe türevdir. Başlangıç değer sorununda en yüksek türev n ise işlevin kendisinden (n-1)inci türevine dek işlevlerin başlangıç koşulu verilir.

Başlangıç değer sorunu dışında kıyı (sınır) değer sorunları da önem kazanır. Burada değişik anlardaki işlev değerleri bilinmektedir. Bu tür sorunlardaki çözüm yollarından biri bu yapıları önce başlangıç değer sorununa dönüştürmektir.

Ayrıca, bu anlatımda hep zamana göre türev vurgulandı. Hep zamana göre türevin söz konusu olduğu yapılar sıradan türevli denklemlerdir (adi diferansiyel denklemler, İng. ordinary differential equations). Başka işlevlere göre türevlerin de olduğu yapılar ise göre türevli denklemlerdir (kısmi diferansiyel denklemler, İng. partial differential equations). Göre türevli denklem çözümündeki başat yollardan biri de önce bu denklemleri önce sıradan türevli denklemlere dönüştürmektir.

Yukarıdaki anlatım türevli denklemleri, biraz da aşırı yalınlaştırarak, bir bilgisayar mühendisinin gereksinim duyacağı biçimde gündeme getirmektedir. Bilimsel yazında bu tür bir anlatıma pek de rastlanılmamaktadır. Bu yüzden yazına gönderimde (atıf) bulunulmamıştır. İşin matematiğine daha çok ilgisi olan okuyucu, üniversite lisans öğrencilerine yönelik temel diferansiyel denklemler kitaplarını inceleyerek daha derin bilgi edinebilir.

1.1. Diferansiyel Denklemler ve Van Der Pol Osilatörü

Diferansiyel denklemler, matematiksel ifadelerdir ve bir değişkenin fonksiyonunu ve onun türevlerini içermektedir. Birçok fiziksel, mühendislik, ekonomi

ve diğer bilim alanlarında doğal olayları ve sistemleri modellemek için yaygın olarak kullanılmaktadır. Diferansiyel denklemlerin kökenleri eski dönemlere dayanmakta ve matematiksel analiz tarihinde önemli bir rol oynamaktadır.

Diferansiyel denklemlerin ortaya çıkışıyla ilgili temel tarihî kaynaklardan biri, 17. yüzyılda yaşamış olan Isaac Newton ve Gottfried Wilhelm Leibniz'in (Kalaycıoğlu, 2017, s. 64-68) diferansiyel ve integral hesabı üzerine çalışmalarıdır. Özellikle Newton, hareketin matematiksel analizi için diferansiyel denklemleri kullanmış ve bu denklemlerin fiziksel olayları modellemede güçlü bir araç olduğunu göstermiştir (Upton, 2004). Leibniz'in çalışmaları da bu alanda büyük bir etki yaratmış ve integral ve diferansiyel hesaplamaları daha sistemli bir şekilde ele almıştır (Öztürk, 2017, s. 576). Daha sonraki dönemlerde, diferansiyel denklemler matematiksel analizin ve fizik biliminin temel taşlarından biri haline gelmiştir. Özellikle 18. ve 19. yüzyıllarda, matematikçiler ve fizikçiler diferansiyel denklemleri çeşitli doğal ve fiziksel süreçleri modellemek için kullanmışlardır (Sevimli, 2016, s.154-171). Örneğin, sıcaklık dağılımı, elektrik akımları, titreşimler ve difüzyon gibi olayları incelemek için diferansiyel denklemler yaygın bir şekilde kullanılmıştır.

Bu bağlamda modern diferansiyel denklemler teorisi, 20. yüzyılın başlarındaki matematiksel gelişmelerle önemli ölçüde genişlemiştir. Bu dönemde, diferansiyel denklemlerin çözümlerinin varlığı, benzersizliği ve kararlılığı gibi temel konular üzerinde daha derin bir anlayış geliştirilmiştir. Bu çalışmalar, diferansiyel denklemlerin matematiksel teorisini daha sağlam hale getirmiş ve uygulamalı bilimlerdeki pratik uygulamalarını daha da güçlendirmiştir (Sevimli, 2016, s.154-171).

Van der Pol Osilatörü ise zamanla değişen bir sistemin dinamiğini ifade eden bir diferansiyel denklem takımıdır (Van der Pol, 1927). Genellikle, bu osilatör bir elektrik devresi içindeki bir elektronik tüpün (triode gibi) davranışını modellemek için kullanılır (Strogatz, 1994). Van der Pol osilatörü, doğrusal olmayan sistemlerin davranışını anlamak için bir model olarak öne çıkar ve özellikle kaotik davranışların analizinde kullanışlıdır.

Matematiksel olarak, Van der Pol osilatörü aşağıdaki diferansiyel denklemle ifade edilir:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\left(\frac{dx}{dt}\right) + x = 0$$

x: sistemin durumu (pozisyonu)

t: zaman

μ: sönümleme parametresi (genellikle pozitif bir değer)

d^2x/dt^2: x'in zamana göre ikinci türevi (ivme)

dx/dt: x'in zamana göre birinci türevi (hız)

Van der Pol Osilatörü, ilk olarak Hollandalı mühendis ve fizikçi Balthasar Van Der Pol (1889-1959) tarafından belirli koşullarda negatif direnç gösteren lambalı radyolarda güç yükselticilerini modellemek için geliştirilmiştir (Van der Pol, 1927, s. 978–992). Bu model, elektrik mühendisliği, fizik, biyoloji ve diğer birçok alanda dalga formasyonları ve osilasyonlar üzerine yapılan çalışmalarda kullanılmıştır (Çakır, vd., 2021, s.81-91).

Van der Pol, elektrik devrelerindeki osilasyonları (dalgalanmaları) incelediğinde, belirli bir enerji seviyesine ulaşıldığında osilasyonların (dalgalanmanın) sabit bir amplitüde (genlik) sahip olduğunu fark etmiştir. Bu fenomen, o dönemdeki radyo alıcılarının tasarımı ve işleyişinde önemli bir yer tutmuştur. Van der Pol denkleminin kendine has özelliği, doğrusal olmayan bir terime sahip olmasıdır. Bu doğrusal olmayan terim, sistemin enerji kazanmasını veya kaybetmesini sağlar, bu da denklemin birçok fiziksel ve mühendislik sistemlerinde uygulanmasına olanak tanır. Örneğin, elektronik devrelerdeki osilasyonlar, biyolojik ritimlerdeki düzenli döngüler ve hatta astrofiziksel nesnelerin dinamikleri bu denklemin uygulama alanları arasındadır.

Bu denklemin tarihsel gelişimi, matematiksel modelleme ve analiz tekniklerinin ilerlemesiyle paralellik göstermiştir. Özellikle, Van der Pol denkleminin çözümüne yönelik sayısal yöntemlerin geliştirilmesi, bilgisayarların ve hesaplama teknolojilerinin ilerlemesiyle hız kazanmıştır. Bu sayede, denklemin daha karmaşık ve gerçekçi modellerinin incelenmesi mümkün hale gelmiştir.

Matematiksel olarak, Van der Pol denklemi şu şekilde ifade edilir:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\left(\frac{dx}{dt}\right) + x = 0$$

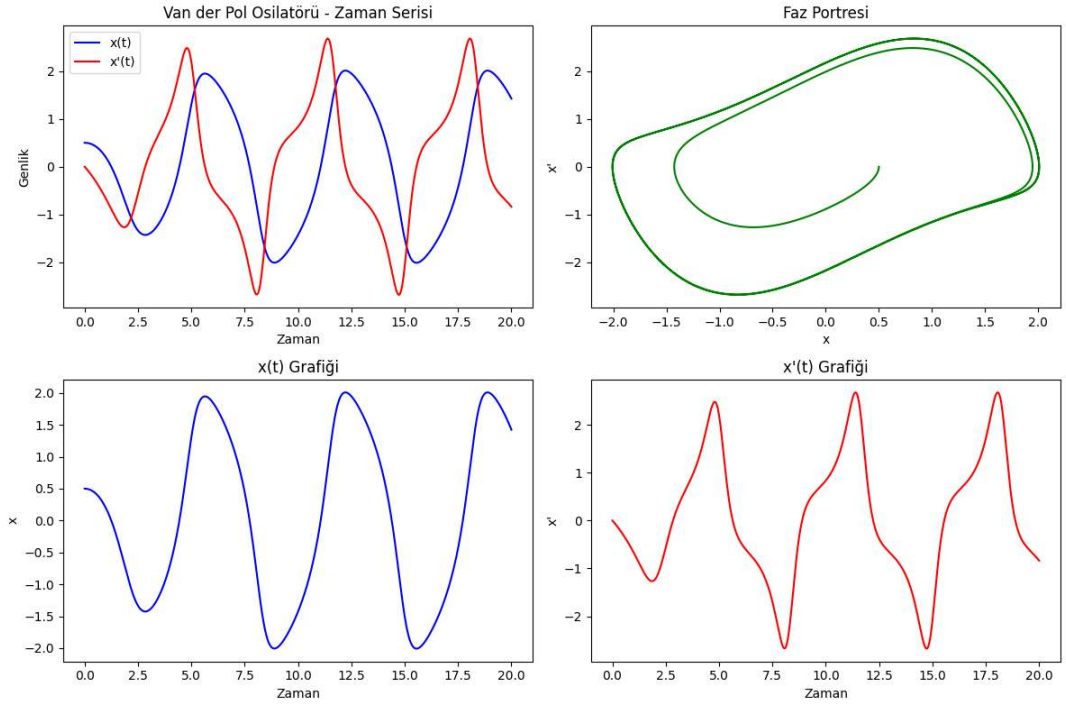
$$\frac{dx}{dt} = \mu\left(x - \frac{x^3}{3} - y\right)$$

$$\frac{dy}{dt} = \frac{x}{\mu}$$

Burada verilen ikinci mertebeden diferansiyel denklem, iki adet birinci mertebeden diferansiyel denklem olarak da yazılmıştır. Bu iki denklem, sistemin davranışını tanımlamak için kullanılabilir. Denklemde x osilatörün durumunu, μ ise sistemin doğrusal olmayan direncini temsil eder. μ parametresi, osilasyonun şiddetini ve döngüsünü kontrol eder. μ değeri arttıkça, sistem daha güçlü bir limit cycle davranışına sahip olur.

Van der Pol osilatörünün çözümü genellikle sayısal yöntemlerle yapılır çünkü çoğu durumda analitik bir çözüm bulmak mümkün değildir. Bu osilatör, elektronik osilatörlerin, kalp atış ritimlerinin ve nöronların ateşleme modellerinin anlaşılmasında önemli bir rol oynamaktadır. Bu çalışmada, başlangıç koşulu olarak x ve y değerleri sırasıyla 1/2 ve 1/2 olarak verilmiştir.

Örnek denklem çözümü ve çıktısı için Python programlama dili kullanılarak tabloda belirtilen sonuçlar elde edilmiştir. Şekil 1'deki çizimlerde başlangıç koşulu 1/2 ve 0 olarak alınmıştır.



Şekil 1. Van der Pol Osilatörü

Kaynak: Yazar tarafından oluşturulmuştur.

Sıradan türevli denklem çözümü denildiğinde, aslında vurgulanan olgu belirtik sıradan türevli denklem takımlarının başlangıç değer sorununun çözümüdür. Bu cümle uzun olduğu için, kısaca sıradan türevli denklem çözümü diye de söylenmektedir. Denklem takımı ise, bir veya daha çok denklemi içerir. Ama yine söylem olarak, denklem dendiğinde, bahsedilen olgu denklem takımı olabilmektedir.

Bu bölümdeki amaç, sıradan türevli denklem çözümü yöntemleri ilgili kapsamlı bir derleme yapmak değildir. Bilimsel yazındaki derleme makaleleri bile çoğunlukla belirli ortak özellikleri olan yöntemlere odaklanmakta, bütüncül bir çabadan kaçınmaktadır. Bütüncül bir çaba yerine, olasılıksal evrim yöntemleri açısından önemli olabilecek olgulara önem verilmiş ve bazı temel çözüm yöntemlerinden bir seçki oluşturulmuştur. Seçkiyi oluştururken derleme makalelerindeki ve kitaplardaki yapıları yinelemekten kaçınılmış, tez yazarı ve danışmanı tarafından ön planda olması gerektiği düşünülen ama o kadar ön planda olmayan çalışmalara da değinilmiştir.

Bilimsel uygulamalarda ve mühendislik uygulamalarında birçok yöntem ayrıklaştırmaya dayanır. İşlevlerin sürekli yapıları ile çalışmak yerine, onların belirli noktalarındaki değerler ile çalışmak sıklıkla izlenen yoldur. Ayrıklaştırımın

olumsuzlukları da iyi bilinmektedir. Dolayısıyla, ayrıklaştırım tabanlı yöntemlerin daha yüksek doğruluklu sonuçlar vermesini sağlama amacına odaklanan birçok çalışma mevcuttur (Kovalnogov, vd., 2022; Kovalnogov, vd., 2023). Bu çalışmalarda, aslında ayrıklaştırımın doğası gereği oluşan sayısal yanılgı (İng. error), daha yüksek duyarlıklı aritmetik kullanılarak azaltılmaya çalışılmaktadır.

Ilyashenko (1969, s. 365-378) sağ yanı iki çokçokterimlinin oranı olan yapılara odaklanmaktadır. Bu yapı oldukça özel bir yapı olarak görünse de aslında birçok denklem takımı bu yapıda yeniden yazılabilir. Çokçokterimli sözcüğü İngilizce multinomial sözcüğüne karşılık olarak kullanılmaktadır. Parker-Sochacki yöntemleri, yine ayrıklaştırmadan kaçınır. Bu yöntemlerde de olasılıksal evrim kuramında olduğu gibi Taylor açılımı ve Picard yinelemesi ile Taylor açılımının yeniden yazımı söz konusudur (Sochacki, 2010, s. 441-450). Carothers ve diğerleri (2012, s. 175-185) ise bilimsel yazında otomatik türevleme (İng. automatic differentiation) başlığı altında anılan yöntemlerin Taylor açılımı tabanlı yöntemlerle ilişkisini vurgular. Otomatik türevleme için ise Eberhard ve Bischof (1999, s. 717-731) öne çıkan bir çalışmadır.

Barrio (2005, s. 525-545) Taylor açılımı tabanlı yöntemlerin başarımına (İng. performance) değinmektedir. Taylor açılımı tabanlı yöntemlerin gerçeklendiği birçok bilgisayar programı da bulunmaktadır. Jorba ve Zou (2005, s. 99-117) bu tür başarılı çabalara örnek olarak gösterilebilir. Ayrıca, bu yöntemlerin çok yüksek duyarlıkta çözüm üretimi için gündeme getirilebileceği de birçok çalışmada vurgulanmıştır (Barrio, 2005; Bailey vd., 2012, s. 10106-10121).

Bu tezin omurgasını oluşturan, çözümlerin bir veri tabanında tutulup, daha sonra istenildiğinde veri tabanından sonuçların getirilmesi düşüncesi de eskilere dayanmaktadır (Barrio vd., 2015, s. 76-83). İnternet hızının, veri tabanı sorgularının yanıtlanma hızının ve en önemlisi veri depolama olanaklarının artması, bu yaklaşımı günümüzde daha geçerli ve anlamlı konuma getirmektedir.

Bir başka ilginç konu ise, kombinatorik (İng. combinatorics) olarak anılan alan ile türevli denklem çözümü arasındaki ilişkidir. Stanley (1980, s. 175-188) bu bağlamdaki çalışmalara örnektir. Sonlu ve sayılabilir bir kümeden seçme ve sıralama, Taylor açılımının işlenebilmesinde önem kazanmaktadır.

Bruno (1997, s. 471-491) Taylor açılımı bağlamındaki çözüme geometrik bir bakış açısı getirmektedir. Bostan ve diğerleri (2007, s. 1012-1021) de olasılıksal evrim kuramına benzer bazı olguları içermesi bakımından önemlidir. Yine oldukça geniş bir denklem takımı ailesinin çözümü için kullanılabilen Adomian ayrıştırımı tabanlı yöntemler de bu bağlamda bir seçenek olarak gündeme getirilebilir (Adomian, 1988, s. 501-544).

1.2. Olasılıksal Evrim Yöntemleri Uzay Genişletimi

Olasılıksal evrim kuramı, denklem takımının çözümünü bir sonsuz toplamdizi (seri) açılımı olarak vermektedir. Dolayısıyla, olasılıksal evrim kuramı, biçimsel olarak kesin çözümü üretmektedir. Sonsuz toplamdiziden yapılan kesmeler (İng. truncation) ise, yaklaşık çözümü sayısal değerler olarak vermektedir. Elbette, yakınsama olgusu seriler için önemli bir kavramdır. Gözükırmızı ve Demiralp (2014a, s. 866-880; 2014b, s. 881-898) çalışmalarında yakınsama olgularına değinilmiştir, ama yakınsamayı, güncel gelişmeleri göz önüne alarak yeniden gündeme almak olasılıksal evrim kuramı ile ilgili yapılabilecek önemli bir iş olarak önümüzde durmaktadır.

Olasılıksal evrim kuramının oluşturduğu serinin terimleri çok küçük sayılarla çok büyük sayıların çarpımını gündeme getirdiği için kesin aritmetik (İng. exact arithmetic) kullanılmıştır. Böylelikle bilgisayar ortamındaki *double* yapısının kesin olmayan sayı gösteriliminden dolayı oluşan yanılgı birikiminden kaçınılmıştır.

Olasılıksal evrim kuramı ile ilgili en güncel olgular ve yöntemin koşutlaştırımı Gözükırmızı (2024) çalışmasında gündeme getirilmiştir. Demiralp (2018, s. 25-42) ise olasılıksal evrim kuramı ile ilgili oldukça önemli bir derleme makalesidir. Olasılıksal evrim kuramının nasıl sonuç ürettiği ve nerelerde kullanılabileceği sorusuna yanıt arayan okuyucu, bu iki çalışmayı inceleyerek sorusuna yanıt bulabilir. Burada yinelemeden kaçınma amacıyla, bu makalelerde bulunan olgular yeniden gündeme getirilmeyecektir.

Olasılıksal evrim kuramı tümcesindeki olasılıksal sözcüğü, bu kuramın nicem (İng. quantum) dizgelerin (İng. systems) beklenen değer devinimi (İng. expectation value dynamics) bağlamında çözümü için geliştirilmiş olmasından kaynaklıdır. Olasılık olgusu nicem dizgelerde kendiliğinden bulunmaktadır. Fakat daha sonra, bu

kuramın doğrudan klasik (nicem olmayan) dizgeler için de kullanılabileceği gözlemlendi. Burada odaklanılan durum da nicem olmayan durumdur. Dolayısıyla, olasılıksal evrim kuramının bu tez bağlamındaki uygulamasında, olasılıkla ilgili bir olgu yoktur. Olasılık kavramı, nicem dizgelerin çözümünde gündeme gelmektedir.

Evrim olgusu ise devingen dizgenin (İng. dynamic system) nasıl evrildiğini vurgular. Yani zaman geçtikçe sistemin hangi değerleri elde ettiği, sistemin evrimini betimler.

Bu yöntem geliştirildiğinde, "olasılıksal evrim yaklaşımı" tümcesi tercih edilmişti. Hunutlu ve diğerleri (2013) çalışması, olasılıksal evrimin zamansal olarak öncül yapısını gösteren çalışmalardan biridir. 2013 yılında yayımlanan bu çalışmada da van der Pol sistemi odağa alınmıştır, ama çözüm için bir özdeğer sorunu çözümü gereklidir. Özdeğer belirlenmesi, bilgisayar açısından ederi yüksek bir işlemdir. Olasılıksal evrim, özdeğer belirleme gerektirmeyecek biçimde yeniden bağıntılandırılınca, olasılıksal evrim yaklaşımı yerine olasılıksal evrim kuramı denmeye başlanmıştır. Bu biçimde, yapının tutarlı bir teoriye dönüştüğü vurgulanmıştır.

Bu tezde van der Pol denklemine odaklanılmasının nedeni van der Pol denkleminin sıklıkla kullanılan bir sınama denklem takımı olmasıdır. Bu tez, çok daha geniş bir denklem takımı ailesi için çözümleri sunabilecek yapıya kolaylıkla genelleştirilebilir. Genelleştirme eylemi, daha sonraya bırakılmıştır.

Yeniden vurgulamak gerekir ki, olasılıksal evrim kuramı ile doğrudan ilgili makale ve bildirilerin toplamı yüzlercedir. Bu bölümde bu bağlamda bir derleme (İng. survey) yapılmayacaktır. İlgili okuyucu Gözükırmızı (2024, s. 654-680) ve Demiralp (2018, s. 25-42) makalelerinden başlayarak ve bu makalelerde gönderimde bulunan yayınları inceleyerek konu hakkında derinlik elde edebilir.

Birçok türevli denklem takımı olasılıksal evrim kuramının işleyeceği hale getirilebilir. Olasılıksal evrim kuramı, ışın arama ile sadece ikinci derecelileştirim içerecek şekilde doğrudan işleyebilir. Bu yapı, Gözükırmızı (2024) makalesinin 1. denkleminde ayrıntılı olarak verilmiştir.

$$\dot{x_1}(t) = \alpha_{1,1} x_1(t)^{k_{1,1,1}} \cdots x_n(t)^{k_{1,1,n}} + \cdots + \alpha_{1,m_1} x_1(t)^{k_{1,m_{1,1}}} \cdots x_n(t)^{k_{1,m_1,n}}$$

$$\dot{x_2}(t) = \alpha_{2,1} x_1(t)^{k_{2,1,1}} \cdots x_n(t)^{k_{2,1,n}} + \cdots + \alpha_{2,m_2} x_1(t)^{k_{2,m_{2,1}}} \cdots x_n(t)^{k_{2,m_2,n}}$$

$$\dot{x_3}(t) = \alpha_{3,1} x_1(t)^{k_{3,1,1}} \cdots x_n(t)^{k_{3,1,n}} + \cdots + \alpha_{3,m_3} x_1(t)^{k_{3,m_{3,1}}} \cdots x_n(t)^{k_{3,m_3,n}}$$

$$\vdots\vdots$$

$$\dot{x_n}(t) = \alpha_{n,1} x_1(t)^{k_{n,1,1}} \cdots x_n(t)^{k_{n,1,n}} + \cdots + \alpha_{n,m_n} x_1(t)^{k_{n,m_{n,1}}} \cdots x_n(t)^{k_{n,m_n,n}}$$

(1)

Burada α ile gösterilen büyüklükler zaman bağımlılığı içermeyen katsayılardır. α'ların alt sırasayılandırımında iki öğe bulunmaktadır. Bunlardan birincisi hangi denkleme ait olduğunu, ikincisi ise o denklemin sağ yanındaki kaçıncı terim olduğunu göstermektedir. İşlevlerin üst sırasayılandırımı *k* gösterilmiştir. *k* ile gösterilen bu büyüklüklerin üç alt sırasayısı vardır. Bunlardan birincisi hangi denkleme ait olduğunu, ikincisi kaçıncı toplamsal terimde olduğunu, üçüncüsü ise hangi işlevin üssü olduğunu göstermektedir. α ve k değerleri elbette soruda verilmiştir.

Van der Pol dizgesi yukarıda belirtilen yapıda bir dizgedir.

Uzay genişletimi kavramı, denklem takımını doğrudan işlemek yerine, öncelikle istenilen yapıya getirip, sonra işlemekle ilgilidir. Uzay genişletimi bir ön işleme (İng. preprocessing) adımı olarak değerlendirilebileceği gibi, yöntemin bir parçası olarak da algılanabilir.

Denklem takımında bulunan işlevler, bir işlev uzayı tanımlar. Matematiksel anlatımla, bu uzay bir Hilbert uzayıdır (İng. Hilbert space). Bu uzayın taban takımı öğeleri işlevlerdir. Bu işlevlerin doğrusal birleştirimleri, uzayda bulunan bütün işlevleri üretir. Doğrusal birleştirim kavramı ise, yalın bir anlatımla, doğrusal birleştirime giren işlevlerin, istenilen sayılarla çarpılıp ve sonra toplanması anlamına gelmektedir.

Van der Pol denklemi bağlamında sıklıkla kullanılan Lienard dönüşümü de bir uzay genişletimidir. İkinci mertebe bir adet denklemle verilen van der Pol, birinci mertebe iki adet denkleme Lienard dönüşümü ile dönüştürülür. Yeni tanımlanan işlev, asıl (aranan) işleve işlevsel bağımlıdır (asıl işlevden üretilir).

Uzay genişletimi kavramı türevli (kısmi diferansiyel) denklemlerin seri çözümü için de kullanılmaktadır. Üsküplü ve Demiralp (2010, s. 266-286) bu yaklaşıma örnek olarak gösterilebilir.

Uzay genişletimi kavramının bir alt alanı da ikinci dereceleştirmedir (dördülleştirme, İng. quadratization). Sağ yanı ikinci derece çokçokterimli olan sıradan türevli denklemlerin çözümü için yöntemler bulunmaktadır. Olasılıksal evrim kuramı bu yöntem ailelerinden biridir. Bychkov ve Pogudin (2021, s. 122-136) ikinci dereceleştirme ile ilgili önemli bir çalışmadır. Bu çalışmayı temel alan Bychkov (2021) yazılımı ise, denklem takımlarını oldukça yüksek başarımla ikinci dereceleştirebilen bir yazılımdır. Buradaki temel yaklaşım önce çokçokterimleştirmek, sonra da ikinci dereceleştirmektir. Yani, başlangıçta sağ yandaki işlevler çokçokterimli olmasa bile, dönüşümler ile çokçokterimli hale getirilebilir ve daha sonra ikinci dereceleştirim gerçekleştirilebilir.

İkinci dereceleştirim üzerinde bu kadar çok durulmasının nedeni, doğrusallıktan (birinci derecelilik) sonra en iyi durumun ikinci derecelilik olmasıdır. Doğrusal denklemlerin sonucunun kolaylıkla elde edilebildiği bilinmektedir, ama çok az sayıda denklem doğrusal denkleme dönüştürülebilir. Dolayısıyla, doğrusallık söz konusu değilse, deyim yerindeyse, bir sonraki en iyi şeyi kullanmaya çalışmak önem kazanmaktadır.

İkinci dereceleştirim dışında yaklaşımlar da bulunmaktadır. Denklem takımının sağ yan işlev derecelerini oldukça yüksek hale getirerek de daha kolay işlenebilen yapılar elde edilebilir (Kalay ve Demiralp, 2018, s. 2024-2043).

Bychkov (2021) yazılımında ve Gözükırmızı ve Demiralp (2019, s. 367) makalesinde izlenen yaklaşım, ikinci dereceleştirimi bir labirentin çıkışını bulma sorunu olarak düşünmektir. Sağ yanı çokçokterimli olan denklem takımının ikinci dereceleştirimi bu türde, tam sayılar üzerinden tanımlanmış bir eniyileme sorunudur. Bir labirentte çıkışı ararken dikkat edilmesi gereken birçok olgu vardır. Bunlardan en önemlisi aynı çıkmaz olduğunu bildiğiniz yolu işaretleyip, yeniden oraya gitmeyi engellemektir. Ayrıca, çıkışı bulduğunuzda, izlediğiniz yolun en kısa yol olup olmadığı bilmek de isteyebilirsiniz. Yani, ikinci dereceleştirimde, elde ettiğiniz yapının en az yeni işlev tanımı olan yapı olması da önem kazanabilmektedir.

Bir önceki paragrafta anlatılan olgular yapay zekâ alanının arama algoritmaları alt alanının konularıdır. Gözükırmızı ve Demiralp (2019), Gözükırmızı (2023), ve Gözükırmızı (2022) olasılıksal evrim kuramı bağlamındaki ikinci dereceleştirimi ayrıntılandırmaktadır. Bu çalışmada kullanılan algoritmada, arama algoritmalarından ışın arama (beam search) kullanılmıştır. En iyi (optimal) sonucu üretme garantisi vermemesine karşın, algoritma eğer iyi bir buluşsal (İng. heuristic) kullanılmışsa oldukça kısa sürede sonuç vermektedir. Daha ayrıntılandırmak gerekirse, bu çalışmada ışın aramanın özel bir durumu olan, her adımda yalnızca bir seçeneğin değerlendirildiği tepe tırmanışı (İng. hill climbing) arama algoritması kullanılmıştır. Ama daha genel bir terim olan ışın arama tümcesi, bu anlatımda kullanılmaktadır. Olasılıksal evrim kuramı bağlamındaki ışın arama Gözükırmızı (2022, s. 100-114) çalışmasında ayrıntılandırılmıştır.

Uzayın "genişlemesi" olgusu ise denklem takımına yeni bir denklem eklemek ile ilgilidir. Kullanılan yeni işlev önceki durumdaki bilinmeyen (aranan, İng: unknown, sought) işlevlere işlevsel olarak bağımlıdır ama onlardan doğrusal bağımsızdır. Bu işlevler bir Hilbert uzayı tanımladığına göre, yeni bir taban işlevi eklemek uzayı genişletmeye karşılık gelmektedir. Dolayısıyla yeni bir işlev için denklem oluşturmak, Hilbert uzayına yeni bir taban işlevi ekleyerek uzayı genişletmektir.

Bu aşamada, vurgulamak gerekir ki, uzay genişletimi ve olasılıksal evrim kuramının matematiği ile ilgili bu anlatımlar, tezin omurgasını oluşturmamaktadır. Tez, uzay genişletimi ve olasılıksal evrim kuramına kuramsal bir katkı sunmamaktadır. Tez, daha önce yapılan kuramsal gelişmelerin ve daha önce geliştirilen algoritmanın çok daha etkin biçimde kullanımına yöneliktir.

1.3. Veri Tabanı Sistemleri ve Önemi

Verilerin birbirleriyle ilişkili olsun veya olmasın belli bir düzen altında tutuldukları yerlere veri tabanı denir (Gültekin vd., 2015, s. 198-209). Veri tabanı, yapılandırılmış verilerin saklandığı, yönetildiği ve erişildiği bir elektronik depodur. Genellikle bilgisayar sistemlerinde kullanılan veri tabanları, verilerin etkin bir şekilde depolanması, düzenlenmesi, güncellenmesi ve geri alınması için tasarlanmıştır. Veri tabanları, bilgilerin işlenmesini ve veriye dayalı kararların alınmasını kolaylaştırır.

Veri tabanı türleri çeşitli özelliklere ve kullanım senaryolarına göre farklılık gösterebilir. En yaygın veri tabanı türleri aşağıda belirtilmiştir.

İlişkisel Veri Tabanları, verilerin tablolar şeklinde düzenlendiği ve ilişkisel veri modeli üzerine kurulan veri tabanlarıdır. Örneğin, MySQL, PostgreSQL, Oracle, SQL Server gibi ilişkisel veri tabanı yönetim sistemleri (RDBMS) bu kategoriye girer.

NoSQL Veri Tabanları, ilişkisel modelin dışında, özellikle büyük veri hacimlerini işleme, dağıtık sistemlerde çalışma ve esnek veri yapılarını destekleme ihtiyacı olan uygulamalar için tasarlanmıştır. Örnek olarak MongoDB, Cassandra, Redis, Neo4j gibi NoSQL veri tabanları verilebilir.

Doküman Tabanlı Veri Tabanları, JSON veya XML gibi belge tabanlı veri yapılarını kullanan ve verileri belgeler şeklinde depolayan veri tabanlarıdır. MongoDB gibi NoSQL veri tabanları bu kategoriye örnektir.

Açkı-Değer Dağarcıkları, basit açkı-değer çiftleriyle verileri saklayan ve hızlı erişim sağlayan veri tabanlarıdır. Redis ve Memcached bu kategoriye örnektir.

Graf (Çizge) Tabanlı Veri Tabanları, ağ veya graf yapısını kullanan ilişkisel veri tabanı türüdür. Bağlantıların karmaşık ilişkilerini modellemek için kullanılır. Örnek olarak Neo4j verilebilir.

Kolon (düşeysıra) Ailesi (Column-family) Veri Tabanları, genellikle dağıtık sistemlerde kullanılan, kolon tabanlı veri depolama yöntemini kullanan veri tabanlarıdır. Apache Cassandra bu tür bir veri tabanıdır.

Veri tabanı sistemleri, geniş kapsamlı araştırmalarda elde edilen verilerin düzenlenmesi, saklanması ve çözümlenmesi için temel bir altyapı sağlar. Veri tabanları farklı gereksinimlere ve kullanım senaryolarına uygun olarak seçilir ve kullanılır. Bu tez kapsamında, Van der Pol denkleminin sayısal çözümlemelerinden elde edilen veriler MySQL veri tabanı kullanılarak yönetilmektedir. MySQL, açık kaynaklı ve yaygın kullanılan bir ilişkisel veri tabanı yönetim sistemidir, bu özellikleri ile büyük veri setleri üzerinde yüksek performans ve esneklik sunar.

Bu projede, veriler SQL sorgularıyla işlenen bir veri tabanında saklanmaktadır. SQL, verilerin nasıl düzenleneceğini ve bu veriler üzerinde hangi işlemlerin yapılabileceğini belirten bir dildir. Tezde, denklemin farklı parametrelerle çözümüne

ilişkin veriler, veri tabanındaki belirli tablolar ve sütunlar içinde depolanmıştır. Bu yapı, sonuçların hızlıca sorgulanmasına ve analiz edilmesine olanak tanır.

Örnek uygulamalar şu şekildedir:

Veri Saklama ve Erişim, Van der Pol denkleminin çözümlemeleri sırasında elde edilen veriler, MySQL veri tabanında çeşitli tablolarda saklanır. Bu tablolar, denklemin parametreleri, çözümlemelerden elde edilen değerler ve simülasyon metadatasını içerir.

Sorgulama ve Analiz, araştırmacılar, MySQL üzerindeki SQL sorgulama kabiliyetini kullanarak, çeşitli senaryolar altında elde edilen veriler arasında karşılaştırmalar yapabilir ve çeşitli analizler gerçekleştirebilirler. Bu, modelin davranışını daha iyi anlamak ve teorik sonuçları doğrulamak ve karşılaştırmak için kullanılır.

Raporlama ve Görselleştirme, veri tabanından elde edilen veriler, raporlama ve görselleştirme araçları ile entegre edilebilir. Bu sayede, verilerin grafiksel sunumları aracılığıyla daha anlaşılır bilgiler elde edilir, bu da bilimsel sunumlar ve akademik yayınlar için değerli bir kaynak oluşturur.

2. SİSTEM TASARIMI VE GELİŞTİRME

Bu bölümde, Van der Pol denkleminin sayısal çözümlerini elde etmek ve bu çözümleri yönetmek için geliştirilen sistemin genel yapısı özetlenmektedir. Sistem, veri toplama, işleme ve saklama süreçlerini bütünleştiren yazılım ve donanım bileşenlerinden oluşmaktadır. Yazılım altyapısı olarak Node.js ve Express.js kullanılarak dinamik bir sunucu ortamı sağlanmış, C++ dilinde yazılmış sayısal çözüm algoritmalarıyla Van der Pol denkleminin çeşitli parametreler altında çözümleri elde edilmiştir. Donanım altyapısı ise yüksek işlem kapasitesine sahip bir sunucu üzerinde yapılandırılmıştır. Elde edilen veriler, MySQL veri tabanına kaydedilerek düzenli bir şekilde saklanmakta ve analiz edilmektedir. Bu sistem tasarımı, Van der Pol denkleminin çözümlerini hesaplamak, işlemek ve analiz etmek için optimize edilmiştir.

2.1. Yazılım ve Donanım Gereksinimleri

Bu tez çalışmasında kullanılan yazılım ve donanım altyapısı, Van der Pol denkleminin çözümlerinin elde edilmesi, işlenmesi ve sunulması için gerekli bileşenleri içermektedir. Sistem, 6 çekirdekli Xenon işlemcili, 16 GB RAM'e ve 1 TB SAS disk kapasitesine sahip bir sanal sunucu üzerinde çalışmakta olup, bu donanım konfigürasyonu hesaplama yoğunluğu düşük işlemler için yeterli performansı sağlamaktadır. Ayrıca, kullanılan bütün yazılımlar ücretsiz olarak indirilebilen özgür yazılımlar olup, Node.js ve Express.js gibi modern teknolojiler ile C++ kodlarının entegrasyonundan yararlanılmaktadır. Bu altyapı, sistem performansını ve kullanılabilirliğini artırırken, bilimsel bulguların verimli bir şekilde işlenmesini ve sunulmasını sağlamaktadır.

2.2. Van der Pol Denkleminin Modellenmesi

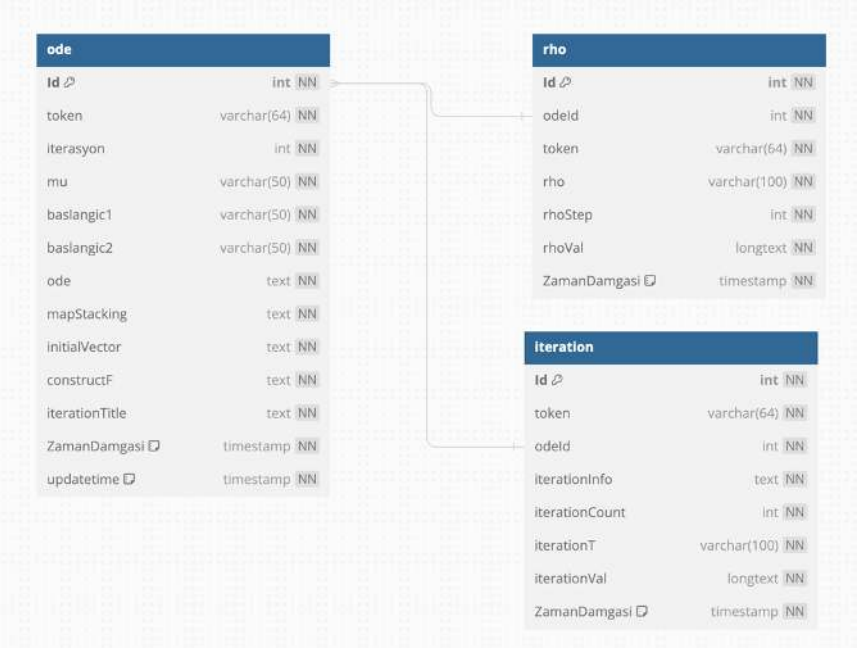
Van der Pol denklemi, nonlineer dinamikler içeren fiziksel ve mühendislik sistemlerinin analizinde kullanılan ikinci mertebeden bir diferansiyel denklemdir. Bu tez çalışması kapsamında, Van der Pol osilatörünün sayısal çözümleri için bir modelleme yaklaşımı kullanılmıştır.

2.3. Veri Tabanı Tasarımı

Bu tezde kullanılan veri tabanı tasarımı, Van der Pol denkleminin çözümlerini düzenli, güvenli ve etkin bir şekilde saklamayı ve sorgulamayı amaçlamaktadır. MySQL veri tabanı sistemine dayanan bu tasarım, ilişkisel veri tabanı modeli kullanarak verileri yönetir. Veri tabanı, çözümleme sonuçları ve çeşitli simülasyon parametrelerini içeren tablolar ile yapılandırılmıştır. Tablolar arasındaki ilişkiler, veri bütünlüğünü korurken, karmaşık sorgulamaların da kolaylıkla yapılmasına olanak tanır. İndeksleme, sık kullanılan sorgu alanlarında performansı artırmak için kullanılır.

Veri tabanının güvenliği ve erişim kontrolü de büyük önem taşır. Kullanıcı yetkilendirmesi ile veri tabanına erişim sınırlandırılır, bu sayede yetkisiz erişimlerin önüne geçilir. Ayrıca, düzenli yedekleme ve veri kurtarma stratejileri ile veri güvenliği sağlanır.

Bu veri tabanı tasarımı, Van der Pol denkleminin çözümlerine ait verilerin düzenli bir şekilde saklanmasını, sınıflandırılmasını ve etkin bir şekilde sorgulanmasını sağlar. Veri tabanı temelleri hakkında daha ayrıntılı bilgi edinmek isteyenler, Elmasri ve Navathe (2017) 'nin "Veri Tabanı Sistemleri" kitabına başvurabilirler.



Şekil 2. Veri Tabanı Yapısı

Kaynak: Yazar tarafından oluşturulmuştur.

2.4. Sistem Tasarımı ve Geliştirme Süreci

Projenin genel mimarisi şu adımları içerir:

Kullanıcı etkileşimi, kullanıcı, web arayüzü üzerinden parametreleri girer.

Sunucu işlemleri, Node.js sunucusu, kullanıcının girdiği parametreleri işler ve ilgili C++ kodunu çalıştırır

C++ hesaplamaları, C++ kodu, verilen parametrelerle Van der Pol denklemini çözer.

Veri tabanı İşlemleri, çıktılar JSON formatında işlenir ve MySQL veri tabanına kaydedilir.

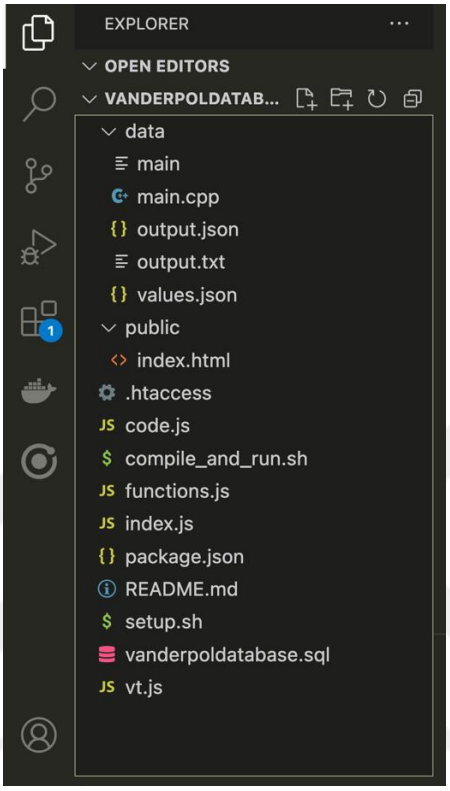
Sonuçların sunumu, veri tabanından alınan veriler, web arayüzü üzerinden kullanıcıya sunulur.

2.4.1. Geliştirme Ortamı Visual Studio Code (VS Code)

Visual Studio Code (VS Code), Microsoft tarafından geliştirilen, ücretsiz ve açık kaynaklı bir kod düzenleyicisidir. Windows, macOS ve Linux platformlarında çalışabilen bu çapraz platform aracı, geniş eklenti desteği, IntelliSense (kod tamamlama), hata ayıklama ve Git (sürüm/versiyon kontrol sistemi) entegrasyonu gibi özellikleriyle popüler bir tercihtir. VS Code, performans ve hafiflik açısından yüksek verimlilik sunarken, aynı zamanda ücretsiz olması ve geniş bir kullanıcı kitlesine sahip olması nedeniyle tercih edilmektedir. JavaScript, Node.js, C++, MySQL ve birçok dili destekleyen zengin eklenti yelpazesi, VS Code'u esnek bir geliştirme ortamı haline getirmektedir. Ayrıca, Git ile doğrudan entegrasyon sağlayarak kod yönetimini kolaylaştırmakta, IntelliSense, hata ayıklama ve kod parçacığı desteği gibi gelişmiş özellikler sunarak geliştirme süreçlerini daha verimli hale getirmektedir. VS Code projeyi geliştirme sürecinde kullanılmış bir kod yazma editörüdür. Sistemi kullanan kullanıcıların, denklem çözümlerini analiz etmede ve sonuçları değerlendirmede VS Code bilmeleri veya kullanmaları gerekmemektedir.

2.4.2. Proje Dosyaları ve Fonksiyonlar

Bu bölümde, projede kullanılan başlıca dosyalar ve fonksiyonların ne yaptığını detaylandırılmaktadır.



Şekil 3. Projenin Dosya Yapısı

Kaynak: Yazar tarafından oluşturulmuştur.

Tablo 1. Proje Dosyaları ve Açıklamaları

Dosya Dizin	**Açıklama**
data/	Proje verilerini ve C++ kodlarını içeren klasördür.
main.cpp	Van der Pol denkleminin sayısal çözümlerini gerçekleştiren C++ kodunu içerir.
output.json	Çıktı verilerini JSON formatında saklar.
output.txt	Çıktı verilerini metin formatında saklar
values.json	Parametre değerlerini saklar.
public/	Web arayüzü dosyalarını içerir
index.html	Ana HTML dosyasıdır, kullanıcı arayüzünü tanımlar.
code.js	Uygulamanın işlevselliğini sağlayan JavaScript kodlarını içerir.
functions.js	Sunucu tarafı işlevselliği yönetir. Bazı fonksiyonların çalışma şekli şöyledir. getCode: C++ kodunu okur ve kullanıcıya sunar. getValues ve setValues: Parametre değerlerini okur ve ayarlar.
index.js	Web uygulamasının ana giriş noktasıdır. Sunucuyu başlatır ve yönlendirmeleri tanımlar.
vt.js	MySQL veritabanı operasyonları için yazılmış fonksiyonları içerir.
compile_and_run.sh	C++ kodunu derleyip çalıştırmak için kullanılan Shell script dosyasıdır.
package.json	Node.js projesi için bağımlılıkları ve diğer proje bilgilerini tanımlar.
README.md	Proje hakkında bilgi veren döküman dosyasıdır.
setup.sh	Proje kurulumunu gerçekleştiren kabuk betiği dosyasıdır. Linux sistemler için projeyi ve bağımlıklarını kurmayı sağlar.
.htaccess	Apache web sunucusu için yapılandırma dosyasıdır. Sadece Apache ile kullanılır. Nginx ve IIS gibi farklı web sunucular da var ve bu sunucular için farklı yapılandırma dosyaları vardır.

Kaynak: Yazar tarafından oluşturulmuştur.

Örnek Fonksiyon olarak aşağıda getCppOutput verilmiş ve çalışma şekli açıklanmıştır.

```
119     getCppOutput: async function(req, res) {
120         const compileAndRunCommand = path.join(__dirname, 'compile_and_run.sh');
121
122         exec(compileAndRunCommand, { timeout: 5000 }, async (error, stdout, stderr) =>
            {
123             if (error) {
124                 console.error(`Hata oluştu: ${error}`);
125                 res.status(500).json({ error: 'Dosya çalıştırma hatası', message:
                    stderr });
126                 return;
127             }
128             try {
129                 await this.outputJson(); //output.json dosyasını oluştur
130                 res.send({ output: "success" });
131             } catch (err) {
132                 res.status(500).send('Çıktı dosyası okunamadı.');
133             }
134         });
135     },
```

Şekil 4. Örnek getCppOutput Fonksiyonu

Kaynak: Yazar tarafından oluşturulmuştur.

Fonksiyonun kullanıcının C++ kodunu derlemesini ve çalıştırmasını sağlar. compile_and_run.sh script dosyasını çağırarak C++ kodunu derler ve çalıştırır. Elde edilen çıktıyı kullanıcıya sunar. İsteğe bağlı olarak kodun çalıştırılması için gerekli parametreler alınır ve işlenir.

2.4.3. Projenin GitHub Üzerinde Paylaşımı

Bu proje, GitHub üzerinde paylaşılmıştır ve https://github.com/mustafaozturkdev/vanderpoldatabase bağlantısından erişilebilir: GitHub, projelerin versiyon kontrolünü sağlamak ve kodları paylaşmak için kullanılan bir platformdur. Bu bağlantı aracılığıyla projeye erişebilir, kaynak kodlarını inceleyebilir ve projeyi kendi bilgisayarınıza klonlayabilir, açık kaynak olarak kullanabilirsiniz. Proje hakkında daha fazla bilgi ve kullanım talimatları GitHub deposunda mevcuttur.

3. UYGULAMA VE VERİ TOPLAMA

Bu bölümde, Van der Pol denkleminin çözüm süreçlerinin uygulanması ve bu süreçler sırasında verilerin nasıl toplandığı ayrıntılı bir şekilde incelenmektedir. Projenin temel amacı olan Van der Pol Osilatörünün çözümlerini hızlı ve etkin bir şekilde üretmektir. Bu verileri organize bir şekilde saklamak için geliştirilen uygulamalar ve veri toplama yöntemleri, bu bölümün odak noktasını oluşturmaktadır.

Projede kullanılan uygulama, Node.js ve Express.js framework'leri kullanılarak geliştirilmiş bir web tabanlı arayüzdür. Bu arayüz, kullanıcıların Van der Pol denkleminin parametrelerini kolaylıkla değiştirmesine ve denklemin çözümlerini gerçek zamanlı olarak gözlemlemesine olanak tanır. Uygulama, aynı zamanda kullanıcı girişlerini ve sistem etkileşimlerini kaydetme yeteneğine sahiptir, böylece kullanıcı davranışları üzerine veri analizleri yapılabilir.

Veri toplama süreci, denklemin çözümü için gerekli olan başlangıç değerleri, parametre ayarları ve çözüm sonuçları gibi bilgileri içerir. Bu veriler, kullanıcılar tarafından web arayüzü üzerinden girilir ve ardından C++ ile yazılmış olan ve sunucuda çalışan hesaplama motoru tarafından işlenir. İşlenen veriler, çeşitli sayısal ve istatistiksel analizler için kullanabilmek üzere veri tabanına kaydedilir.

Veri toplama ve işleme süreçleri hem kullanıcı girdileri hem de otomatik olarak üretilen verileri kapsar. Bu, sistem üzerinde yapılan bütün işlemlerin kaydedilmesini ve gelecekteki analizler için kullanılmasını sağlar. Ayrıca, bu süreç sistem performansını izlemek ve potansiyel iyileştirmeleri belirlemek için de önemlidir.

Uygulamanın tasarımı ve işlevselliği, kullanıcıların ihtiyaçlarına yanıt vermek üzere düzenlenmiştir. Bu sayede, kullanıcılar denklemin çözümünü etkileyebilecek her türlü değişikliği kolayca yapabilir ve sonuçları anında görebilirler. Bu interaktif yaklaşım, kullanıcı deneyimini zenginleştirirken, eğitim ve araştırma amaçlı kullanımlar için de örnek bir platform sunmaktadır.

Bu bölümde tanıtılan uygulama ve veri toplama yöntemleri, Van der Pol denkleminin daha iyi anlaşılmasına ve bu bilgilerin geniş bir kullanıcı kitlesi ile paylaşılmasına olanak tanımaktadır. Ayrıca, toplanan verilerin analizi, denklemin çeşitli parametreler altında nasıl davrandığını daha iyi anlamamızı sağlayacak bilgiler sunmaktadır.

3.1. Node.js Kullanımı

Bu projede, Van der Pol denkleminin çözümlerini web tabanlı bir platform üzerinden kullanıcılarla etkileşimli bir şekilde işlemek ve sunmak için Node.js teknolojisi kullanılmıştır. Node.js, açık kaynaklı ve platformlar arası bir JavaScript çalışma zamanı ortamıdır. Neredeyse her türlü proje için popüler bir araçtır. Node.js, Google Chrome'un çekirdeği olan V8 JavaScript motorunu tarayıcının dışında çalıştırır. Bu, Node.js'nin oldukça performanslı olmasını sağlamaktadır (Nodejs.com, 2024).

Projede Node.js'in kullanılmasının temel nedeni, bu platformun tek thread üzerinde non-blocking I/O işlemleri gerçekleştirebilmesidir. Bu özellik, sistemin büyük miktarda veri işlemesini ve birden fazla kullanıcıya aynı anda hizmet vermesini sağlar. Ayrıca, JavaScript'in hem sunucu hem de istemci tarafında kullanılabilmesi, uygulamanın geliştirilmesi sürecinde tutarlılık ve verimlilik sağlar.

Node.js, aşağıdaki özellikleri dolayısıyla projemiz için tercih edilmiştir:

Asenkron ve Olay Tabanlı:

Node.js, asenkron, olaya dayalı bir mimari kullanır. Bu, I/O işlemlerinin bloklanmadan, yani sunucu kaynaklarını boşa harcamadan gerçekleştirilmesine olanak tanır. Bu özellik, veri yoğun ve gerçek zamanlı uygulamalar için idealdir, çünkü birçok isteği aynı anda ve verimli bir şekilde işleyebilir. Bu özellik, Van der Pol denkleminin çözüm süreçlerini hızlı ve etkin bir şekilde işlememize olanak tanır.

Tek Dil Üzerinden Geliştirme:

JavaScript kullanarak hem sunucu tarafı hem de istemci tarafı kodlarını yazma imkânı, geliştirme sürecini basitleştirir ve hızlandırır. Bu, proje içerisinde teknoloji tutarlılığını ve kod bakımını kolaylaştırır.

Modüler Yapı:

Node.js, modüler bir yapıya sahiptir ve NPM (Node Package Manager) aracılığıyla erişilebilen binlerce kütüphane ile desteklenir. Bu proje kapsamında, Express.js gibi framework'ler ve diğer yardımcı kütüphaneler kullanılmıştır. Express.js, REST API'lerin kolayca oluşturulması için yalın ve esnek bir yöntem

sunar, bu da veri yönetimi ve sunumu için gerekli endpointlerin hızla geliştirilmesine yardımcı olur.

Ölçeklenebilirlik:

Node.js uygulamaları, yatay ve dikey olarak kolayca ölçeklenebilir. Bu, projenin gelişen gereksinimlere uyum sağlamasına ve artan kullanıcı sayısı veya veri hacmi ile başa çıkabilmesine olanak tanır.

Node.js kullanımının projemize getirdiği bir diğer avantaj da geliştirme sürecinde JavaScript kullanarak hem sunucu tarafında hem de istemci tarafında çalışabilmesidir. Bu da geliştirme sürecini daha hızlı ve etkili hale getirir. Özellikle web tabanlı uygulamalar ve gerçek zamanlı veri işleme gerektiren sistemlerde Node.js, bu yönleriyle ön plana çıkmaktadır.

Projede Node.js kullanımı şe şekilde gerçekleşmektedir:

API Geliştirme:

Express.js framework'ü kullanılarak RESTful API'lar geliştirilmiştir. API (Application Programming Interface), farklı yazılım bileşenlerinin birbirleriyle iletişim kurmasına olanak tanıyan bir arabirimdir. Bu API'lar, uygulama içerisindeki veri akışını yönetir ve kullanıcı isteklerine yanıt verir. Kullanıcılar, API'lar aracılığıyla sistemdeki veri kümeleri üzerinde sorgulama yapabilir, veri ekleyebilir, güncelleyebilir veya silebilirler. Ayrıca, API'lar, kullanıcıların sistemde çeşitli işlemleri tetiklemelerine ve verileri işlemelerine olanak tanır. API'lar, uygulamanın diğer yazılımlar ve hizmetlerle entegrasyonunu da sağlar, bu sayede farklı platformlar arasında veri paylaşımı ve etkileşim mümkün hale gelir. API kullanımı ileriki bölümde ayrıntılandırılacaktır.

C++ Entegrasyonu:

Node.js, child processes (çocuk süreçler) oluşturarak, C++ yazılımının derlenmesi ve çalıştırılması işlemlerini yönetir. Bu sayede, sayısal çözümler üretilir ve sonuçlar sistemde işlenir.

Veri İşleme ve Yönetimi:

Elde edilen sayısal çözüm verileri, Node.js tarafından işlenir, analiz edilir ve MySQL veri tabanına kaydedilir. Ayrıca, veri güvenliği ve erişim kontrolü gibi önemli görevler de Node.js tarafından yönetilir.

Sonuç olarak, Node.js teknolojisi, bu projede Van der Pol denkleminin çözümlerinin etkin bir şekilde işlenip kullanıcılara sunulması için kritik bir rol oynamaktadır. Bu teknoloji sayesinde, sistemimiz yüksek performanslı, ölçeklenebilir ve kullanıcı dostu bir platform olarak işlev görebilmektedir.

3.2. Express.js ile API Tasarımı

Projemizde, Van der Pol denkleminin çözümlerini işlemek ve bu çözümleri kullanıcıya sunmak için Express.js framework'ü kullanılarak bir RESTful API tasarlanmıştır. Express.js, Node.js uygulamaları için hızlı, esnek ve yalın bir web frameworküdür. Bu teknoloji, API'nin geliştirilmesini basitleştirerek, uygulama üzerinde veri alışverişini kolaylaştırır ve geliştirme sürecini hızlandırır (Expressjs.com, 2024).

API (Application Programming Interface), iki uygulama arasında belirli tanımlar aracılığıyla iletişim kurmayı sağlayan bir yazılım aracıdır. Bir uygulamanın verilere, sunucu yazılımına veya diğer programlara erişebilmesini sağlayan bir bağlantı arayüzüdür. Web tabanlı veya mobil uygulamalar, API'ler sayesinde verilere kolayca erişebilir ve iletişim kurabilirler. API terimi, 1974 yılında Christopher J. Date tarafından yayımlanan bir makalede ortaya çıkmıştır. API'lerin kullanımı giderek yaygınlaşmakta olup, sosyal medya, e-ticaret, hava durumu, finansal veriler gibi birçok alanda sıklıkla kullanılmaktadır. API'ler HTML, XML ve JSON gibi farklı protokollerle kullanılır ve web geliştiricileri ile program geliştiricileri tarafından sıklıkla kullanılır.

API'lerin avantajları arasında şunlar bulunmaktadır: güvenlik sağlaması, işlemlerin hızlı ve kolay bir şekilde gerçekleştirilmesi, program geliştirmeyi kolaylaştırması, zamandan ve paradan tasarruf sağlaması, trafiği ve görünürlüğü artırması (Coderspace.io, 2024). Express.js, bu süreçler için gerekli olan route'ları (yönlendirmeleri) tanımlamada büyük kolaylık sağlar. Her bir route, bir HTTP

metoduna (GET, POST, PUT, DELETE gibi) bağlı olarak belirli bir işlevi yerine getirir ve böylece uygulamanın modüler ve bakımı kolay bir yapıda olmasını sağlar.

Projemizde kullanılan API yapısında POST metodu kullanılmıştır, "/api" ana endpoint ve gönderilen "action" parametresi üzerinden diğer istek parametrelerini alır ve kullanıcıların sistemle etkileşimini kolaylaştırır ve uygulamanın farklı bölümleri arasında veri akışını yönetir.

API güvenliği ve performansı, bir uygulamanın sağlıklı ve güvenli bir şekilde çalışabilmesi için kritik öneme sahiptir. Güvenlik, API'ler aracılığıyla iletilen verilerin gizliliğini, bütünlüğünü ve erişilebilirliğini sağlamak için gereklidir. Güvenlik zayıflıkları, hassas bilgilerin sızdırılmasına, yetkisiz erişime veya kötü amaçlı saldırılara yol açabilir. Performans ise API'lerin hızlı ve etkin bir şekilde çalışmasını sağlayarak kullanıcı deneyimini iyileştirir. Yavaş veya aşırı yüklenmiş API'ler, kullanıcıların beklemesine veya hatalara neden olabilir, bu da kullanıcı memnuniyetini ve uygulama kullanımını olumsuz etkileyebilir. Bu nedenle, API güvenliğinin sağlanması ve performansının optimize edilmesi, güvenilir ve kullanıcı dostu bir uygulama geliştirmenin önemli bir parçasıdır. Express.js, middleware katmanlarını kullanarak API güvenliğini artırmak için çeşitli araçlar sunar. CORS (Cross-Origin Resource Sharing) ve güvenlik duvarı ayarları, API'ye yapılan isteklerin güvenliğini sağlamak için konfigüre edilmiştir. Ayrıca, yüksek trafik durumlarında API performansını korumak için rate limiting ve caching mekanizmaları uygulanmıştır.

API'nin doğru çalıştığından emin olmak için ise, kapsamlı test süreçleri uygulanmıştır. Bu testler, birim testleri, entegrasyon testleri ve yük testleri olmak üzere üç ana kategori altında toplanmıştır.

Express.js ile tasarlanan API, projemizin temel taşıdır ve Van der Pol denkleminin çözümlerinin etkin bir şekilde işlenip kullanıcılara sunulmasında kritik bir role sahiptir. API'nin modüler yapısı, geliştirme sürecinde esneklik sağlar ve gelecekteki gereksinimlere göre sistemde yapılacak değişikliklere veya genişlemelere olanak tanır. Bu yaklaşım, uygulamanın hem teknik hem de kullanıcı deneyimi açısından başarısını desteklemektedir.

3.3. C++ ile Sayısal Çözüm Kodunun Shell Script (Kabuk Betiği) ile Çalıştırılması

Bu bölüm, Van der Pol denkleminin sayısal çözümleri için kullanılan C++ kodunun, Shell Script aracılığıyla nasıl derlenip çalıştırıldığını açıklamaktadır. Prof. Dr. Metin Demiralp ve ekibinin geliştirdiği metodolojiler, özellikle açık (belirtik, İng: explicit) diferansiyel denklemlerin uzay genişletme teknikleri kullanılarak çözülmesini amaçlamaktadır. Bu metodolojilerle Dr. Coşar Gözükırmızı tarafından oluşturulan C++ kodu, uzay genişletme ve olasılıksal evrim yöntemi ile bu tezde ele alınan sayısal çözümün temelini oluşturur.

Projede Van der Pol denkleminin sayısal çözümü için C++ programlama dili kullanılmıştır. C++, yüksek performanslı matematiksel hesaplamalar için geniş kabul görmüş bir dil olup, bu projede de denklemi başarımlı bir şekilde çözebilmek için tercih edilmiştir. C++ ile geliştirilen sayısal çözüm kodunun, sistemde otomatik ve düzenli olarak çalıştırılması için bir Shell script kullanılmaktadır. Bu yaklaşım, hesaplama süreçlerinin yönetimini otomatikleştirir ve kullanıcı etkileşimini gerektirmeden veri işleme ve analiz işlemlerini kolaylaştırır. C++ kodu, Van der Pol denklemini sayısal yöntem kullanarak çözer. Kod, başlangıç koşulları, zaman aralığı ve diğer parametreler gibi girdileri alır ve denklemin çözümünü adım adım hesaplar. Bu hesaplama sürecinde, denklem yüksek doğrulukta sonuçlar üretir.

C++ kodunun çalıştırılması ve derlenmesi için kullanılan Shell script, Linux ve Unix tabanlı sistemlerde komut dosyası olarak işlev görür. Bu script, belirli zaman aralıklarıyla veya tetikleyici bir olay sonucunda C++ kodunu otomatik olarak çalıştırır, böylece kullanıcıların manuel müdahalesine gerek kalmadan sürekli veri akışı sağlar. Shell script ayrıca, hesaplama sonuçlarını düzenli bir biçimde kaydetme, hata yönetimi ve sistem kaynaklarını izleme gibi görevleri de üstlenir. Bu, sistemin kararlılığını ve güvenilirliğini artırır, aynı zamanda olası problemlerin hızla tespit edilip çözülmesine olanak tanır.

Shell script kullanımı, projenin otomasyon ve verimlilik yönlerini önemli ölçüde güçlendirir. Script, belirlenen zamanlama veya olaya bağlı olarak C++ kodunu çalıştırarak, süreçleri otomatize eder ve böylece sürekli ve düzenli veri üretimi sağlar. Bu otomasyon sayesinde, projede iş yükü azalır ve veri yönetimi süreçleri daha az hata

ile daha hızlı yürütülür. Bu yaklaşım, projenin genel performansını ve kullanıcı memnuniyetini artırarak, bilimsel araştırmalar ve eğitim uygulamaları için değerli bir kaynak sunar.

3.4. Verilerin Saklanması ve Yönetimi

MySQL veri tabanı, Van der Pol denkleminin çeşitli parametrelerle elde edilen sayısal çözümlerini saklamak için kullanılmaktadır. Veri tabanı, çözüm sonuçlarını, kullanıcı girdilerini ve hesaplama parametrelerini içeren çeşitli tablolara sahiptir. Her bir tablo, verilerin hızlı sorgulanabilmesi ve kolay erişilebilir olması için uygun şekilde indekslenmiştir. Veri tabanı tasarımında, veri bütünlüğünü sağlamak için foreign key bağlantıları kullanılmış ve gerektiğinde veri tipi kısıtlamaları getirilmiştir.

Verilerin saklanması, çözümleme sürecinden elde edilen çıktıları sistematik bir şekilde düzenlemek amacıyla tasarlanmıştır. Veri modeli, çözümleme sonuçları, başlangıç değerleri, kullanılan iterasyon sayısı ve parametreler gibi bilgileri içerir. Ayrıca, her bir hesaplama işlemi için benzersiz bir tanımlayıcı (ID) saklanarak, veriler arasındaki ilişkilerin net bir şekilde tanımlanması sağlanmıştır.

Veri güvenliği ise projenin en önemli unsurlarından biridir. Veri tabanı, yetkisiz erişime karşı korunmakta ve veri erişim kontrolü için kullanıcı bazlı yetkilendirme mekanizmaları uygulanmaktadır. Ayrıca, veri tabanı düzenli olarak yedeklenmekte ve potansiyel veri kaybına karşı önlemler alınmaktadır.

Veri yönetimi, veri tabanının düzenli olarak bakımının yapılmasını ve verilerin güncellenmesini içerir. Veri tabanı yönetimi, performansı optimize etmek ve veri bütünlüğünü korumak için gereken indeksleme ve veri tabanı normalizasyon işlemlerini de kapsar. Sistemde meydana gelebilecek herhangi bir arıza durumunda hızlı bir şekilde müdahale edilebilmesi için log kayıtları tutulmaktadır.

Projemizde Van der Pol denkleminin çözümlerinin verimli bir şekilde saklanması ve yönetilmesi, veri bütünlüğünü korurken aynı zamanda kolay erişilebilirlik ve analiz imkânı sunan bir MySQL veri tabanı kullanılarak gerçekleştirilmektedir.

4. VERİLERİN İŞLENMESİ VE ANALİZİ

Bu bölüm, Van der Pol denkleminin çözümlerinin nasıl işlendiğini inceler. Projede kullanılan teknolojiler ve metodolojiler, elde edilen verilerin maksimum fayda sağlayacak şekilde işlenmesini amaçlamaktadır. Bu süreçler, veri toplama ve saklama aşamalarından sonra gelen önemli bir adımdır.

Projede elde edilen ham veriler, öncelikle işlenmek üzere bir dizi ön işlemden geçirilir. Bu işlemler, verilerin temizlenmesi, normalleştirilmesi ve gerekli hesaplamaların yapılması için gerekli adımları içerir. Node.js ortamında geliştirilen bir backend servisi, C++ ile üretilen sayısal çözümleri alır ve bunları JSON formatına dönüştürür. Bu dönüşüm, verilerin daha sonraki aşamalarda daha kolay işlenebilmesi için önemlidir. Dönüştürülen veriler, daha sonra uygun formatta MySQL veri tabanına veri analizi için kaydedilir. Parametre değişikliklerinin Van der Pol denkleminin çözümü üzerindeki etkilerini anlamak için ilişkili olarak çözüm iterasyonlarını ilişkilerine kolay ulaşmamıza olanak tanır. Sonuçların etkili bir şekilde sunulması için ise, verilerin listelenmesi teknikleri önemli bir rol oynar. Projede, çözüm sonuçlarının listelenmesi, kullanıcıların sonuçları daha kolay yorumlamalarına olanak tanır. Node.js ve çeşitli JavaScript kütüphaneleri (örneğin, DataTables) kullanılarak dinamik listeler/tablolar oluşturulur. Bu listeler/tablolar, web arayüzü üzerinden kullanıcılara sunulur ve kullanıcıların çözüm parametrelerine göre denklemin davranışını anlamalarını sağlar. Son olarak, veri işleme ve analiz süreçlerinin performansının sürekli olarak değerlendirilmesi ve optimizasyonu gerçekleştirilir. Bu değerlendirme, süreçlerin verimliliğini artırmak, hesaplama sürelerini azaltmak ve genel sistem performansını iyileştirmek için yapılır.

4.1. C++ Kodunun Çalıştırılması

Bu projede, Van der Pol denkleminin çözümü için geliştirilen C++ kodunun çalıştırılması, projenin temel taşlarından biridir. Dr. Coşar Gözükırmızı tarafından daha önce yazılmış olan bu C++ denklem çözüm kodu, proje kapsamında kullanılmıştır. C++ dili, yüksek performansı ve geniş matematiksel kütüphane desteği ile bilimsel hesaplamalar için sıklıkla tercih edilen bir programlama dilidir. Bu projede C++ kullanılmasının ana sebebi, kompleks matematiksel işlemlerin hızlı ve verimli bir şekilde gerçekleştirilmesini sağlamasıdır.

Kodun Çalıştırılması Süreci

C++ ile yazılan sayısal çözüm kodu, sistemin çekirdeğini oluşturur. Bu kod, Van der Pol denkleminin çeşitli parametreler altında nasıl davrandığını gözlemlemek için kullanılır. Kodun çalıştırılması, bir Shell script aracılığıyla otomatize edilmiştir. Bu script, belirlenen zaman aralıklarıyla veya kullanıcı tarafından yapılan bir tetikleme ile kodu çalıştırır, böylece kullanıcıların hesaplamaları manuel olarak başlatmalarına gerek kalmaz.

Shell script, sistemin farklı bileşenleri arasında köprü görevi görür. Script, C++ kodunu çalıştırır, derler ve sonuçları bir çıktı dosyasına kaydeder. Daha sonra, bu çıktılar Node.js tarafından işlenmek üzere okunur ve uygun formatlara dönüştürülür.

Kodun Optimizasyonu ve Güvenliği:

C++ kodunun optimizasyonu, hesaplama sürelerini minimize etmek ve sistem kaynaklarını etkin bir şekilde kullanmak için kritik öneme sahiptir. Kod, çeşitli test senaryoları üzerinden değerlendirilmiş ve en yüksek verimliliği sağlamak üzere ayarlanmıştır. Ayrıca, kodun güvenliği de büyük önem taşır; hatalı veri girişlerine karşı korumalar içerir ve olası hata durumlarında sistemin stabil kalmasını sağlayacak şekilde default parametreler ile tasarlanmıştır.

Süreçlerin İzlenmesi ve Raporlanması:

Kodun çalıştırılması sırasında, sistem performansını izlemek ve süreçlerin düzgün işlediğinden emin olmak için kapsamlı bir loglama mekanizması devreye alınmıştır. Bu loglar, sistemin hangi aşamada olduğunu, herhangi bir hata durumunu ve işlem sürelerini detaylı bir şekilde raporlar. Bu bilgiler, sistemin sürekli iyileştirilmesi ve optimizasyonu için değerlidir.

4.2. Kodun Çıktılarının Pars Edilmesi (Ayrıştırılması) ve Yönetilmesi

Projede kullanılan C++ kodu tarafından üretilen sayısal çözüm sonuçları, başlangıçta .txt formatında kaydedilir. Bu çıktılar, sistemin veri işleme bileşeni tarafından ele alınarak daha işlenebilir bir formata dönüştürülür. Bu süreç, çıktıların JSON formatına pars edilmesini ve ardından bu verilerin veri tabanına kaydedilmesini

kapsar. Bu bölüm, bu dönüşüm süreçlerini ve veri yönetimini detaylı bir şekilde açıklamaktadır.

Çıktıların Pars Edilmesi:

C++ uygulamasından elde edilen output.txt dosyası, çözümle ilgili ham verileri içerir. Bu dosya, Node.js tarafından kullanılan bir script aracılığıyla okunur. Script, ham verileri alır ve bu verileri sistem tarafından daha kolay işlenebilen bir yapıya, yani JSON formatına dönüştürür. JSON formatı hem insanlar hem de makineler tarafından kolayca okunabilir ve ayrıştırılabilir olması sebebiyle tercih edilir. Bu dönüşüm işlemi, veri akışını standardize eder ve farklı sistem bileşenleri arasında veri entegrasyonunu kolaylaştırır. Bu pars işlemi sırasında, veriler ayrıca doğrulanır ve temizlenir. Bu, olası formatlama hatalarını veya eksik veri girdilerini tespit etmek ve düzeltmek için önemlidir. Ayrıca, veri setindeki her bir öğenin doğru şekilde etiketlendiğinden ve uygun veri tiplerine sahip olduğundan emin olunur.

Veri Yönetimi:

JSON formatına dönüştürüldükten sonra, veriler bir API aracılığıyla MySQL veri tabanına aktarılır. Veri tabanı, verileri çeşitli tablolar arasında uygun şekilde ilişkilendirir, böylece kullanıcıların çeşitli sorgular yapmalarına olanak tanır. Veri yönetimi süreçlerinde, veri bütünlüğü ve güvenliği kritik öneme sahiptir. Veri tabanı katmanında uygulanan referans bütünlüğü kısıtlamaları ve erişim denetimi gibi güvenlik politikaları, verilerin yetkisiz erişim ve manipülasyona karşı korunmasını temin eder. Ek olarak, artan ve tam yedekleme stratejileri ile verilerin düzenli olarak yedeklenmesi sağlanır; bu yöntem, beklenmedik sistem arızaları durumunda veri bütünlüğünün korunması ve veri kaybının önlenmesi için etkin bir kurtarma çözümü sunar.

Performans İzleme:

Verilerin pars edilmesi ve yönetilmesi süreçlerinin performansı, sistem logları ve izleme araçları kullanılarak sürekli olarak değerlendirilir. Bu izleme, süreçlerin verimliliğini artırmak, gecikmeleri azaltmak ve kullanıcı deneyimini iyileştirmek için potansiyel iyileştirme alanlarını belirlemeye yardımcı olur.

Kodun Çıktılarının Pars Edilmesi ve Yönetilmesi:

Bu süreçler, projenin genel veri yönetim stratejisinin ve performansının bir yansımasıdır ve projenin başarısında önemli bir role sahiptir. Bu yapısal yaklaşım, projenin bilimsel ve teknik hedeflerine ulaşmasında kritik bir faktördür.

4.3. Verilerin JSON Formatında İşlenmesi ve Veri Tabanında Saklanması

C++ kodundan çıkan ham veriler (output.txt), ilk olarak işlenebilir bir formata dönüştürülmesi gerektiğinde, bu veriler JSON formatına (output.json) çevrilir. JSON (JavaScript Object Notation), veri alışverişi için hafif, okunması kolay ve insanlar tarafından yazılabilir bir format sağlar. Bu dönüşüm, Node.js tabanlı bir script kullanılarak gerçekleştirilir, script ham verileri okur ve her bir veri öğesini JSON objelerine dönüştürür. Bu objeler, denklemin parametrelerini, başlangıç değerlerini ve hesaplama sonuçlarını içerir.

JSON formatındaki veriler hem JavaScript hem de diğer programlama dillerinde kolayca kullanılabilir ve bu özellikleri sayesinde web tabanlı uygulamalar ile RESTful API'ler için idealdir. Bu formatın kullanımı, verilerin sistem genelinde sorunsuz bir şekilde entegre edilmesini ve işlenmesini sağlar. JSON verileri, MySQL veri tabanına kaydedilmeden önce, veri ilişkileri ve bütünlüğü göz önünde bulundurularak yapılandırılır. Veri tabanına eklenen bu veriler, önceden tanımlanmış şemalarla uyumlu hale getirilir, bu da verilerin güvenilir ve tutarlı bir şekilde saklanmasını sağlar.

Bu süreçte veri bütünlüğü, veri tabanı tarafından uygulanan kısıtlar ve ilişkisel bütünlük kuralları ile sağlanır. Ayrıca, veri güvenliği ve erişim kontrolü, yetkilendirme mekanizmaları ve şifreleme teknikleri kullanılarak güçlendirilir. Verilerin veri tabanında saklanması, uzun vadeli depolama, hızlı erişim ve karmaşık sorgulamalar için ideal bir çözüm sunar ve büyük veri setleri üzerinde etkin analizler yapılmasına olanak tanır.

Verilerin JSON formatında işlenmesi ve veri tabanında saklanması süreçleri sırasında, performans ve güvenlik önemli önceliklerdir. Veri transferi ve işleme hızı, sistem kaynaklarının etkin kullanımıyla optimize edilir. Güvenlik, veri sızıntılarını

önlemek ve verilerin yetkisiz erişime karşı korunmasını sağlamak için kritik bir bileşendir. Bu, güçlü kimlik doğrulama protokolleri, veri şifreleme ve güvenli API tasarımı ile sağlanır.

4.4. Veri Tabanı İşlemleri ve SQL Sorguları

Projemizde kullanılan veri tabanı, çeşitli tabloları içerir ve bu tablolar arasında ilişkisel bağlantılar kurulmuştur. Her bir tablo, Van der Pol denkleminin çözüm süreçlerinden elde edilen farklı türdeki verileri saklar; örneğin, hesaplama parametreleri, kullanıcı girdileri ve çözüm sonuçları. Veri tabanı şeması, veri bütünlüğünü ve verimli sorgu performansını sağlamak üzere dikkatle tasarlanmıştır.

SQL sorguları, veri eklemek, güncellemek, silmek ve çekmek için kullanılır. Bu sorgular, veri tabanına kaydedilen veriler üzerinde çeşitli işlemler yapılmasını sağlar ve kullanıcıların ihtiyaçlarına göre özelleştirilmiş bilgileri elde etmelerine olanak tanır.

4.4.1. Veri Tabanı Yapısı ve İlişkisel Bağlantılar

Temel veri alanları için her bir denkleme ait başlangıç parametreleri (mu, baslangic1, baslangic2), bu parametrelerin benzersiz bir kombinasyonu olarak tanımlanır. Bu değerler, denklemin her çözümünde başlangıç koşullarını ve denklemin kendine has parametrelerini belirler.

4.4.2. İlişkisel Yapı ve Saklama

“ode” tablosu, her bir denklemin parametrelerini ve çözümlerini saklar. Bu tablo, “iteration” ve “rho" tabloları ile ilişkilendirilir, burada her odeId'ye karşılık gelen iterasyon sayıları ve rho değerleri saklanır.

4.4.3. SQL Sorguları ve Optimizasyonları

SQL sorguları kullanılarak, çözümle ilgili veriler çeşitli kriterlere göre sorgulanabilir. Örneğin, belirli bir mu değeri ve başlangıç koşulları için tüm çözüm iterasyonlarını sorgulamak mümkündür.

4.4.4. Güvenlik ve Erişim Kontrolleri

Verilerin güvenliği, erişim izinleri ve kullanıcı rolleri ile sağlanır. Bu kontroller, verilerin yetkisiz erişimlere karşı korunmasını ve veri bütünlüğünün muhafazasını garantiler.

4.4.5. Uygulama Arayüzü ve Kullanıcı Etkileşimi

Kullanıcı arayüzü üzerinden parametre değişiklikleri yapıldığında, ilgili C++ kodu Shell script aracılığıyla derlenir ve sonuçlar txt formatında kaydedilir. Node.js, bu txt dosyalarından verileri pars ederek JSON formatında yeni dosyalar oluşturur ve bu verileri MySQL veri tabanına aktarır. Eğer kullanıcı farklı iterasyon parametreleri girerse ve veri tabanında kayıtlı iterasyon sayısı yetersizse, eksik iterasyonlar hesaplanır ve ilgili tablolara eklenir. Bu süreç, veri tabanı işlemleriyle entegre şekilde çalışarak kullanıcının deneyimini ve veri yönetimini optimize eder. Sistem kullanımına 6. bölümde detaylı olarak değinilmiştir.

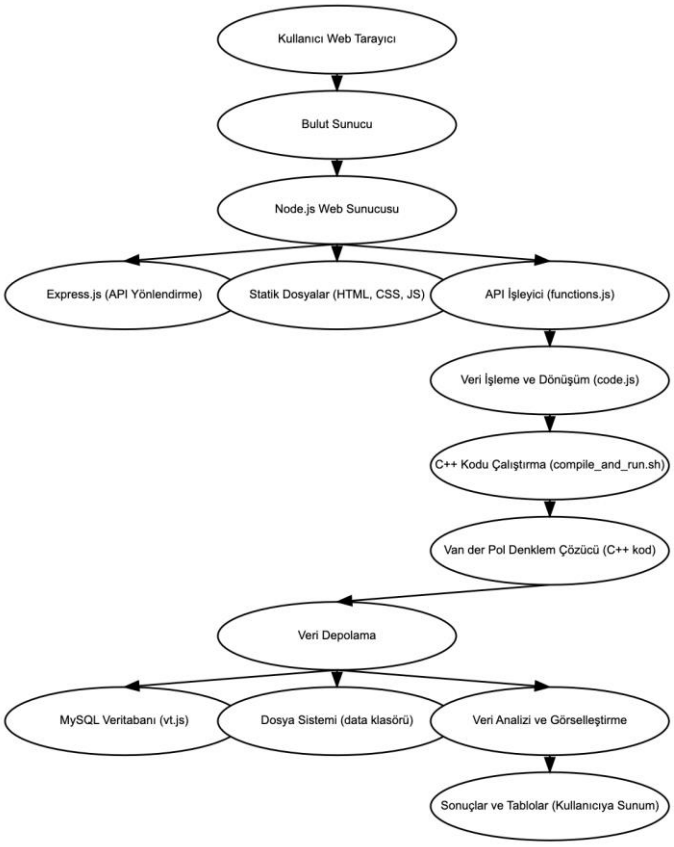
SQL sorguları ise, veri toplama ve işleme işlemlerinin temelini oluşturur. Örneğin, çözüm parametrelerine göre filtrelenmiş sonuçları çekmek için SELECT sorguları kullanılır. Bu sorgular, belirli kriterlere göre veri setlerini sıralamak ve gruplamak için de kullanılabilir. INSERT sorguları, yeni verilerin veri tabanına eklenmesi sürecinde kullanılırken, UPDATE ve DELETE sorguları, mevcut verilerin güncellenmesi veya silinmesi işlemlerini gerçekleştirir.

```
SELECT * FROM ode WHERE parameter = 'specific_value';

INSERT INTO ode (Id, value) VALUES (1, 'new_value');

UPDATE ode SET value = 'updated_value’WHERE Id = 1;

DELETE FROM ode WHERE Id = 1;
```

SQL sorgularının kullanımı, veri bütünlüğünün korunmasını da içerir. Veri tabanı, foreign key kısıtlamaları ve diğer bütünlük kuralları ile donatılmıştır, bu da veriler arasındaki tutarlılığı ve doğruluğu garanti eder.

5. WEB UYGULAMASI VE KULLANICI ARAYÜZÜ

Bu bölümde, Van der Pol denkleminin olasılıksal evrim yöntemi ile çözümü için geliştirilen web uygulamasının yapısı ve kullanıcı arayüzü ele alınmaktadır. Proje, Node.js kullanılarak geliştirilmiş olup, kullanıcı arayüzü modern web teknolojileri ve araçlarıyla tasarlanmıştır.



Şekil 5. Projenin İşleyiş Şeması

Kaynak: Yazar tarafından oluşturulmuştur.

Kullanıcı Arayüzü Tasarımı şu şekildedir:

HTML ve CSS:

Kullanıcı arayüzü, index.html dosyasında tanımlanmıştır ve Bootstrap 5 bileşenleri kullanılarak estetik ve kullanıcı dostu bir tasarım sağlanmıştır. Bu tasarım, web uygulamasının yanıt verme hızını ve etkileşimli özelliklerini güçlendirir.

JavaScript ve Node.js:

index.js ve diğer JavaScript dosyaları, web uygulamasının dinamik özelliklerini yönetmek için kullanılmaktadır. Bu scriptler, kullanıcı etkileşimlerini işleyerek, veri isteklerini ve uygulama içi navigasyonu destekler.

Veri Yönetimi ve Görüntüleme

DataTable Entegrasyonu:

Verilerin görsel olarak listelenmesi için DataTables kütüphanesi kullanılmıştır. Bu kütüphane, büyük veri setlerinin etkili bir şekilde işlenmesi ve kullanıcıya sunulması için gelişmiş özellikler sunar. Veriler, kullanıcı tarafından sıralanabilir ve filtrelenebilir hale getirilmiştir, bu da işleme ve erişilebilirlik açısından değer katar.

Dinamik Veri İşlemleri:

functions.js ve vt.js dosyaları, veri tabanı işlemlerini yönetmek için kullanılmaktadır. Bu dosyalar, veri ekleme, güncelleme, silme ve sorgulama gibi işlemleri dinamik olarak yönetir, böylece uygulama verimli ve güncel kalır.

Güvenlik ve Erişim Kontrolleri:

Erişim Güvenliği:

Uygulama, kötü niyetli erişimlere karşı korumak için güvenlik önlemleri ile donatılmıştır. CORS (Cross-Origin Resource Sharing) politikaları ve güvenlik token'ları, uygulamanın sadece yetkili kullanıcılar ve sistemler tarafından erişilebilir olmasını sağlar.

5.1. Kullanıcı Arayüzü Tasarımı

Bu bölüm, Van der Pol denkleminin çözümleri için geliştirilen web uygulamasının kullanıcı arayüzü tasarımını detaylandırmaktadır.

HTML ve CSS Kullanımı:

Kullanıcı arayüzü, Bootstrap 5 bileşenleri kullanılarak oluşturulmuştur. Bu çerçeve, duyarlı tasarım özellikleri sunar, böylece uygulama farklı ekran boyutları ve cihazlarda etkili bir şekilde görüntülenebilir. Bootstrap'ın bileşen kitaplığı, butonlar,

formlar ve tablolar gibi kullanıcı etkileşim elemanlarını içerir, bu da arayüzün estetik ve fonksiyonel olmasını sağlar.

JavaScript ve Uygulama Mantığı:

index.js ve diğer JavaScript dosyaları, arayüzün dinamik öğelerini kontrol eder. Bu scriptler, kullanıcı etkileşimlerine yanıt olarak uygulama içi veri akışını yönetir ve API isteklerini işler.

DataTables kütüphanesi kullanılarak oluşturulan veri tabloları, kullanıcılara çözüm verilerini sıralama, filtreleme ve inceleme imkânı tanır. Bu özellik, büyük veri setleri üzerinde hızlı ve etkili analiz yapılmasını kolaylaştırır.

Kullanıcı Deneyimi ve Erişilebilirlik:

Kullanıcı arayüzü, kullanıcıların sisteme kolayca adapte olmalarını ve gerekli işlevlere hızlı erişim sağlamalarını destekleyecek şekilde tasarlanmıştır. Ayrıca, kullanıcıların sistemi anlamalarını ve etkili kullanmalarını sağlayacak yardım dokümantasyonu ve arayüz ipuçları entegre edilmiştir.

Arayüz, kullanıcılar için uygun erişim kontrolleri ile donatılmıştır. Bu, sistem üzerinde işlem yapan kullanıcıların IP ve User Agent bilgileri loglanır.

5.2. Dinamik İçerik Yönetimi

Bu bölümde, Van der Pol denkleminin çözümleri için geliştirilen web uygulamasının dinamik içerik yönetim sistemine dair detaylar ele alınmaktadır. Uygulama, kullanıcı etkileşimini üst düzeye çıkaracak şekilde tasarlanmıştır ve bu, kullanıcıların veri setleri üzerinde sorunsuz bir şekilde işlem yapabilmesini sağlar. Dinamik yapı sayesinde uygulama hem hızlı hem de esnek bir kullanım sunar.

Node.js ile Güçlü ve Esnek Sunucu Yönetimi

Uygulamanın omurgasını oluşturan Node.js, arka planda veri işleme görevlerini hızlı ve güvenilir bir şekilde yerine getirir. Bu altyapı, kullanıcı isteklerine anında yanıt verirken, veri akışını da etkin bir şekilde yönetir. Node.js ile oluşturulan sunucu tarafı yapı, uygulamanın sorunsuz çalışması ve yüksek performans göstermesi için optimize edilmiştir.

Dinamik Veri Yönetimi: Anlık İşlemler ve Güncellemeler

Uygulama, veri yönetiminde esnekliği ve dinamikliği ön planda tutar. Functions.js, vt.js ve code.js gibi JavaScript dosyaları, kullanıcının yaptığı her bir değişikliği anında veri tabanına yansıtır. Bu, kullanıcıya verileri ekleme, güncelleme veya silme işlemlerinde maksimum kontrol sunar ve sistemin verilerle nasıl etkileşime girdiğini daha anlaşılır hale getirir.

Kullanıcı Deneyimini Zenginleştiren Dinamik İçerik Sunumu

Kullanıcı arayüzü, interaktif bir deneyim sunacak şekilde tasarlanmıştır. Index.js, kullanıcıların arayüzde yaptığı işlemleri hemen algılar ve dinamik olarak yanıt verir. Örneğin, veri sorgulama veya filtreleme işlemleri, anında ekranda güncellenir ve kullanıcıya anlık geri bildirim sağlar. Bu, kullanıcıların uygulamayı daha verimli kullanmalarını sağlar.

DataTables Entegrasyonu ile Güçlü Veri Yönetimi

Uygulamanın veri listeleme ve yönetim işlevleri, DataTables kütüphanesi ile entegre edilmiştir. Bu entegrasyon, büyük veri setlerini yönetmeyi kolaylaştırır. Kullanıcılar, verileri istedikleri parametrelerle sıralayabilir, arayabilir ve bu süreçleri hızlı bir şekilde gerçekleştirebilir. DataTables, kullanıcıya sunduğu esneklik ve hız ile uygulamanın en güçlü bileşenlerinden biridir.

Güvenlik ve Performans: En Üst Düzeyde Koruma ve Verimlilik

Uygulamanın güvenlik altyapısı, kullanıcı verilerini korumak ve yetkisiz erişimleri önlemek için titizlikle tasarlanmıştır. Sunucu tarafında uygulanan güvenlik protokolleri, veri bütünlüğünü sağlamakta ve kullanıcı bilgilerini güvence altına almaktadır. Ayrıca, performans iyileştirmeleri için veri önbelleğe alma ve yük dengeleme gibi teknikler kullanılmıştır. Bu sayede, uygulama yüksek trafik altında bile sorunsuz ve hızlı çalışmaya devam eder.

Etkili ve Kullanıcı Dostu Dinamik İçerik Yönetimi

Geliştirilen dinamik içerik yönetimi sistemi, uygulamanın verimliliğini artırmak ve kullanıcı deneyimini mükemmelleştirmek amacıyla tasarlanmıştır. Bu sistem, kullanıcıların veriler üzerinde tam kontrol sahibi olmasını sağlarken, veri yönetimini

kolaylaştırır ve hızlandırır. Uygulamanın dinamik yapısı, Van der Pol denkleminin çözümlerini incelemek isteyen kullanıcılar için büyük bir kolaylık sunmaktadır.

5.3. İstemci-Sunucu İletişimi

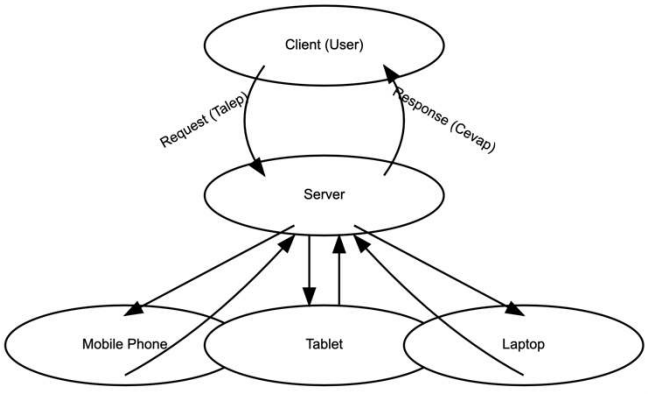
Bu bölümde, Van der Pol denklemlerinin çözümü için geliştirilen web uygulamasındaki istemci-sunucu iletişimi detaylandırılmaktadır. İstemci-sunucu modeli, modern web uygulamalarının temel iletişim yapısını oluşturmakta ve bu model, kullanıcıların (istemci) sunucu ile veri alışverişinde bulunmasını sağlamaktadır.

İstemci-Sunucu Modeli:

İstemci-sunucu modelinde, "istemci" genellikle bir web tarayıcısı veya mobil uygulama gibi kullanıcı tarafından kontrol edilen bir uygulamadır. "Sunucu" ise, veri saklama, işleme ve istemcilere bu verileri sunma işlemlerini gerçekleştiren bir sistemdir.

İletişim Süreci:

İstemci, bir web sayfası üzerinden belirli bir işlem yapılmasını talep eder (örneğin, bir form doldurarak veri sorgulama). Bu talep, HTTP protokolü aracılığıyla sunucuya gönderilir. Sunucu, bu talebi işler ve gerekli bilgileri işleyerek tekrar istemciye bir yanıt olarak döner.



Şekil 6. Sunucu İstek ve Cevap Şeması

Kaynak: Yazar tarafından oluşturulmuştur.

Sunucu Tarafı İşlemler:

Uygulamanın sunucu tarafı, Node.js kullanılarak geliştirilmiştir. Node.js, asenkron işlemleri destekleyen olay tabanlı bir yapıya sahiptir, bu da onu yüksek trafikli web uygulamaları için ideal kılar.

Express.js Framework'ü:

Express.js, Node.js için yalın bir web uygulama framework'üdür ve RESTful API'lerin geliştirilmesini kolaylaştırır. Bu framework, istemci-sunucu arasındaki iletişimde URL yönlendirmeleri ve HTTP isteklerinin yönetimi gibi görevleri üstlenir.

HTTPS Kullanımı:

Güvenli veri transferi, HTTPS protokolü aracılığıyla sağlanır. HTTPS, veri iletişimini şifreleyerek, istemci ve sunucu arasındaki bilgilerin güvenliğini artırır.

Erişim Kontrolü ve Doğrulama:

Kullanıcı kimlik doğrulama, yetkilendirme ve erişim kontrol mekanizmaları, istemci-sunucu iletişiminin güvenliğini sağlamak için uygulanır. Bu süreçler, kullanıcıların yetkilerine uygun işlemleri yapmalarını garanti eder ve yetkisiz erişimleri önler.

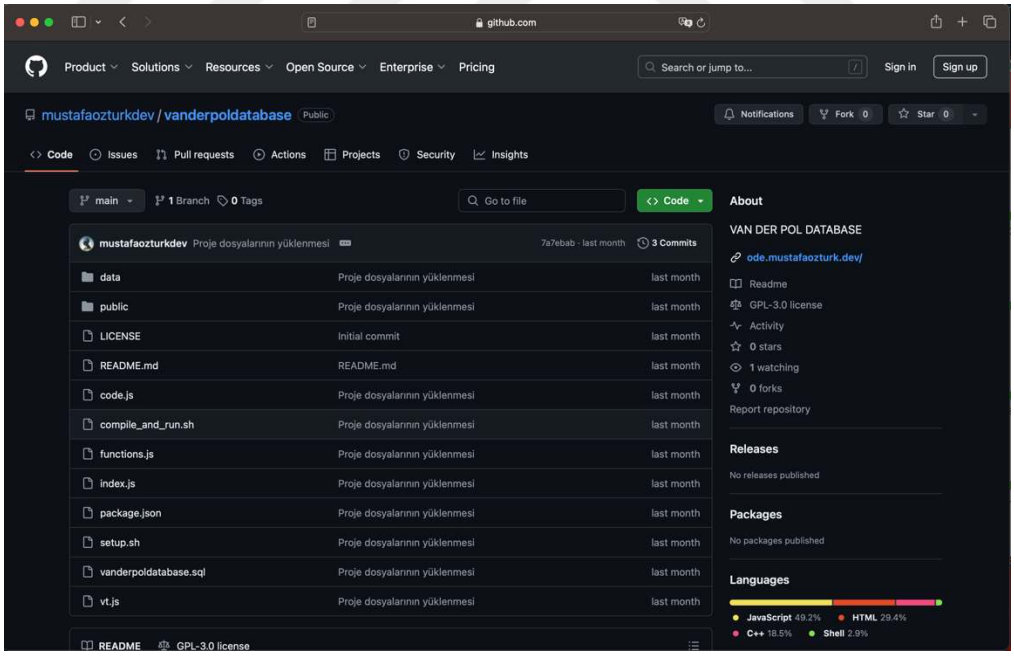
6. PROJENİN KULLANIMI, KURULUM REHBERİ VE GİTHUB HESABI

Bu bölümde, projenin kullanımına yönelik adımlar detaylı bir şekilde açıklanmaktadır. Kullanıcıların projeyi nasıl çalıştıracağı, hangi menü seçenekleriyle hangi işlemleri yapabilecekleri ve projenin GitHub üzerindeki kaynak kodlarına nasıl erişebilecekleri anlatılacaktır.

6.1. Proje Dosyalarına Erişim ve Kurulum

Proje, "https://ode.mustafaozturk.dev" adresinden erişilebilmektedir. Projede beş menü bağlantısı bulunmaktadır: Projenin kaynak kodları, "https://github.com/mustafaozturkdev/vanderpoldatabase" adresinde paylaşılmıştır. Bu GitHub deposu, proje kodlarını ve nasıl kurulacağını açıklayan bir README.md dosyasını içermektedir. Linux işletim sistemi için otomatik kurulum scripti de bu depoda mevcuttur. Kolaylıkla farklı işletim sistemlerine de kurulum yapılabilir.

Projenin yukarıda açıklanan menülerin detaylı kullanımı ve github deposunun ekran görüntüleri aşağıda verilmiştir.



Şekil 7. Github Hesabı Menü Bağlantısı

Kaynak: Yazar tarafından oluşturulmuştur.

6.1.1. Kurulum

https://github.com/mustafaozturkdev/vanderpoldatabase bağlantısındaki repoyu klonlayın: git clone. Proje dizinine gidin: cd repo_name (bilgisyara kaydettiğimiz klasör ismi) Kurulum script'ini root yetkisiyle çalıştırın: “sudo bash setup.sh” Kurulum tamamlandıktan sonra, projeyi başlatmak için: npm start komutunu girin. Proje localhost (yerel sunucu) üzerinden çalışacaktır.

Kurulum script'i (setup.sh) aşağıdaki işlemleri gerçekleştirir:

C++ kodunu derlemek ve çalıştırmak için gerekli paketleri yükler. MySQL'i kurar (eğer kurulu değilse). MySQL kullanıcısı ve veritabanını oluşturur, “vanderpoldatabase.sql” dosyasını import (içeri aktarır) eder. Node.js ve npm'i kurar (eğer kurulu değilse). Gerekli npm paketlerini yükler.

6.1.2. Gereksinimler

Linux işletim sistemi (Ubuntu 18.04 veya üzeri önerilir). Root erişimi ve İnternet bağlantısı (gerekli paketlerin yüklenmesi için)

6.1.3. Manuel Kurulum

Aşağıdaki adımları takip ederek projeyi manuel olarak kurabilir ve çalıştırabilirsiniz. Herhangi bir sorunla karşılaşmamak için adımların sırasıyla ve eksiksiz uygulanması önemlidir.

Gerekli Paketlerin Kurulumu (C++ Derleyicisi ve Kütüphaneler)

Linux için terminali açın ve aşağıdaki komutları sırayla çalıştırarak C++ geliştirme araçlarını ve gerekli kütüphaneleri yükleyin. build-essential: GCC ve diğer geliştirme araçlarını içerir. libgmp-dev: GNU Multiple Precision (GMP) kütüphanesi için geliştirme dosyalarını içerir.

libgmpxx4ldbl: GMP için C++ bağlama dosyalarını içerir.:

sudo apt-get update

sudo apt-get install build-essential libgmp-dev libgmpxx4ldbl

Windows için MinGW: GCC derleyicisini içerir. MSYS2: Bir paket yöneticisi ve bash ortamı sağlar. MinGW ve MSYS2'yi indirin ve kurun. MSYS2 terminalini açın ve aşağıdaki komutları çalıştırarak gerekli kütüphaneleri yükleyin:

pacman -S mingw-w64-x86_64-gcc mingw-w64-x86_64-gmp

macOS için terminali açın ve Xcode Komut Satırı Araçları'nı yükleyin: "xcode-select --install" komutunu çalıştırın. Homebrew paket yöneticisini yükleyin (eğer yüklü değilse): /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)" komutunu çalıştırın. Homebrew ile gerekli kütüphaneleri yükleyin: "brew install gmp" komutunu çalıştırın.

MySQL Kurulumu ve Yapılandırması

Linux'da MySQL sunucusunu yüklemek için "sudo apt-get install mysql-server" komutu çalıştırın. MySQL servisinin başlamasını ve otomatik olarak başlatılması için "sudo systemctl start mysql" ve "sudo systemctl enable mysql" komutlarını çalıştırın. MySQL güvenlik ayarlarını yapmak için "sudo mysql_secure_installation" komutunu çalıştırabilirsiniz. MySQL'e root "sudo mysql -u root" olarak bağlanın ve aşağıdaki komutlarla gerekli veritabanını ve kullanıcıyı oluşturun:

CREATE DATABASE IF NOT EXISTS vanderpoldatabase;

CREATE USER IF NOT EXISTS 'vanderpoldatabase'@'localhost' IDENTIFIED BY 'vanderpoldatabase';

GRANT ALL PRIVILEGES ON vanderpoldatabase.* TO 'vanderpoldatabase'@'localhost';

FLUSH PRIVILEGES;

Oluşturulan veri tabanına "vanderpoldatabase.sql" dosyasını içe aktarmak için "mysql -u vanderpoldatabase -pvanderpoldatabase vanderpoldatabase < /path/to/vanderpoldatabase.sql" komutunu çalıştırabilirsiniz. "/path/to/" kısmı "vanderpoldatabase.sql" dosyanızın bulunduğu yolu ifade ediyor.

Windows ve macOS'da MySQL'i yüklemek için resmi MySQL web sitesinden MySQL Community Server'ı indirin ve kurun. MySQL Workbench veya

terminal/komut istemcisini kullanarak root olarak bağlanın ve yukarıdaki Linux adımlarındaki SQL komutlarını çalıştırın.

"vanderpoldatabase.sql" dosyasını terminal veya komut istemcisinde aşağıdaki komutla içe aktarın:

"mysql -u vanderpoldatabase -pvanderpoldatabase vanderpoldatabase < C:\path\to\vanderpoldatabase.sql" komutunu çalıştırabilirsiniz. "/path/to/" kısmı (Windows için dosya yolu "C:\" gibi, macOS için "/Users/username/" gibi olmalıdır.)

Node.js ve npm Kurulumu

Linux'da Node.js ve npm'i kurmak için terminalde "sudo apt-get install nodejs" komutunu çalıştırın.

Windows'da Node.js'in resmi web sitesinden Windows Installer (LTS versiyonu) indirin ve yükleyin.

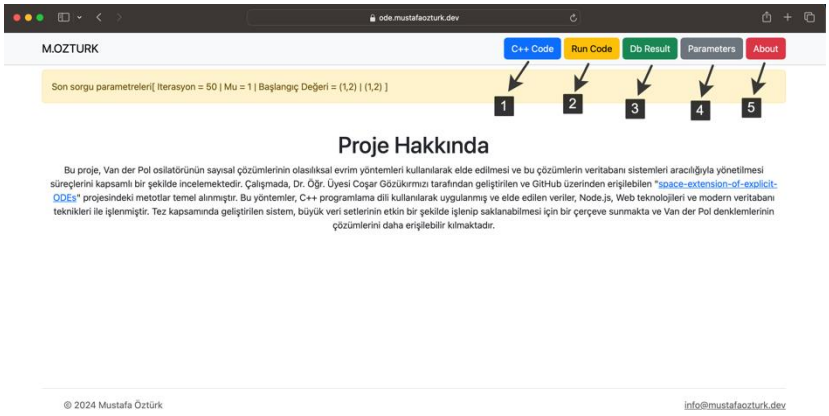
macOS'da Homebrew kullanarak Node.js ve npm'i yüklemek için "brew install node" komutunu çalıştırınız.

NPM Paketlerinin Kurulumu

Tüm İşletim Sistemlerinde proje dizinine gidin ve NPM paketlerini yüklemek için terminal veya cmd'de "npm install" komutu çalıştırın. Projeyi çalıştırmak için "npm start" komutunu çalıştırabilirsiniz. Proje "http://localhost:3000" sanal web sunucusunda çalışacaktır.

6.2. Programın Kullanım Rehberi

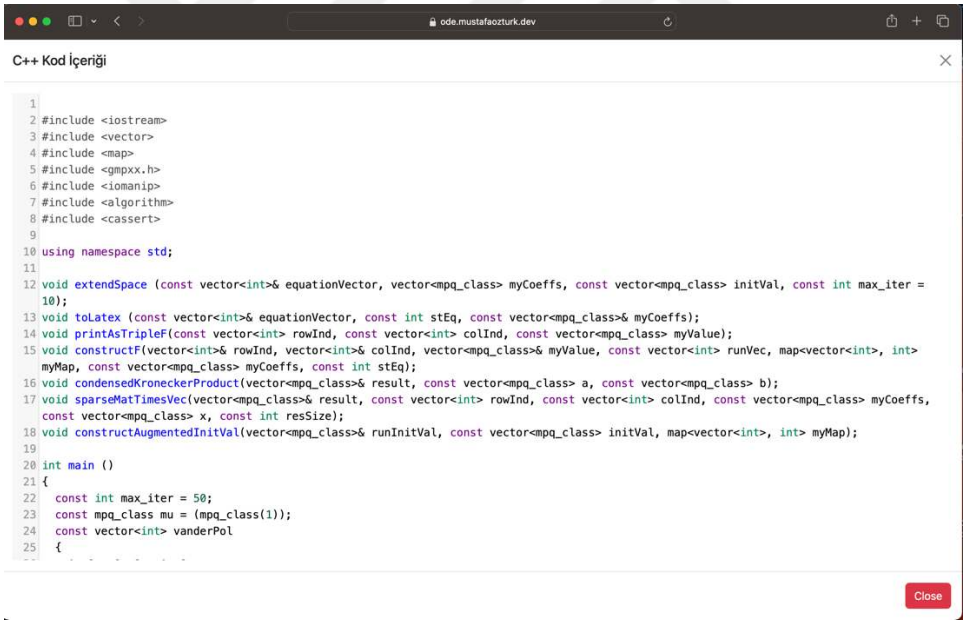
Bu bölümde, programın nasıl kullanılacağına dair kapsamlı bir rehber sunulmaktadır. Programın kullanıcı arayüzü, adım adım açıklamalar ve ekran görüntüleri ile detaylandırılmıştır.



Şekil 8. Proje Giriş Sayfası

Kaynak: Yazar tarafından oluşturulmuştur.

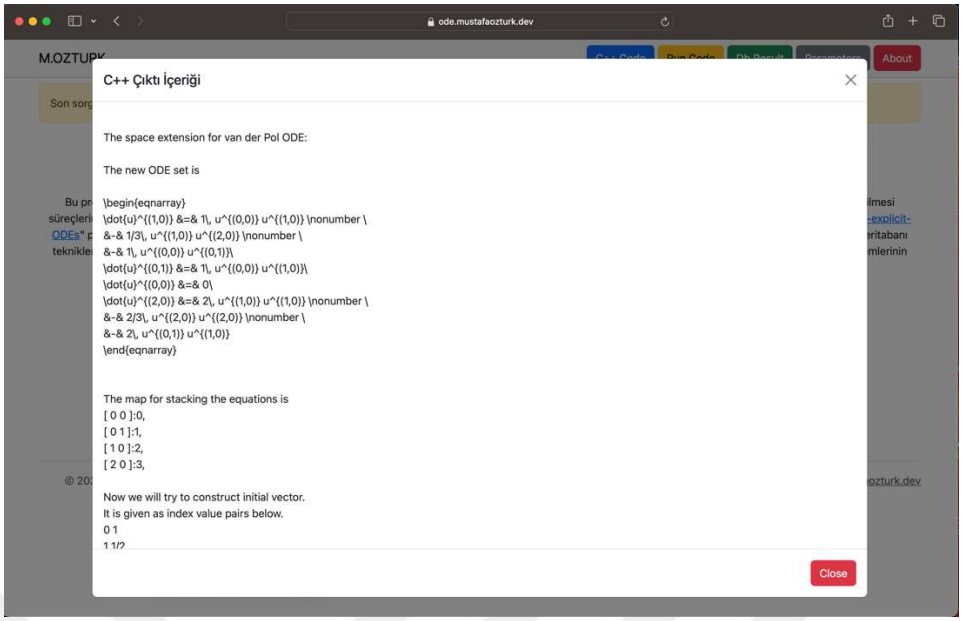
1 C++ Code: Bu bağlantı, mevcut parametrelerle oluşturulmuş C++ kodunu yalnızca okunabilir (read-only) modda gösterir. Kullanıcılar bu ekrandan C++ kodunun nasıl oluşturulduğunu inceleyebilirler.



Şekil 9. C++ Code Menü Bağlantısı

Kaynak: Yazar tarafından oluşturulmuştur.

2 Run Code: Bu bağlantı, atanmış parametrelerle C++ kodunu çalıştırır ve hesaplanan sonuçları veritabanına kaydeder. Eğer sonuçlar daha önce kaydedilmişse, tekrar kaydedilmez. Bu özellik, kullanıcının dinamik olarak kodu çalıştırmasını ve sonuçları veritabanında tutmasını sağlar.



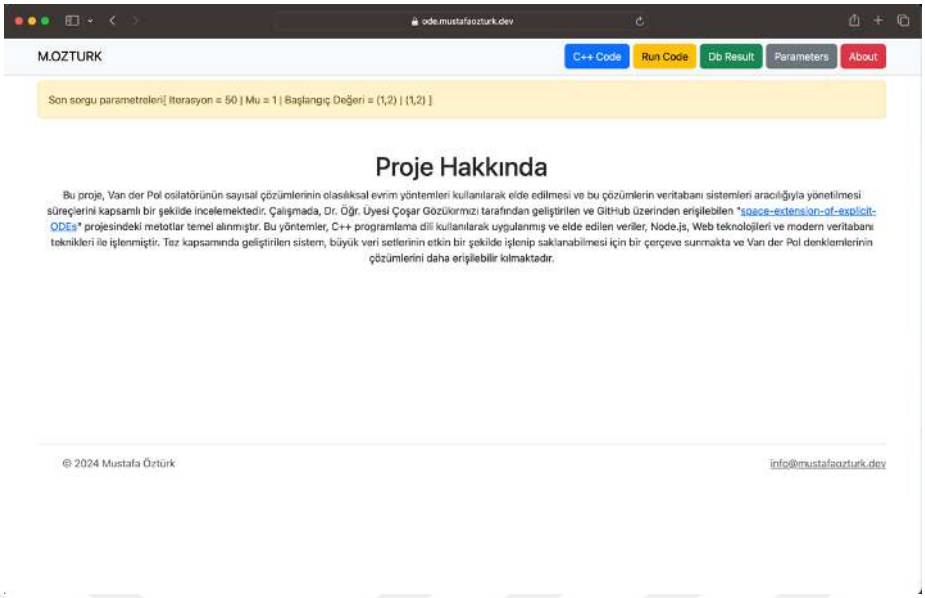
Şekil 10. Run Code Menü Bağlantısı

Kaynak: Yazar tarafından oluşturulmuştur.

3 Db Result: Bu bağlantı, projede daha önce çözülmüş denklemlere ait kayıtları listeler. Kullanıcılar bu ekran üzerinden sonuçları inceleyebilir ve kayıtlar arasında gezinebilirler. Bu bölüm aşağıda detaylandırılacaktır.

4 Parameters: Bu bağlantı, kullanıcıların mu değeri, başlangıç koşulları ve iterasyon sayısını değiştirebileceği parametre ekranını açar. Değişiklikler kaydedildikten sonra, "Run Code" menüsü üzerinden kod yeniden çalıştırılabilir. Bu bölüm aşağıda detaylandırılacaktır.

5 About: Bu bağlantı, proje hakkında kısa bir açıklama içerir. Projenin amacı, işlevselliği ve kullanım detaylarına dair genel bilgi bu ekranda sunulmaktadır.

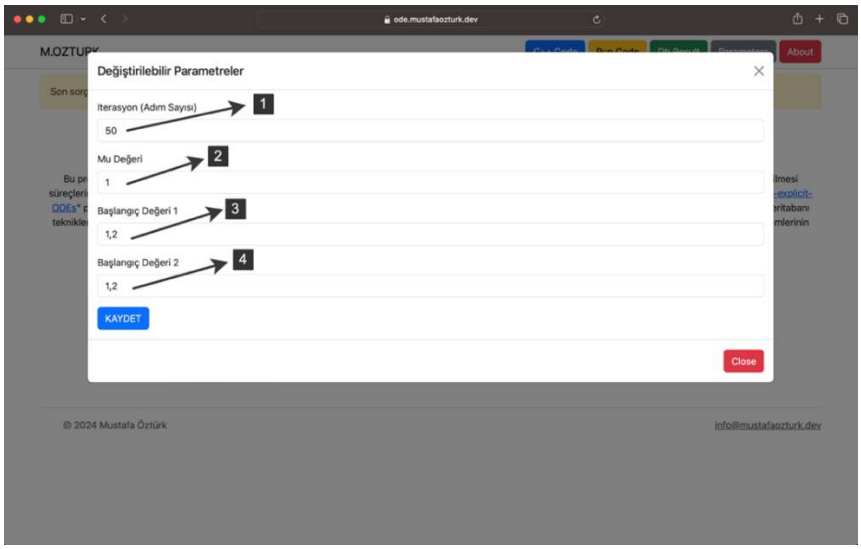


Şekil 11. About Menü Bağlantısı

Kaynak: Yazar tarafından oluşturulmuştur.

6.2.1. Parametre Girişi

Bu bölümde, programın kullanıcı arayüzünde Parameters bağlantısında yer alan "Değiştirilebilir Parametreler" bölümünde gerçekleştirilen parametre giriş işlemleri ayrıntılı olarak ele alınacaktır. Bu parametreler, denklemin çözüm sürecine doğrudan etki eden kritik girdilerdir ve her birinin doğru şekilde girilmesi, hesaplama süreçlerinin sağlıklı bir biçimde yürütülebilmesi açısından önemlidir.



Şekil 12. Parametre Girişi

Kaynak: Yazar tarafından oluşturulmuştur.

Şekil 12’de gösterilen numaralandırılmış alanların her birinin çalışma şekli aşağıda verilmiştir.

1 İterasyon (Adım Sayısı): İterasyon, denklemin çözümü esnasında gerçekleştirilecek tekrarlama sayısını belirlemektedir. Bu alana girilecek değerin tam sayı olması gerekmektedir. İterasyon sayısı, çözümün doğruluğunu artırmakla birlikte, aynı zamanda hesaplama süresinin de uzamasına neden olabilmektedir. Kullanıcılar, çözümün doğruluğu ile hesaplama süresi arasındaki dengeyi göz önünde bulundurarak uygun bir iterasyon sayısı seçmelidir.

2 Mu Değeri (μ): Mu değeri (μ), denklemin doğrusal olmayan dinamik davranışlarını belirleyen bir parametredir. Bu alana girilecek olan değer, denklemin çözümünde önemli bir rol oynamaktadır. Kesirli değerlerin girilmesi gerektiğinde, belirli bir format kullanılmalıdır; örneğin, 1/2 değeri 1,2 olarak girilmelidir. Bu format, kesir ayracı olarak virgülün kullanıldığı bir gösterimdir ve bu biçim, sistemin doğru bir şekilde çalışması için gereklidir. Kullanıcıların bu formatı doğru şekilde uygulamaları, denklemin çözüm sürecinin sağlıklı bir şekilde yürütülmesini sağlayacaktır.

3 Başlangıç Değeri 1: Başlangıç Değeri 1, denklemin birinci başlangıç koşulunu temsil etmektedir. Başlangıç koşulları, denklemin çözümü esnasında sistemin başlangıçtaki durumunu belirler ve bu sebeple girilen değerlerin doğruluğu büyük önem taşır. Kesirli değerler gerektiğinde, bu alanın 1,2 formatında doldurulması zorunludur. Virgülün kesir ayracı olarak kullanılması, sistemin girdileri doğru şekilde işlemesi için elzemdir. Başlangıç koşullarının doğru bir şekilde belirlenmesi, denklemin çözüm sürecinde elde edilecek sonuçların geçerliliği açısından kritik bir rol oynamaktadır.

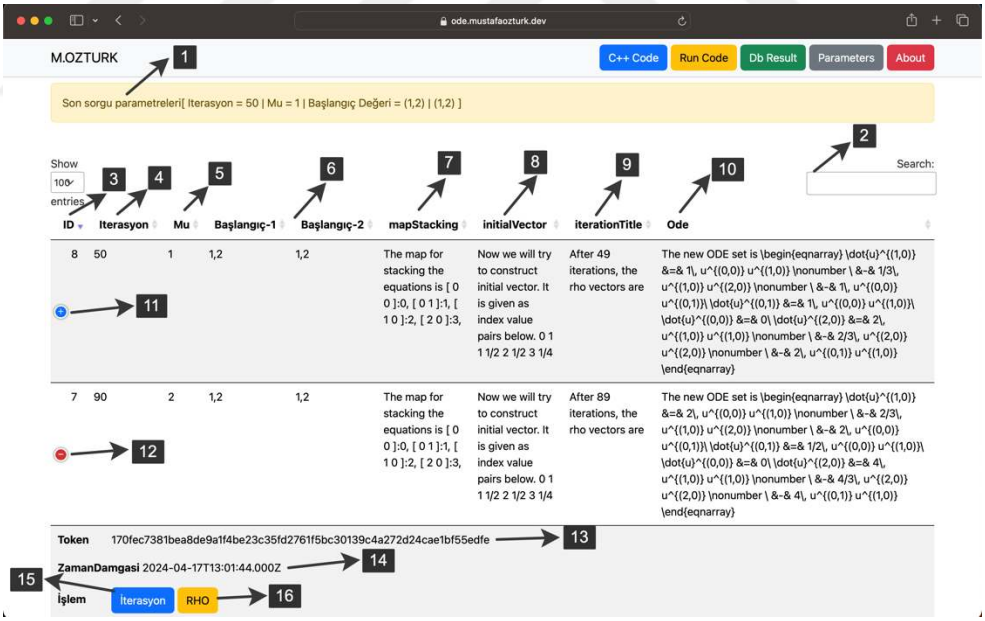
4 Başlangıç Değeri 2: Başlangıç Değeri 2, denklemin ikinci başlangıç koşulunu ifade eder ve sistemin başlangıçtaki durumunu belirleyen bir diğer önemli parametredir. Tıpkı Başlangıç Değeri 1'de olduğu gibi, kesirli girişler için 1,2 formatı kullanılmalıdır. Bu formatın doğru bir şekilde uygulanması, hesaplama süreçlerinin doğruluğunu sağlamak açısından gereklidir.

Kullanıcılar, bu parametreleri girdikten sonra "KAYDET" butonuna tıklayarak girdileri onaylamalı ve sistemin hesaplama sürecine hazırlanmasını sağlamalıdır.

Parametre girişinde dikkat edilmesi gereken en önemli husus, girdilerin uygun formatta (özellikle kesirli değerlerde virgül ayracı ile) girilmesidir. Bu ayracın doğru bir şekilde kullanımı, sistemin bu değerleri doğru bir biçimde yorumlaması ve işlemesi için gereklidir. Değişiklikler kaydedildikten sonra, "Run Code" menüsü üzerinden kod yeniden çalıştırılmalıdır. Girilen parametrelere ait sonuçlar "Run Code" menüsünden C++ kodu çalıştırıldıktan sonra üretilecektir. Sonuçlar veri tabanında saklanıyor olup girilen parametrelerin varsa daha önceki kayıtlı sonuçlarına “Db Result” bağlantısından ulaşılabilinmektedir.

6.2.2. Sonuçların Analizi

Bu bölümde, programın kullanıcı arayüzünde “Db Result” bağlantısında yer alan Van der Pol denkleminin olasılıksal evrim yöntemi ile çözümünden elde edilen sonuçların analizi ve sunumu detaylı olarak ele alınmaktadır. Geliştirilen web uygulaması, kullanıcılara denklem çözümlerini kapsamlı bir şekilde inceleme ve karşılaştırma imkânı sunmaktadır.



Şekil 13. Db Result Bağlatısı Denklem Sonuçları

Kaynak: Yazar tarafından oluşturulmuştur.

Şekil 13’de gösterilen numaralandırılmış alanların her birinin çalışma şekli aşağıda verilmiştir.

1 Kullanıcı Arayüzü: Arayüzün üst kısmında, menü başlığı altında, en son kullanılan parametreler gösterilmektedir. Bu, kullanıcıların mevcut analiz bağlamını hızlıca kavramasını sağlar. Örneğin, "Son sorgu parametreleri[Iterasyon = 50 | Mu = 1 | Başlangıç Değeri = (1,2) | (1,2)]" şeklinde sunulan bu bilgi, önceki hesaplama işlemlerinde en son hangi parametrelerin kullanıldığını göstermektedir.

2 Arama Fonksiyonu: Sonuçlar tablosunun üst kısmında yer alan arama çubuğu, kullanıcıların geniş veri setleri içerisinde hızlı bir şekilde arama yapmasına olanak tanır. Bu özellik, tüm alanlarda eş zamanlı arama yapabilme yeteneği ile kullanıcı deneyimini önemli ölçüde artırır. Arama fonksiyonu, kullanıcıların belirli çözüm parametrelerini veya sonuçları hızlıca bulmasını sağlar.

3-6 Temel Parametreler: Sonuçlar tablosunda, her bir çözüm için temel parametreler açıkça gösterilmektedir:

ID: Her bir çözüm için benzersiz bir tanımlayıcıdır. Bu ID, veritabanında her çözümü eşsiz bir şekilde tanımlar ve çözümün hızlıca geri çağrılmasını sağlar.

İterasyon: Denklemin çözümünde kullanılan son iterasyon sayısını gösterir. İterasyon sayısı, çözümün doğruluğunu ve detayını belirleyen önemli bir faktördür.

Mu: Denklemin çözümünde kullanılan μ değerini ifade eder. μ değeri, Van der Pol denkleminin doğrusal olmayan dinamiklerini etkileyen bir parametredir.

Başlangıç-1 ve Başlangıç-2: Denklemin başlangıç koşullarını gösterir.

7 MapStacking: Bu sütun, belirli bir işlevin çözümünün dizinin hangi indeksinde tutulduğunu gösterir. Bir nevi adres defteri gibi çalışan bu yapı, olasılıksal evrim yöntemi sırasında her bir denklemin veya işlevin sonuçlarının belirli indekslerde saklanmasını sağlar. Örneğin, "The map for stacking the equations is [0]:0, [0 1]:1, [1 0]:2, [2 0]:3" ifadesi, C++ kodu içerisinde belirlenen çözüm stratejisini yansıtarak, bu indekslerin hangi işlevlere karşılık geldiğini ve nasıl adreslendiğini gösterir.

8 Initial Vector: Başlangıç vektörünün nasıl oluşturulduğunu gösterir. Örneğin, "Now we will try to construct initial vector. It is given as index value pairs below. 0 1 1/2 2 1/2 3 1/4" ifadesi, başlangıç koşullarının olasılıksal evrim yönteminde nasıl vektörleştirildiğini ve bu vektörlerin çözüm sürecine nasıl dahil edildiğini açıklar. Bu yapı, çözümün doğruluğunu etkileyen ilk adımı oluşturur.

9 Iteration Title: İterasyon sürecinin son durumunu özetler. Örneğin, "After 49 iterations, the rho vectors are" ifadesi, 0 – 49, 0 dahil 50 iterasyon sonrasında elde edilen rho vektörlerinin durumunu belirtir. Rho vektörleri, olasılıksal evrim yönteminde çözümün yakınsamasını kontrol etmek ve sonuçların doğruluğunu sağlamak için kullanılan önemli bir parametredir.

10 Ode: Bu sütun, denklemin LaTeX formatında uzay genişletimi adımından sonraki halini içerir. Bu matematiksel gösterim, Van der Pol denkleminin olasılıksal evrim yöntemi ile uzay genişletilerek nasıl ifade edildiğini detaylı bir şekilde sunmaktadır. Örneğin, 8 ID'li denklem çözümüne ait sorgu parametreleri [Iterasyon = 50 | Mu = 1 | Başlangıç Değeri = (1,2) | (1,2)] ile elde edilen uzay genişletimi şu şekilde ifade edilmiştir:

$$\dot{u}^{(1,0)} = 1u^{(0,0)}u^{(1,0)} - \frac{1}{3}u^{(1,0)}u^{(2,0)} - 1u^{(0,0)}u^{(0,1)}$$

$$\dot{u}^{(0,1)} = 1u^{(0,0)}u^{(1,0)}$$

$$\dot{u}^{(0,0)} = 0$$

$$\dot{u}^{(2,0)} = 2u^{(1,0)}u^{(1,0)} - \frac{2}{3}u^{(2,0)}u^{(2,0)} - 2u^{(0,1)}u^{(1,0)}$$

Bu gösterim, problemin çözümünden önceki adım olan uzay genişletimi sürecini LaTeX formatında matematiksel olarak ifade eder.

11-12 Veri Görüntüleme Kontrolleri: Bu butonlar, kullanıcılara tablodaki ek sütunları gösterme veya gizleme imkânı sunar, böylece kullanıcılar ihtiyaç duydukları detay seviyesini ayarlayabilirler. Görüntülenen sütun sayısını kontrol etmek, verinin karmaşıklığını yönetmek açısından kullanıcıya esneklik sağlar.

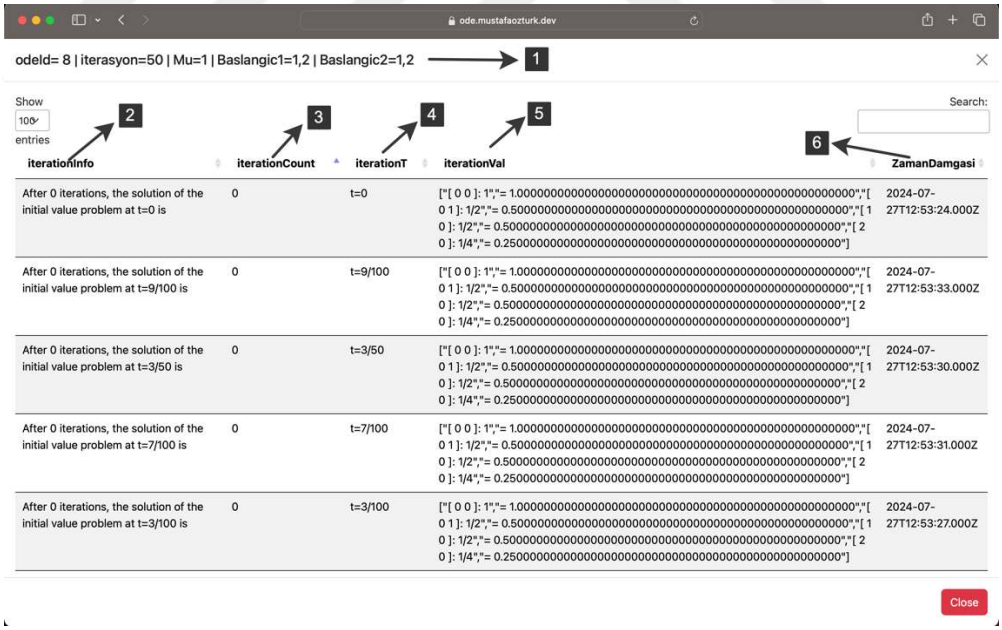
13 Güvenlik Tokeni: Veri tabanında saklı her çözüm için benzersiz bir token üretilir, bu da kayıtlı sonuçların güvenliğini ve izlenebilirliğini artırır. Bu token, her bir çözümün doğrulanmasını ve yetkisiz erişimlerin engellenmesini sağlar.

14 Zaman Damgası: Çözümün gerçekleştirildiği son zamanı gösterir. Bu, kullanıcının analizlerini zaman çizelgesi üzerinde takip etmesine olanak tanır ve elde edilen verinin güncelliğini değerlendirir.

15-16. Detaylı Analiz Butonları: "İterasyon" ve "RHO" butonları, kullanıcılara daha detaylı analiz imkânı sunar. RHO, olasılıksal evrim yönteminde çözümün yakınsamasını kontrol etmek için kullanılan önemli bir parametredir. Bu butonlar, kullanıcıların çözüm sürecine dair daha derinlemesine bilgi edinmelerini sağlar.

6.2.3. Sonuçların İterasyonları

Bu bölümde, Van der Pol denkleminin olasılıksal evrim yöntemi ile çözümünden elde edilen sonuçların iterasyon bazında detaylı analizi ele alınacaktır. İterasyon süreci, denklemin zamana bağlı olarak nasıl geliştiğini anlamak açısından kritik öneme sahiptir. Geliştirilen web uygulaması, kullanıcılara denklem çözümünün her bir adımını detaylı bir şekilde gözlemleme imkânı sunmaktadır. Aşağıda, arayüzde sunulan ve numaralandırılmış olan her bir alanın işlevi detaylı olarak açıklanmaktadır.



Şekil 14. Denklem Çözümüne Ait İterasyonlar

Kaynak: Yazar tarafından oluşturulmuştur.

1 Denklem Çözümüne Ait Parametrelerin Özeti ve OdeID: Bu alan, seçilen denklemin çözümünde kullanılan temel parametrelerin özetini ve bu çözümün veri tabanında kayıtlı olduğu odeId'sini göstermektedir. OdeID, çözümün veri tabanındaki

benzersiz tanımlayıcısıdır ve bu tanımlayıcı sayesinde çözüm sonuçlarına kolayca erişim sağlanabilir. Bu parametreler, denklemin çözüm sürecine başlangıçta hangi değerlerle giriş yapıldığını özetler.

2 IterationInfo: IterationInfo sütunu, her bir iterasyonda t parametresinin aldığı değerin açıklamasını içerir. C++ kodunda tanımlanan bu alan, belirli bir iterasyon aşamasında zaman parametresi t'nin hangi değerde olduğunu belirtir. Bu bilgi, sistemin evrim sürecini ve zamanla nasıl değiştiğini analiz etmek için kritik bir rol oynar. IterationInfo, aynı zamanda başlangıç değer problemi çerçevesinde, sistemin başlangıç koşullarına ne ölçüde bağlı olarak geliştiğini göstermektedir.

3 IterationCount: Bu sütun, çözümün hangi iterasyon aşamasında olduğunu belirtir. İterasyon sayısı, denklemin çözüm sürecindeki aşamalarını temsil eder ve sistemin her bir iterasyonda nasıl geliştiğini izlemeyi mümkün kılar. İterasyon sayısı arttıkça, sistemin daha fazla çözüm adımından geçtiği ve bu sayede daha hassas sonuçlar elde edildiği anlaşılmaktadır.

4 IterationT: IterationT sütunu, zaman parametresi t'nin aldığı değeri göstermektedir. Bu değer, her bir iterasyon adımında sistemin mevcut zamanını temsil eder. t parametresi, zamanın belirli bir anında sistemin hangi durumda olduğunu göstermek için kullanılır ve zamanla sistemin nasıl evrildiğini anlamak için belirli bir öneme sahiptir.

5 IterationVal: Bu sütun, en kritik bileşenlerden biri olarak, belirli bir t değerine karşılık gelen fonksiyon değerlerini içerir. Burada gösterilen veriler, t'nin belirli bir anında sistemin durumunu sayısal olarak ifade eder. Bu veriler, sistemin zamanla nasıl evrildiğini anlamak ve olasılıksal evrim yöntemi ile elde edilen sonuçların doğruluğunu değerlendirmek için kullanılır. Fonksiyon değerleri, her bir iterasyon adımında belirli başlangıç koşullarına ve iterasyon sayısına bağlı olarak elde edilir. Bu sayede, sistemin çözüm süreci boyunca nasıl davrandığı detaylı olarak incelenebilir.

6 Zaman Damgası: Zaman Damgası, her bir iterasyonun gerçekleştirilme zamanını gösterir. Bu bilgi, çözüm sürecinin hangi tarihte ve saatte veri tabanına kaydedildiğini gösterir.

6.2.4. Sonuçların Rho Değerleri

Bu bölümde, Van der Pol denkleminin olasılıksal evrim yöntemi ile çözüm sürecinde elde edilen rho değerleri analiz edilmektedir. Rho, olasılıksal evrim yönteminde çözümün yakınsamasını izlemek ve doğrulamak için kullanılan önemli bir parametredir. Aşağıda, arayüzde sunulan ve numaralandırılmış olan her bir alanın işlevi detaylı bir şekilde açıklanmaktadır.

Şekil 15. Denklem Çözümüne Ait Rho Değerleri

Kaynak: Yazar tarafından oluşturulmuştur.

1 Denklem Çözümüne Ait Parametrelerin Özeti ve OdeID: Bu alan, seçilen denklemin çözümünde kullanılan temel parametrelerin özetini ve bu çözümün veri tabanında kayıtlı olduğu OdeID'sini göstermektedir. OdeID, çözümün veri tabanındaki benzersiz tanımlayıcısıdır ve bu tanımlayıcı sayesinde çözüm sonuçlarına kolayca erişim sağlanabilir. Burada gösterilen parametreler, denklemin çözüm sürecinde hangi başlangıç koşulları ve iterasyon sayısı ile çalışıldığını özetlemektedir.

2 rhoStep: RhoStep, iterasyon sürecinde elde edilen rho değerlerinin adımını ifade eder. Bu alan, hangi iterasyonda rho değerlerinin hesaplandığını gösterir. İterasyon adımları, sistemin zamana bağlı olarak nasıl evrildiğini ve çözümün ne derece yakınsadığını izlemek için kritik öneme sahiptir. Her bir rhoStep, belirli bir iterasyonda sistemin durumunu ve bu durumun bir önceki adımla nasıl ilişkilendirildiğini gösterir.

3 rho: Bu alan, belirli bir iterasyon adımında hesaplanan rho vektörlerini ifade eder. Rho vektörleri, sistemin olasılıksal evrim yöntemi ile çözüm sürecinde, belirli bir iterasyon aşamasında elde edilen geçici sonuçları temsil eder. Rho'nun değeri, sistemin her adımda nasıl evrildiğini ve sistemin bir sonraki iterasyon için nasıl hazırlandığını göstermektedir. Bu vektörler, çözümün yakınsama sürecinde doğruluğunu kontrol etmek amacıyla kullanılır.

4 rhoVal: RhoVal, rho vektörlerinin belirli bir iterasyon adımındaki sayısal değerlerini ifade eder. Bu değerler, rho vektörlerinin elemanlarının, ilgili iterasyon sürecinde hangi sayısal sonuçları verdiğini gösterir. RhoVal, her iterasyonda sistemin çözüm sürecinin nasıl ilerlediğini detaylı olarak izlemek için kullanılan kritik bir veri setidir. Bu değerler, olasılıksal evrim yönteminin her adımında sistemin durumu hakkında bilgi sağlar ve çözümün ne kadar yakınsadığını değerlendirir.

SONUÇ

Bu bölüm, "Van der Pol denkleminin olasılıksal evrim yöntemi ile çözümü için veri tabanı oluşturma" başlıklı tez çalışmasının kapsamlı bir değerlendirmesini sunmaktadır. Çalışma süresince geliştirilen sistem, Van der Pol osilatörünün çeşitli parametrelerle sayısal çözümlerinin nasıl elde edildiği, bu çözümlerin nasıl işlendiği, saklandığı ve yönetildiği üzerine detaylı bilgiler sağlamaktadır. Projede elde edilen sonuçlar, tezin hedeflerine ne derece ulaşıldığı ve bu süreçte karşılaşılan süreçler ve elde edilen sonuçlar detaylı bir şekilde incelenmiştir. Bu değerlendirme, projenin bilimsel ve teknik katkılarını ortaya koymakta ve gelecekteki çalışmalar için önerilerde bulunmaktadır.

Projenin Genel Değerlendirmesi

Bu proje 3 aşamadan oluşmaktadır. Van der Pol denkleminin çözüm süreçlerinden elde edilen verilerin işlenmesi ve yönetimi, verilerin JSON formatına dönüştürülmesi ve MySQL veri tabanında saklanması aşamalarını kapsamaktadır.

Projede geliştirilen sistem, Van der Pol denkleminin çeşitli parametreler altında nasıl davrandığını anlamak için güçlü bir araç olarak işlev görmektedir. Kullanılan olasılıksal evrim yöntemleri ve bu yöntemlerin Node.js ve C++ gibi teknolojilerle entegrasyonu, etkin bir biçimde çözmemizi sağlamaktadır. Web tabanlı arayüz, kullanıcıların denklemin parametrelerini kolayca değiştirmesine ve sonuçları anında gözlemlemesine olanak tanımakta ve bu da sistemin interaktif ve kullanıcı dostu olmasını sağlamaktadır.

Veri tabanı sistemlerinin etkin kullanımı, büyük veri setlerinin yönetilmesi ve işlenmesi süreçlerinde kritik bir rol oynamaktadır. Verilerin JSON formatında işlenmesi ve SQL sorguları ile yönetilmesi, veri bütünlüğünü ve erişilebilirliği artırmakta, sistem üzerinde yapılan sorgulamaların hız ve verimliliğini önemli ölçüde iyileştirmektedir.

Kısıtlar ve Zorluklar

Çalışmada her ne kadar sistem başarılı bir şekilde implemente edilmiş olsa da bazı limitasyonlar ve zorluklar da yaşanmıştır. Özellikle büyük veri setlerinin

işlenmesi ve saklanması süreçlerinde performans optimizasyonları gerektiği tespit edilmiştir. Ayrıca, kullanılan sayısal çözümleme metodolojilerinin bazı spesifik durumlarda hassasiyetlerinin sınırlandırılması gerekebilir. Bu tür durumlar, gelecekteki çalışmalarda metodolojinin daha da rafine edilmesine olanak tanıyacak şekilde değerlendirilmiştir. Bu değerlendirme, projenin genel başarısını ve potansiyel iyileştirme alanlarını ortaya koymuştur. Sonuçlar, Van der Pol denkleminin çözümlerinin bilimsel ve teknolojik açıdan anlaşılmasını ilerletme potansiyeline sahip olduğunu göstermektedir. Bu çalışma, benzer sayısal analiz projeleri için bir referans noktası olarak hizmet edebilir ve daha geniş bilimsel topluluk içinde ileri araştırmalar için bir örnek oluşturabilir. Sistemin, çok daha geniş bir denklem takımları ailesi için kolayca uyarlanabilir olduğu da vurgulanmalıdır. Bu esneklik, metodolojinin farklı bilimsel problemlere uygulanabilirliğini artırarak, gelecekteki çalışmalarda daha çeşitli ve karmaşık dinamik sistemlerin analizinde kullanılmasına olanak tanıyacaktır.

Bulunan Sonuçların Analizi

Veri tabanında saklanan çözüm setleri üzerinde yapılan sorgulamalar, farklı başlangıç koşulları ve parametre değerlerinin sonuçlar üzerindeki etkilerini gözler önüne sermiştir.

Yapılan çalışmalarda bulunan sonuçların karşılaştırmaları tablo, çözüm ve grafik halinde eklerde verilmiştir. Ek-1 ve Ek-2'de, Python dilinde Runge-Kutta yöntemi ile elde edilen sayısal sonuçlar ile C++ dilinde olasılıksal evrim yöntemi ile elde edilen sonuçlar karşılaştırılmaktadır.

Python uygulaması, SciPy kütüphanesinin solve_ivp fonksiyonunu kullanarak Runge-Kutta yöntemini uygular. Başlangıç koşulları. $x(0) = 0.5$ ve $\dot{x}(0)=-0.04166$ olarak belirlenmiştir ve parametre (mu değeri) μ=1 olarak alınmıştır. Bu hesaplama sürecinde, Van der Pol denklemi ikinci mertebeden diferansiyel denklem olarak ele alınmıştır. Sonuçlar Ek-2'de gösterilmektedir. C++ uygulaması ise, çözümün sayısal kararlılığına odaklanan olasılıksal evrim yöntemi kullanır. Çıktı Ek-1'de gösterilmektedir. Yapılan karşılaştırma, her iki yöntemin de iterasyon değerlerinde benzer eğilimler gösterdiğini ortaya koymaktadır.

Olasılıksal Evrim Teorisi (Probabilistic Evolution Theory - PREVTH), doğrusal olmayan diferansiyel denklemler için sayısal çözümler üretmede güçlü bir yöntem sunar. Bu yöntemde, denklem çözümleri Taylor serisi açılımı kullanılarak hesaplanır ve yeni Taylor terimi ekleyerek geliştirilir.

Ek-3 ve Ek-4' de yer alan veriler, bu yöntemin uygulanması sonucu elde edilen sayısal sonuçları göstermektedir. Denklem çözümünde kullanılan parametreler başlangıç koşulları $x(0) = ½, y(0) = ½, \mu = 1$ ve iterasyon sayısı 50 olarak belirlenmiştir. t değeri 1/20 için her bir iterasyonda elde edilen değerler tabloya yansıtılmıştır. Projede ondalık basamak sayısı 50 karakter olarak ele alınmıştır, iterasyon ilerledikçe aynı değerlerin tekrar ediyor olması karakter sayısının sınırlı olarak ele alınmasından kaynaklanmaktadır. Eğer ondalık basamak sayısı daha fazla arttırılırsa yakınsamanın giderek daha kısa aralıklarla devam ettiği görülebilir.

Runge Kutta yöntemleri mühendislik uygulamalarında sıklıkla kullanılmaktadır. Aslında Runge Kutta yöntemleri tek bir yöntemi değil, bir yöntem ailesini ifade etmektedir. Bilimsel kitaplarda sıklıkla anılan Euler yöntemi de bu ailenin içinde yer almaktadır. Ek-5'de ikinci mertebe bir yöntem olan RK23, Ek-6'da dördüncü mertebe bir yöntem olan RK45 ve Ek-7'de sekizinci mertebe bir yöntem olan DOP853 bağlamında sonuçlar gösterilmiştir. Ek-5, Ek-6 ve Ek-7'deki gösterimlerde, üstte olasılıksal evrim kuramı çözümü, altta ise Runge Kutta yöntemleri çözümü aynı bölme içerisinde olacak şekilde karşılaştırmak amacıyla alt alta verilmiştir. Runge Kutta çözümlerini elde etmek için bir Python betiği kullanılmıştır. Python yorumlayıcısının Python 3.11.3 sürümü kullanılmıştır. Python scipy kütüphanesindeki solve_ivp komutuyla Runge Kutta çözümleri elde edilmiştir. Python komutu solve_ivp'nin varsayılan değerleri kullanılmıştır. Runge Kutta yöntemleri, Van der Pol denkleminin ikinci mertebe gösterilimine uygulanmıştır.

Elde edilen sonuçlar Runge Kutta yöntemi mertebesi arttıkça yaklaştırımın olasılıksal evrim kuramı çözümüne yaklaştığını göstermektedir. Bu olgu olasılıksal evrim kuramı çözümünü doğrular niteliktedir.

Öneriler ve Gelecek Çalışmalar

Bu tez çalışmasının sonuçlarına dayanarak, "Van der Pol Denkleminin Olasılıksal Evrim Yöntemi ile Çözümü için Veri Tabanı Oluşturma" konusunda yapılabilecek öneriler ve gelecek çalışmalar aşağıda sıralanmıştır. Bu öneriler, araştırmanın kapsamını genişletmek ve elde edilen bulguları daha da ileriye taşımak için tasarlanmıştır.

Daha Karmaşık Parametreler ve Koşullar:

Van der Pol denkleminin daha karmaşık parametreler ve başlangıç koşulları ile çözülmesi üzerine çalışılabilir. Mevcut çalışmada kullanılan parametreler ve başlangıç koşulları temel düzeydedir. Daha karmaşık ve gerçek dünya senaryolarını simüle etmek için farklı parametreler ve koşullar test edilebilir.

Alternatif Çözüm Yöntemleri:

Runge-Kutta yöntemi dışında, diğer sayısal çözüm yöntemlerinin uygulanması ve karşılaştırılması yapılabilir. Farklı sayısal yöntemlerin performansı ve doğruluğu karşılaştırılarak, Van der Pol denkleminin çözümünde en etkili yöntemin belirlenmesi sağlanabilir.

Veri Görselleştirme ve Analiz:

Çözüm sonuçlarının daha etkili bir şekilde görselleştirilmesi. Elde edilen sonuçların daha anlaşılır ve görsel olarak zengin bir şekilde sunulması için çeşitli grafik ve veri görselleştirme araçları kullanılabilir.

Veri Tabanı Performans Optimizasyonu:

Veri tabanı yapısının ve sorgu performansının optimize edilmesi veya NoSQL bir yapıya geçilmesi. Mevcut veri tabanı yapısı, büyük veri setleri ile çalışırken performans sorunları yaşatabilir. Bu nedenle, veri tabanı optimizasyon teknikleri kullanılarak performans iyileştirmeleri yapılabilir, NoSQL veritabanı kullanılarak yatay bir büyümenin yolu açılabilir.

Bu öneriler ve gelecek çalışmalar, projemiz devam ettiği sürece göz önünde bulundurulmalıdır. Uygulanan iyileştirmeler ve yeni araştırma yönleri, projenin bilimsel katkısını artırırken, teknolojik yenilikleri de entegre ederek sistemimizi daha

etkin ve erişilebilir kılabilmektedir. Bu stratejiler, Van der Pol denkleminin ve benzeri sistemlerin anlaşılmasına yönelik araştırmalarda örnek bir rol olma niteliğindedir. Ancak, gelecekteki çalışmaların, sistemin performansını daha da artırmak, kullanılabilirliği genişletmek ve daha geniş çaplı veri setleri üzerinde çalışabilmesi için iyileştirmeler yapması önerilmektedir.

Mevcut çalışmada μ, başlangıç koşulları ve iterasyon değerleri gibi temel parametreler kod üzerine eklenmiş ve sonra kodun tekrar derlenmesi sağlanmıştır. Ancak, bu parametrelerin harici bir "values.json" dosyasından okunarak kodun tekrar derlemeye gerek kalmadan sonuç üretebilmesi sağlanabilir. Bu hem zaman kazandıracak hem de farklı yapılara entegrasyon sağlanmasını kolaylaştıracaktır.

Bununla birlikte, farklı API entegrasyonları ile sistemin daha geniş bir veri kaynağına erişimi sağlanabilir ve bu sayede sonuçlar farklı sistemlerde de kullanılabilir hale getirilebilir. Bu tür entegrasyonlar, araştırma sürecine yenilikçi bir bakış açısı kazandıracak ve veri işleme kapasitesini artıracaktır.

Sistem İyileştirmeleri

Performans Optimizasyonu:

İşlenen veri miktarının artmasıyla birlikte, sistem performansını artıracak çözümler geliştirilmesi önerilir. Veri tabanı sorgu optimizasyonları, veri önbelleğe alınması ve daha etkili veri saklama yöntemleri bu iyileştirmeler arasında yer alabilir.

Kullanıcı Arayüzü Geliştirmeleri:

Kullanıcı arayüzünün daha işlevsel hale getirilmesi, kullanıcı deneyimini iyileştirebilir. Özellikle, veri görselleştirme araçlarının entegrasyonu, kullanıcıların verileri daha kolay analiz etmelerine yardımcı olacaktır.

Potansiyel Araştırma Yönleri

Yeni Matematiksel Modellerin Entegrasyonu: Van der Pol denkleminin yanı sıra, diğer nonlineer dinamik sistemlerin modellenmesi ve analizi için benzer sistemler geliştirilebilir. Bu, sistemlerin çeşitliliğini artırarak bilimsel anlayışı genişletebilir.

Makine Öğrenmesi Uygulamaları: Veri tabanında biriken büyük veri setlerinin, makine öğrenmesi teknikleri kullanılarak analiz edilmesi önerilir. Bu sayede, sistem dinamikleri üzerine daha derinlemesine çıkarımlarda bulunulabilir ve model tahminleri iyileştirilebilir.

Uluslararası İşbirlikleri: Bu tez kapsamında geliştirilen metodolojilerin, farklı coğrafyalardaki araştırmacılarla paylaşılması ve uluslararası işbirliklerinin kurulması, bilginin yayılmasını ve çeşitli bilimsel sorunlara çözüm bulunmasını teşvik edebilir.

Bu öneriler ve gelecek çalışma yönleri, bu tez çalışmasının oluşturduğu temel üzerine inşa edilmiştir ve bilim ve mühendislik alanlarında yeni araştırma fırsatları sunmaktadır. Bu çalışmalar, bilimsel topluluğun mevcut sınırlarını zorlayarak yeni keşifler yapılmasına olanak tanıyabilir.

KAYNAKÇA

Adomian, G. (1988). A review of the decomposition method in applied mathematics. *Journal of Mathematical Analysis and Applications, 135*(2), 501-544. https://doi.org/10.1016/0022-247X(88)90170-9.

Bailey, D. H., Barrio, R. ve Borwein, J. M. (2012). High-precision computation: Mathematical physics and dynamics. *Applied Mathematics and Computation*, *218*(20), 10106-10121. https://doi.org/10.1016/j.amc.2012.03.087.

Barrio, R. (2005). Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation*, *163*(2), 525-545. https://doi.org/10.1016/j.amc.2004.02.015

Barrio, R., Dena, A. ve Tucker, W. (2015). A database of rigorous and high-precision periodic orbits of the Lorenz model. *Computer Physics Communications*, 194, 76-83. https://doi.org/10.1016/j.cpc.2015.04.007.

Bostan, A., Chyzak, F., Ollivier, F., Salvy, B., Schost, É. ve Sedoglavic, A. (2007). Fast computation of power series solutions of systems of differential equations. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1012-1021.

Bruno, A. D. (1997). Power geometry. *Journal of Dynamical and Control Systems*, *3*(4), 471-491. https://doi.org/10.1007/BF02463279

Bychkov, A. ve Pogudin, G. (2021). Optimal monomial quadratization for ODE systems. In P. Flocchini ve L. Moura (Eds.), *Combinatorial Algorithms* 122-136. Springer, Cham.

Carothers, D. C., Lucas, S. K., Parker, G. E., Rudmin, J. D., Sochacki, J. S., Thelwell, R. J., Tongen, A. ve Warne, P. G. (2012). Connections between power series methods and automatic differentiation. In S. Forth, P. Hovland, E. Phipps, J. Utke, ve A. Walther (Eds.), *Recent Advances in Algorithmic Differentiation* 175-185. Springer.

Çakır, K., Mutlu, R. ve Karakulak, E. (2021). Ters-paralel bağlı Schottky Diyot dizisi tabanlı Van der Pol osilatörü devresinin modellenmesi ve LTspice ve Simulink kullanarak analizi. *Elektrik, Elektronik, Bilgisayar, Biyomedikal, Kontrol Mühendisliği Bilimsel Hakemli Dergisi, 11*(21), 81-91.

Demiralp, M. (2018). Probabilistic evolution theory: Current state and future perspectives. *Mathematics in Engineering, Science ve Aerospace (MESA), 9*(1), 25-42. http://nonlinearstudies.com/index.php/mesa/article/view/1707

Eberhard, P. ve Bischof, C. (1999). Automatic differentiation of numerical integration algorithms. *Mathematics of Computation, 68*(226), 717-731. https://doi.org/10.1090/S0025-5718-99-01027-3

Elmasri, R. ve Navathe, S. B. (2017). *Fundamentals of database systems.* Pearson.

Gözükırmızı, C. (2022). Beam search for space extension in explicit ordinary differential equation conicalization. *Computational Technologies, 27*(6), 100-114. https://doi.org/10.25743/ICT.2022.27.6.009

Gözükırmızı, C. ve Demiralp, M. (2014a). Probabilistic evolution approach for the solution of explicit autonomous ordinary differential equations. Part 1: Arbitrariness and equipartition teorem in Kronecker power series. *Journal of Mathematical Chemistry, 52*(3), 866-880. https://doi.org/10.1007/s10910-013-0298-5

Gözükırmızı, C. ve Demiralp, M. (2014b). Probabilistic evolution approach for the solution of explicit autonomous ordinary differential equations. Part 2: Kernel separability, space extension, and series solution via telescopic matrices. *Journal of Mathematical Chemistry, 52*(3), 881-898. https://doi.org/10.1007/s10910-013-0299-4

Gözükırmızı, C. ve Demiralp, M. (2019). Solving ODEs by obtaining purely second degree multinomials via branch and bound with admissible heuristic. *Mathematics, 7*(4), 367. https://doi.org/10.3390/math7040367

Gözükırmızı, C. (2024). Pure quadratization and solution of ordinary differential equations by probabilistic evolution theory with concurrent computation of

coefficients using exact arithmetic. *J Math Chem* 62, 654–680. https://doi.org/10.1007/s10910-023-01563-8

Gültekin, B., Biroğul, S. ve Yücedağ, İ. (2015). İşe alım süreci aday ön tesbitinde bulanık mantık tabanlı SQL sorgulama yönteminin incelenmesi. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 3*(1), 198–209.

Hunutlu, F., Baykara, N. A. ve Demiralp, M. (2013). Truncation approximants to probabilistic evolution of ordinary differential equations under initial conditions via bidiagonal evolution matrices: simple case. *International Journal of Computer Mathematics. 90*(11), 2326-2337. https://doi.org/10.1080/00207160.2013.774385

Ilyashenko, Y. S. (1969). An example of equations dw/dz = Pn(z,w)/Qn(z,w) having a countable number of limit cycles and arbitrarily large Petrovskii-Landis genus. *Mathematics of the USSR-Sbornik, 9*(3), 365-378. https://doi.org/10.1070/sm1969v009n03abeh001288

Jorba, A. ve Zou, M. (2005). A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Experimental Mathematics, 14*(1), 99-117. https://doi.org/10.1080/10586458.2005.10128904

Kalaycıoğlu, M. (2017). Ben buldum! *Bilim ve Teknik Dergisi,* 64-68.

Kalay, B. ve Demiralp, M. (2018). Somehow emancipating Probabilistic Evolution Theory (PREVTH) from singularities via getting single monomial PREVTH. *J Math Chem* 56, 2024–2043. https://doi.org/10.1007/s10910-017-0815-z

Kovalnogov, V. N., Fedorov, R. V., Karpukhina, T. V., Simos, T. E. ve Tsitouras, C. (2022). Runge–Kutta embedded methods of orders 8(7) for use in quadruple precision computations. *Mathematics*, *10* (18), 3247. https://doi.org/10.3390/math10183247

Kovalnogov, V. N., Fedorov, R. V., Karpukhina, T. V., Simos, T. E. ve Tsitouras, C. (2023). Runge–Kutta pairs of orders 9(8) for use in quadruple precision computations. *Numer Algor* 95, 1905–1919 https://doi.org/10.1007/s11075-023-01632-8

Öztürk, Z. (2017). George Boole'un cebirsel mantığı ve mantık tarihindeki yeri. V. Kamer ve Ş. Ural (Ed.), *VII. Mantık çalıştayı kitabı içinde* (s. 576). Mantık Derneği Yayınları.

Sevimli, E. (2016). Diferansiyel denklemlerin öğreniminde yaşanan zorluklar ve alternatif öğretim yaklaşımları. *SAÜ Eğitim Bilimleri Enstitüsü Sakarya University Journal of Education,* 154-171.

Sochacki, J. S. (2010). Polynomial ODEs - examples, solutions, properties. *Neural, Parallel ve Scientific Computations, 18*(3-4), 441-450.

Stanley, R. P. (1980). Differentiably finite power series. *European Journal of Combinatorics, 1*(2), 175-188. https://doi.org/10.1016/S0195-6698(80)80051-5

Strogatz, S. H. (1994). *Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering.* Perseus Books Publishing.

Upton, S. D. (2004). *Students'solution strategies to differerential equations problems in mathematical and nonmathematical contexts* [Yayımlanmamış doktora tezi]. The Arizona State University.

Üsküplü Altınbaşak, S., Demiralp, M. (2010). Solutions to linear matrix ordinary differential equations via minimal, regular, and excessive space extension based universalization. J *Math Chem* 48, 266–286 https://doi.org/10.1007/s10910-010-9667-5

Van der Pol, B. (1927). On relaxation oscillations. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, 7*(2), 978–992.

İnternet Kaynakları

Bychkov, A, (2021). *QBee*, 16 Mart 2024 tarihinde https://github.com/AndreyBychkov/QBee adresinden edinilmiştir.

Coderspace.io, (2024). *API*, 22 Nisan 2024 tarihinde https://coderspace.io/sozluk/api/ adresinden edinilmiştir.

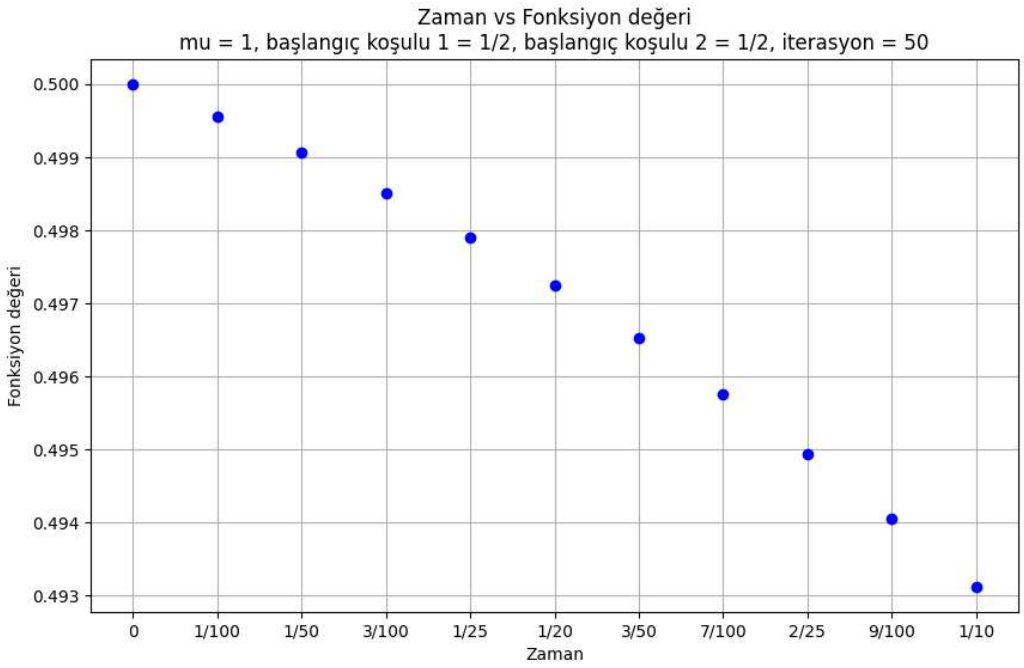
Expressjs.com, (2024). *Glossary*, API. 22 Nisan 2024 tarihinde https://expressjs.com/en/resources/glossary.html/ adresinden edinilmiştir.

Gözükırmızı, C. (2023). *Space-extension-of-explicit-ODEs,* 30 Mart 2024 tarihinde https://github.com/cosargozukirmizi/space-extension-of-explicit-ODEs adresinden edinilmiştir.

Nodejs.com (2024). *Node.js'ye giriş*. 21 Nisan 2024 tarihinde https://nodejs.org/en/learn/getting-started/introduction-to-nodejs/ adresinden edinilmiştir.

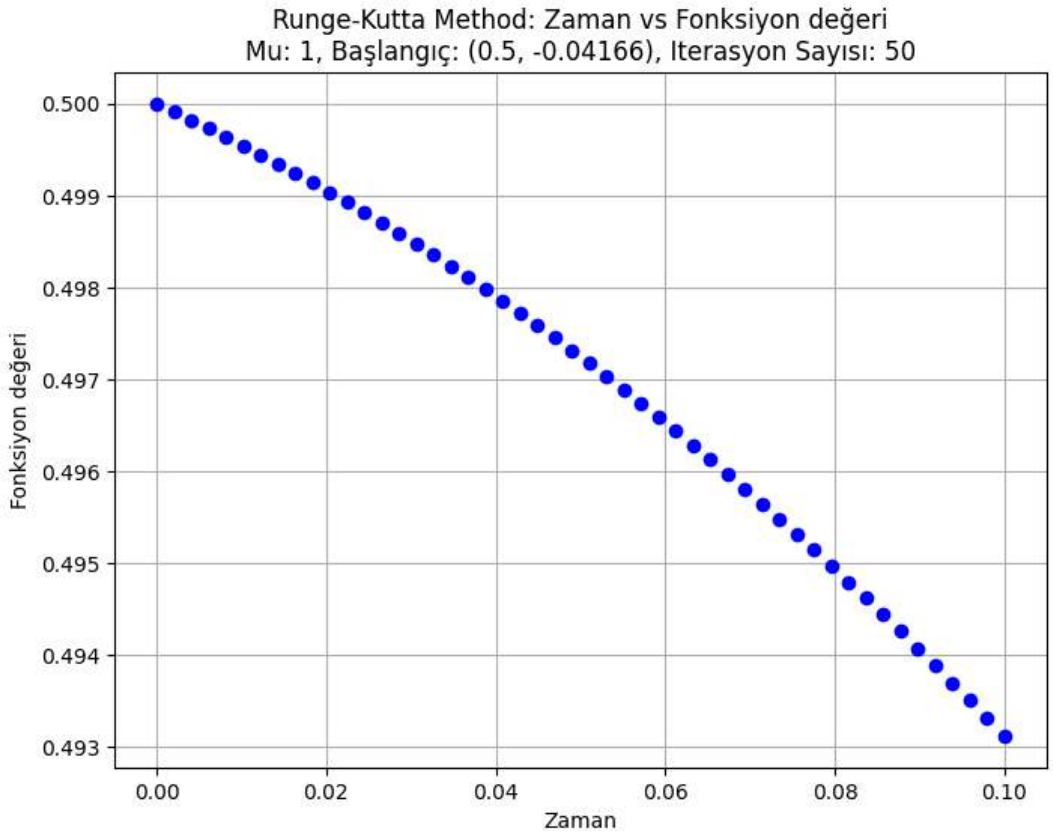
EKLER

Ek-1: Çalışmada C++ Kodu ile Elde Edilen Sonuç



Kaynak: Yazar tarafından oluşturulmuştur.

Ek-2: Runge-Kutta Yöntemiyle Çözüm



Kaynak: Yazar tarafından oluşturulmuştur.

Ek-3: μ = 1 Başlangıç Değeri (1/2,1/2) 50. İterasyona Ait Fonksiyon Değerleri

Zaman	Fonksiyon değeri
0	0.5000000000000000000000000000000000000000000000000000
1/100	0.49955671116356012014755088236925034464949140199171
1/50	0.49905993995422607186759259328317784520997017128747
3/100	0.49850933075648114613762661686551798936835635008772
1/25	0.49790452983370856641405105562802694724981978020222
1/20	0.49724518528440483184769814442697164933417036238901
3/50	0.49653094699655310636598094735478889318794718560178
7/100	0.49576146660021021823897603537809093971812624209293
2/25	0.49493639741836321522525136351475895190866051441382
9/100	0.49405539441611379826059034608763329650279435045666
1/10	0.49311811414825133346270672039989299464596398501126

Kaynak: Yazar tarafından oluşturulmuştur.

Ek-4: μ = 1, Başlangıç Koşulu (1/2, 1/2), İterasyon = 50, T = 1/20 İçin Aralıklı İterasyonlar

iteration	**Fonksiyon değeri**
0	0.5000000000000000000000000000000000000000000000000000000000
5	0.4972451853267999342930169753086419753086419753086419753086
10	0.4972451852844048316160293425357176810755718958243 9
15	0.4972451852844048318476981452064421604637419066311 0
20	0.4972451852844048318476981444269716483227398684093 8
25	0.4972451852844048318476981444269716493341703546595 9
30	0.4972451852844048318476981444269716493341703623890 1
35	0.4972451852844048318476981444269716493341703623890 1
40	0.4972451852844048318476981444269716493341703623890 1
45	0.4972451852844048318476981444269716493341703623890 1
49	0.4972451852844048318476981444269716493341703623890 1

Kaynak: Yazar tarafından oluşturulmuştur.

Ek-5: RK23 Yöntemi ile Elde Edilen Sonuçların Olasılıksal Evrim Kuramı ile Elde Edilen Sonuçlar ile Karşılaştırılması

Zaman	Fonksiyon değeri
0	0.5000000000000000000000000000000000000000000000000000000000 0.5
1/100	0.4995567111635601201475508823692503446494914019917 1 0.4995567110458984
1/50	0.4990599399542260718675925932831778452099701712874 7 0.4990599168363594
3/100	0.4985093307564811461376266168655179893683563500877 2 0.49850924473346
1/25	0.4979045298337085664140510556280269472498197802022 2 0.4979043557360142
1/20	0.4972451852844048318476981444269716493341703623890 1 0.49724491089585826
3/50	0.4965309469965531063659809473547888931879471856017 8 0.4965305712648283
7/100	0.4957614666002102182389760353780909397181262420929 3 0.4957609978947606
2/25	0.4949363974183632152252513635147589519086605144138 2 0.4949358518374912
9/100	0.4940553944161137982605903460876332965027943504566 6 0.4940547941448564
1/10	0.4931181141482513334627067203998929946459639850112 6 0.4931174858686924

Kaynak: Yazar tarafından oluşturulmuştur.

Ek-6: RK45 Yöntemi ile Elde Edilen Sonuçların Olasılıksal Evrim Kuramı ile Elde Edilen Sonuçlar ile Karşılaştırılması

Zaman	Fonksiyon değeri
0	0.5000000000000000000000000000000000000000000000000000000000 0.5
1/100	0.4995567111635601201475508823692503446494914019917 1 0.4995567111897713
1/50	0.4990599399542260718675925932831778452099701712874 7 0.49905994005380505
3/100	0.4985093307564811461376266168655179893683563500877 2 0.49850933093236754
1/25	0.4979045298337085664140510556280269472498197802022 2 0.49790453002997387
1/20	0.4972451852844048318476981444269716493341703623890 1 0.49724518541538804
3/50	0.4965309469965531063659809473547888931879471856017 8 0.4965309470216227
7/100	0.4957614666002102182389760353780909397181262420929 3 0.4957614666040884
2/25	0.4949363974183632152252513635147589519086605144138 2 0.49493639742903456
9/100	0.4940553944161137982605903460876332965027943504566 6 0.494055394427032
1/10	0.4931181141482513334627067203998929946459639850112 6 0.4931181141507546

Kaynak: Yazar tarafından oluşturulmuştur.

Ek-7: DOP853 Yöntemi ile Elde Edilen Sonuçların Olasılıksal Evrim Kuramı ile Elde Edilen Sonuçlar ile Karşılaştırılması

Zaman	Fonksiyon değeri
0	0.5000000000000000000000000000000000000000000000000000000000 0.5
1/100	0.4995567111635601201475508823692503446494914019917 1 0.4995567111635641
1/50	0.4990599399542260718675925932831778452099701712874 7 0.4990599399542286
3/100	0.4985093307564811461376266168655179893683563500877 2 0.49850933075649145
1/25	0.4979045298337085664140510556280269472498197802022 2 0.4979045298337412
1/20	0.4972451852844048318476981444269716493341703623890 1 0.4972451852844517
3/50	0.4965309469965531063659809473547888931879471856017 8 0.4965309469965775
7/100	0.4957614666002102182389760353780909397181262420929 3 0.4957614666001755
2/25	0.4949363974183632152252513635147589519086605144138 2 0.4949363974182794
9/100	0.4940553944161137982605903460876332965027943504566 6 0.4940553944160535
1/10	0.4931181141482513334627067203998929946459639850112 6 0.49311811414825124

Kaynak: Yazar tarafından oluşturulmuştur.