T.C

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DOĞRUSAL OLMAYAN SİSTEMLERDE LYAPUNOV ÜSTELLERİNİ HESAPLAYAN YAZILIMIN GERÇEKLEŞTİRİLMESİ

Fatih ÖZKAYNAK

Tez Yöneticisi

Yrd.Doç.Dr. A. Bedri ÖZER

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2007

T.C.

FIRAT ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DOĞRUSAL OLMAYAN SİSTEMLERDE LYAPUNOV ÜSTELLERİNİ HESAPLAYAN YAZILIMIN GERÇEKLEŞTİRİLMESİ

Fatih ÖZKAYNAK

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı

Bu tez, ... tarihinde aşağıda belirtilen jüri tarafından oybirliği /oyçokluğu ile başarılı / başarısız olarak değerlendirilmiştir.

Danışman:

Üye:

Üye:

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

Bu tez çalışması boyunca ilgi ve yardımlarını esirgemeyen danışman hocam, Sayın Yrd. Doç. Dr. A. Bedri ÖZER' e teşekkür ve şükranlarımı sunarım.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TALOLAR LİSTESİ

SİMGELER LİSTESİ

λ : Lyapunov Üsteli

t : Zaman göstergesi

Δt : Sürekli zamanlarda noktalar arasındaki mesafe

n : Ayrık zaman göstergesi

Δn : Ayrık zamanlarda noktalar arasındaki mesafe

d_n : n. adımdaki fark

D_L : Lyapunov boyutu

ÖZET

Yüksek Lisans Tezi

DOĞRUSAL OLMAYAN SİSTEMLERDE LYAPUNOV ÜSTELLERİNİ HESAPLAYAN YAZILIMIN GERÇEKLEŞTİRİLMESİ

Fatih ÖZKAYNAK

Fırat Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

2007, Sayfa: 74

Bir çok doğrusal olmayan sistemin, zaman içindeki değişiminin düzensiz olması ve tahmin edilememesi kaos olarak adlandırılmıştır. Doğrusal olmayan pek çok sistemde kaos meydana gelmektedir. Kaosun temel karakteristiği, sistemin geçmiş davranışını tekrar etmemesidir. Düzenden yoksun olmasına rağmen, kaotik dinamik sistemler gerekirci (deterministlik) denklemlerden oluşmuştur. Kaosun en belirleyici özelliği, başlangıç şartlarına olan bağımlılığıdır.

Kaosun gösterilmesi için değişik yöntemler vardır. Lyapunov üstelleri dinamik sistemin faz uzayındaki iki komşu başlangıç noktalarının ortalama üstel ıraksama veya yakınsamasını ölçmektedir. Pozitif bir Lyapunov üsteli iki komşu yörüngenin ortalama üstel ıraksadığını ölçerken negatif bir Lyapunov üsteli iki komşu yörüngenin ortalama üstel olarak yakınsadığını ölçmektedir. Pozitif Lyapunov üsteli sistemin kaotik olduğunu göstermektedir.

Bu tezde, doğrusal olmayan sistemlerin Lyapunov üstellerinin hesaplanması için bir yazılım tasarlanmıştır. Lyapunov üstelleri doğrusal olmayan sistemlerde kaosun gösterilmesi için kullanılmıştır.

Anahtar Kelimeler : Kaos, Lyapunov Üstelleri, Lyapunov Üstellerinin Hesaplanması

ABSTRACT

Master Thesis

IMPLEMENTATION OF SOFTWARE THAT CALCULATES THE LYAPUNOV EXPONENTS IN NONLINEAR SYSYEMS

Fatih ÖZKAYNAK

Fırat University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

2007, Page: 74

The irregular and unpredictable time evolution of many nonlinear systems has been called chaos. Chaos occurs in many nonlinear systems. Main characteristic of chaos is that system does not repeat its past behavior. In spite of their irregularity, chaotic dynamical systems follow deterministic equations. The unique characteristic of chaotic systems is dependence on the initial conditions sensitively.

There are various methods for detecting chaos. Lyapunov exponents measure the average exponential divergence or convergence of nearby initial points in the phase space of dynamical system. A positive Lyapunov exponent is a measure of average exponential divergence of two nearby trajectories whereas a negative Lyapunov exponent is a measure of the average exponential convergence of two nearby trajectories. Positive Lyapunov exponent is an indication that the system is chaotic.

In this thesis, a software for estimating Lyapunov exponents of nonlinear systems has been designed. The Lyapunov exponents are used to detect chaos in nonlinear systems.

Keywords : Chaos, Lyapunov Ecponents, Calculation of Lyapunov Exponents,

1 GİRİŞ

Bilim basit şekilde doğal olayların anlaşılmasına yardım eden, insan yaşamını her alanda etkileyen çok önemli bir kavramdır. Ulaşım araçları, elektronik cihazlar, sağlık sektöründeki gelişmeler ve akıllı yazılımlar bilimin yaşamımızı etkileyen uygulama sonuçlarından bazılarıdır.

Bilimsel yollardan elde edilen bilgiler insanoğluna doğal çevresini denetim altına alma olanağı sağlamış ve doğa olanaklarını kendi yaşamını kolaylaştırma daha rahat, daha güvenilir ve daha uzun yaşama yolunda kullanma yeteneğini vermiştir [1].

Klasik dönemdeki bilimsel yöntemler sistemlerdeki düzensiz, kararsız ve birbirleriyle bağlantısız gibi gözüken davranışları göz ardı etmiş, gürültü veya dış etki olarak adlandırmış ve tasarım yoluyla bunlardan kaçınmaya çalışmıştır [2]. Ancak 1600'lı yılların ortasında Newton'un çalışmaları, fiziksel sistemlerin hareketlerinin diferansiyel denklemlerle ifade edilebileceğini göstermiş ve Dünyanın Güneş etrafındaki hareketini hesaplayan problemi çözmüştür. Daha sonra gelen bilim adamları Newton'un çalışmasını Güneş, Dünya ve Ay'ın hareketini hesaplayacak şekilde genişletmek istemişler ancak uzun yıllar başarısız olmuşlardır [3-4].

Bu problem yeni bir bilim dalı olan doğrusal olmayan sistemler teorisini ortaya çıkarmıştır. Doğrusal olmayan sistemlerin, zaman içerisindeki düzensiz ve kestirilemez davranış göstermesi kaos olarak adlandırılmıştır [2]. Kaos teorisi bilimim birçok alanına etki yapmış, uygulamalı bilimdeki birçok araştırmacı sistemlerindeki düzensiz davranışları ve anormallikleri açıklamak için kaostan faydalanmıştır.

Kaos teorisi, evrenin geometrisinden düzensizliğin kaynağına, güneş sisteminin işleyişinden yapısına, küçük etkilerin nasıl büyük değişikliklere neden olabileceğine ve düzensiz yapıların içinde bile nasıl bir düzenli yapının olduğunu göstermektedir [5,6].

Kaotik davranış gösteren sistemlere matematikten biyolojiye, ekonomiden elektronik devrelere, mühendislikten sosyal bilimlere, insan vücudunun davranışından vahşi nüfusun dağılımına kadar çok geniş bir alanda rastlanmaktadır. Kaotik sistemlere mühendislik uygulamaları açısından bakıldığında ise temel problem, karmaşıklığın ve kestirilemezliğin denetlenebilmesi şekline görülmektedir [7-10]. Geliştirilen sistemlerin özelliklerindeki artış sistemin karmaşıklığını artırmakta buda sistemin modellenmesini güçleştirmektedir. Sistem karmaşıklığını karşılayamayan bir modele dayanarak yapılan tasarımın fiziksel uygulamaya geçildiğinde yaratabileceği bilinmeyen dinamik de belli koşullar oluştuğunda aktif olmakta ve sistemin cevabında beklenmeyen bir davranış gözlenebilmektedir. Kaotik dinamik alanında

yapılan birçok çalışmada, bahsedilen bu davranışın denetim altına alınabilmesini amaçlamaktadır [11].

Kaosun disiplinler arası bir konu olması, sistemde kaosun varlığını gösteren yöntemleri ön plana çıkarmıştır. Kaos analizi yapabilmek için ilk adım kullanılacak yöntemin karakteristikleri hakkındaki teorik bilgi oluşturulması ve bu teorik bilginin modellenerek, bilgisayar sistemleri aracılığıyla benzetimi (simülasyonu) ve analizi olmuştur.

Bilişim sektöründeki birçok gelişme bilim insanlarının araştırmalarındaki benzetim ve analiz işlemini kolaylıkla yapma imkânı sağlamıştır ancak dikkat edilmesi gereken temel nokta yapılan benzetim ve analizlerin kullanıcı ihtiyaçlarını doğru şekilde karşılayabilmesi ve doğrululuğunun kabul edilebilir seviyede olmasıdır. Bu sebeplerden ötürü sistemlerin benzetim ve analizinde kullanılacak bilgisayar sistemlerinin bir mühendislik tasarımı olarak geliştirilmesini gerektirmektedir.

1.1 Tezin Amacı

Literatür incelendiğinde kaosu gösteren birçok program olmasına rağmen bu programların genellikle özel sistemler ve uygulamalar için gerçekleştirilmiş olduğu veya test edildiğinde doğru çalışmadığı görülmüştür.

Bu tezin amacı doğrusal olmayan sistemlerde (sürekli veya ayrık zamanlı doğrusal olmayan sistemler) kaos analizi yapabilmek için bir yazılım geliştirmektir. Kaosun analizini yapmak için birçok yöntem kullanılabilmektedir. Bu yöntemlerin çoğu niteliksel yöntemlerdir. Yani kaos analizini niceliksel olarak göstermezler. Lyapunuv üstelleri (veya üstleri) olarak bilinen yöntem ise kaos analizini diğer yöntemlerden farklı olarak niceliksel olarak belirlemeye yarayan bir analiz yöntemidir [2]. Bu nedenle sistemin kaos analizi yapmak için Lyapunov üstelleri yönteminden faydalanılmıştır. Ayrıca geliştirilen sistemin grafik arayüzünün olması analiz işlemini farklı açılardan görebilme olanağını sağlamıştır.

1.2 Tezin Yapısı

Tezin yapısı ise aşağıdaki gibi oluşturulmuştur.

İkinci Bölüm: Bu bölümde kaos teorisinin temelleri açıklanmış ve kaosun oluşumu ayrık ve sürekli zamanlı sistemler üzerinde gösterilmiştir. Bölümüm son kısmında kaos analizinde kullanılan bazı yöntemler verilmiştir.

Üçüncü Bölüm: Bu bölümde geliştirilen yazılımda kaos analizi yapmak için kullanılan Lyapunov üstellerinin karakteristik özellikleri detaylı olarak açıklanmış sürekli ve ayrık zamanlı sistemler için kullanılan yöntemin nasıl hesaplama yaptığı gösterilmiştir.

Dördüncü Bölüm: Bu bölümde ilk olarak sistem, modelleme ve yazılım kavramları açıklanmıştır. Yazılım modelleme dili olan UML (Unified Modeling Language) kısaca özetlenmiş ve geliştiren yazılımın modeli verilmiştir.

Beşinci Bölüm: Bu bölümde gerçekleştirilen yazılımın genel hatlarıyla kurulumu ve kullanımı açıklanmıştır.

Altıncı Bölüm: Bu bölümde gerçekleştirilen yazılım çeşitli sistemler (ayrık ve sürekli zamanlı sistemler) üzerinde çalıştırılarak sistemlerin kaos analizi yapılmış ve yazılımın doğruluğu test edilmiştir.

Yedinci Bölüm: Bu bölümde ilk olarak bir mühendislik ürünü olan yazılımın değerlendirilmesinde kullanılacak ölçütler verilmiş ardından gerçekleştirilen yazılım bu ölçütler göz önünde bulundurularak değerlendirilmiştir. Bölümün son kısmında benzer programlar verilerek programın avantajları gösterilmiştir.

Sonuç Bölümü: Bütün olarak elde edilen sonuçların değerlendirilmesi bu bölümde yapılmıştır. Ayrıca daha sonraki çalışmalara ilişkin önerilerde bu bölümde tartışılmıştır.

2 KAOTİK DİNAMİK

2.1 Kaos Teorisi

Kaos kelimesi genellikle karmaşa veya kestirilemezlik kavramlarıyla beraber anılmaktadır. Birçok doğrusal olmayan sistemin zamandaki değişiminin düzensiz ve kestirilemez olması ve birçok araştırmacının sistemlerindeki kestirilemezliği inceleme düşüncesi son zamanlarda kaos teorisini ön plana çıkarmıştır.

Kaotik sistemlerin matematiksel modelleri doğrusal olmayan bir yapıya sahiptir. Hem sürekli zamanlı diferansiyel denklemlerde hem de ayrık zamanlı fark denklemleri ile ifade edilebilmektedir. Sistemlerin matematiksel modellerinin olması bir gerekircilik (determinizm) ortaya koyarken uzun dönemdeki davranışının kestirilemez olması; kaotik sistemleri doğrusal olmayan sistemler altında incelenen birçok modelden farklı kılmaktadır [7,8].

Kaotik sistemlerin sahip olduğu iki temel karakteristik:

- Sistem gerekirci denklemlerle ifade edilmesine rağmen sistemin geçmişteki davranışını tekrar etmeyerek düzensiz bir yapıya sahip olması
- Sistemin başlangıç koşullarına duyarlı olması

şeklindedir.

2.1.1 Kaotik Sistemlerin Gerekirciliği

Dinamik sistemler çeşitli dinamik kurallar ile tanımlanabilir ve bu dinamik sistemlerin çözümleri tektir. Bu nedenle bu tip sistemler gerekirci olarak adlandırılmaktadır. Dinamik sistemin durumu tüm zamanlar için tektir. Sistemin herhangi bir zamandaki durumuna karar verilebilir [8].

Gerekirci doğrusal olmayan dinamik sistemlerin çözümleri ise rasgele (istatistiğe bağlı durumlar) olabilir. Bu tip davranış gerekirci kaos olarak adlandırılmaktadır. Burada önemli olan temel nokta çözümlerin gerçekten rasgele olup olmadığıdır. Rasgelelik temel olarak gürültü aracılığıyla üretildiğinden buradaki sonuçlar rasgele değildir. Sonuçların rasgele görünmesinin nedeni başlangıç koşullarına bağımlı olan gerekirci denklemlerde kullanılan değişkenlerin değerindeki çok küçük değişimlerdir [3-4].

Kaotik sistemler gerekirci olmasına karşın öngörülebilir değildir. Kaotik sistemlerin başlangıç koşullarına aşırı duyarlılığı, başlangıç değerinin hassas ölçülememesi veya başlangıç

değerinin irrasyonel sayılarla (√7 gibi) ifade ediliyor olabilmesi öngörülebilirliğe engel olur [12]. Kaotik sistemler doğrusal olmayan yapılarından ötürü öngörülebilir olmaması ama aynı zamanda gerekirci bir yapıya da sahip olması klasik mekanikte beraber düşünülen bu kavramların ayrı ayrı ele alınması gerektiğini göstermiştir [12].

2.1.2 Başlangıç Koşullarına Duyarlılık

Kaotik dinamiklerin önemli bir diğer özelliği ise sistem birbirinden çok küçük farklı olan iki başlangıç koşulunda başlatıldığında gözlemlenebilir. Bu küçük farklar kaotik olmayan sistemlerde ölçme hatası olarak ifade edilebilir ve hata zamanla doğrusal olarak artmaktadır. Fakat kaotik sistemlerde hata üstel olarak artmakta ve sistemin gelecekteki durumları kestirilemez olmaktadır. Bu olgu başlangıç koşullarına duyarlılık olarak bilinmektedir [3,4].

Bu alandaki ilk çalışma meteorolojik olayları modellemek için hava tahminlerinden elde edilen sayısal veriler kullanılarak bir model geliştirme fikri sonucu ortaya çıkmıştır. Ancak hava durumum sahip olduğu kaotik durumdan ötürü sayısal verilerin hassasiyeti her bir denemede artmasına rağmen geçerli bir model elde edilememiş verilerdeki küçük bir ihmal farklı sonuçlara neden olmuştur [11].

Kaotik sistemlerde kararlılık ve karasızlık çok küçük değişimlerle birbirine dönüşür. Kaos hali binlerce kararlı ve kararsız durumların birlikte var olduğu bir durumdur [12]. Kararlı bir hal en küçük bir değişiklikle kararsızlığa, kararsızlık ise kararlılığa dönüşür. Bu nedenle “kararlı kararsızlık” durumu diye tanımlanabilir. Kararlı kararsız bölgeler aşırı derecede küçük olduğu için kararlı bölgedeki en küçük bir değişiklik sistemi kaosa götürür. Diğer bir deyişle kaotik sistemler başlangıç koşullarına aşırı duyarlıdır

2.1.3 Kaos Analizi İçin Gerek ve Yeter Koşullar

Bir sistemde kaos analizi yapılabilmesi için bazı gerekli şartlar vardır. Bunlardan ilki sistemde doğrusal olmayan eleman veya elemanlar olmasıdır. Doğrusal sistemde kaosun gözlenmeyeceği bilinmektedir. Şartlardan ikincisi ise, sürekli zamanlı sistemler için ve ayrık zamanlı sistemler için farklılık göstermektedir. Sistem sürekli zamanlı bir sistem ise, kaosun aranabilmesi için ikinci şart en az 3. dereceden bir sistem olmasıdır. Fakat ayrık zamanlı sistemde böyle bir şart yoktur. Ayrık zamanlı sistem birinci dereceden dahi olsa kaos analizi yapılabilmektedir. Lojistik harita (logistic map) buna bir örnektir. Kaos analizi yapabilmek için bahsedilen iki şart gerekli şartlardır, yeterli şartlar değillerdir. Yani bu şartlara uyan sistemlerde

kaos analizi yapılabilir fakat kesinlikle kaotik davranış gösterir diye bir yargıya varılamaz [8]. Bir sistemde kaos gözlenebilmesi için yeter şart ise sistem yörüngesinin başlangıç koşullarına duyarlı olmasıdır.

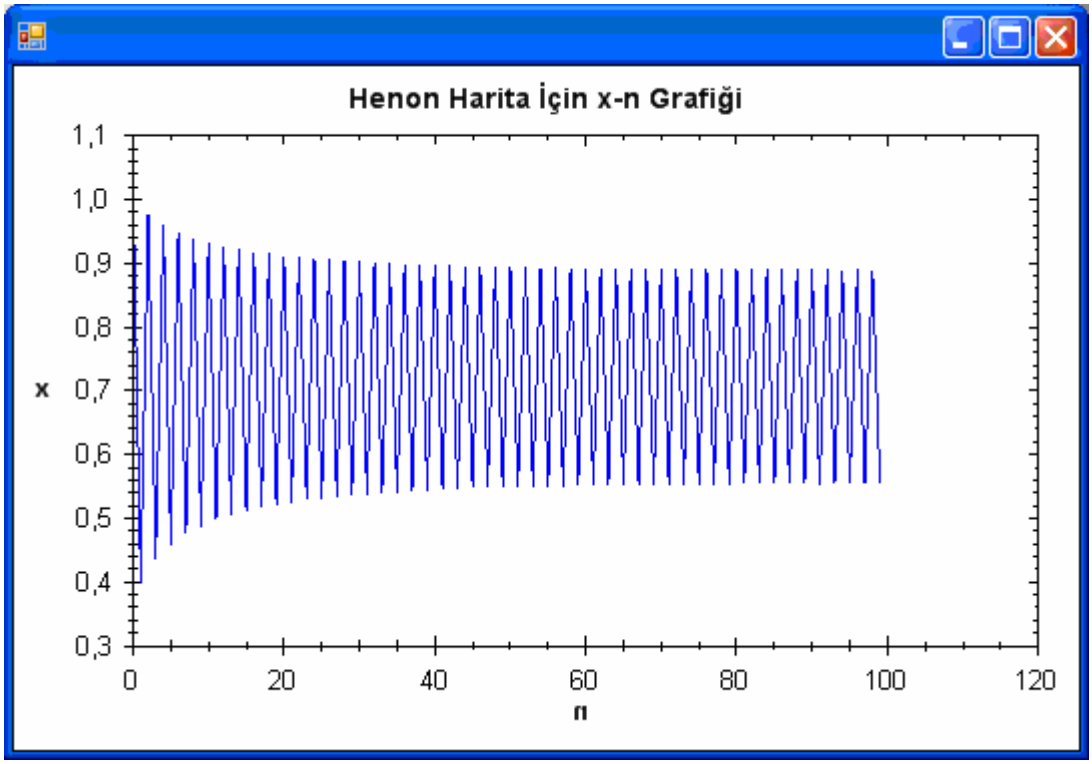
2.2 Kaotik Sistemlerin Davranışları

Bu bölümde biri ayrık zamanlı diğeri sürekli zamanlı iki kaotik sistemin çeşitli parametre değerleri için davranışları verilmiş ve ardından bu sistemlerin başlangıç koşullarına olan hassasiyeti gösterilmiştir.

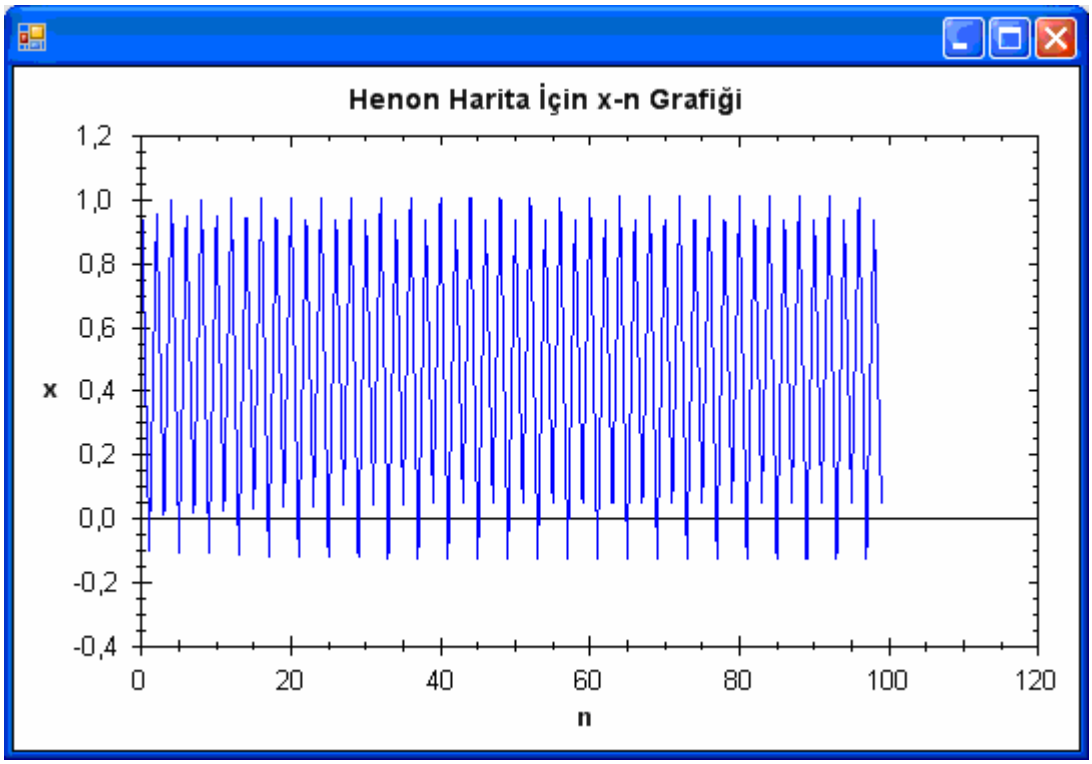
İlk örneğimiz iki boyutlu, ayrık zamanlı, doğrusal olmayan, kaotik bir sistem olan Henon haritadır. Henon haritaya ait dinamikler denklem (2.1)'de verilmiştir. Sistem parametreleri $-2 \leq x_n \leq 2$ ve $-2 \leq y_n \leq 2$ şeklinde değişmektedir. Sistem a ve b parametrelerine göre farklı davranışlar göstermektedir. a=0.9 ve b=0.3 parametreleri için periyodik davranış gösterirken, a=1 ve b=0.3 parametreleri için iki periyotlu (periyod-2) davranış göstermektedir. a=1.4 ve b=0.3 parametreleri için ise kaotik davranış göstermektedir. Henon haritasında (a = 0.9, b = 0.3) parametreleri için x_n'in değişimi şekil 2.1'de, (a=1.4, b=0.3) parametreleri için ise x_n'in değişimin şekil 2.2'de gösterilmiştir.

$$x_{n+1} = 1 - ax_n^{\,2} + y_n \tag{2.1}$$
$$y_{n+1} = bx_n$$

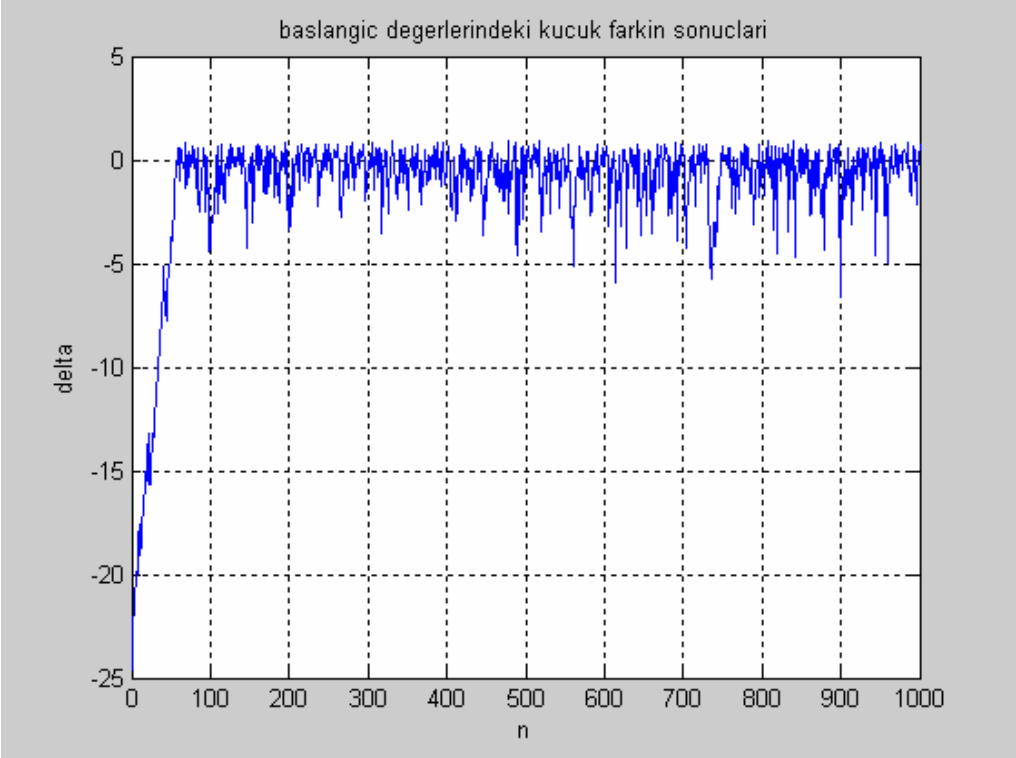
Henon haritanın başlangıç koşullarına olan duyarlılığını göstermek için (a=1.4, b=0.3) parametreleri ve (x_0=0, y_0=0) başlangıç koşulları kullanılarak elde edilen x_n verileri ile (x_0=0.00000, y_0=0.000001) başlangıç koşullarından elde edilen x_n verileri arasındaki fark şekil 2.3'de gösterilmiştir. Şekil 2.3'dende anlaşıldığı gibi sistemin başlangıç koşullarında meydana gelen ve ihmal edilebilecek kadar küçük olan bir değişim çok farklı sonuçlara neden olmuştur. Kaotik sistemlerin karakteristiğindeki küçük bir değişimin üstel artışa neden olması kaotik sistemlerin başlangıç koşullarına olan duyarlılığını göstermektedir. Bunun bir sonucu olarak kaotik sistemlerin uzun vadeli kestirimlerinde başarısız sonuçlar elde edilmektedir. Çünkü pratikte sistemin başlangıç koşullarının sonsuz sayıdaki durumu asla bilinemez. Herhangi bir hata olduğunda uzun vadeli kestirimde bulunmak anlamsız olacaktır. Bu nedenle kaotik sistemler kısa vadede yüksek kestirilebilirlik (zamanda gerekirci değişiminden dolayı) özelliğine sahipken uzun vadeli kestirimlerde bulunamazlar (başlangıç şartlarına bağlı olmalarından dolayı).



Şekil 2.1 a=0.9 ve b=0.3 parametreleri için x_n-n grafiği



Şekil 2.2 a=1.4 ve b=0.3 parametreleri için x_n-n grafiği

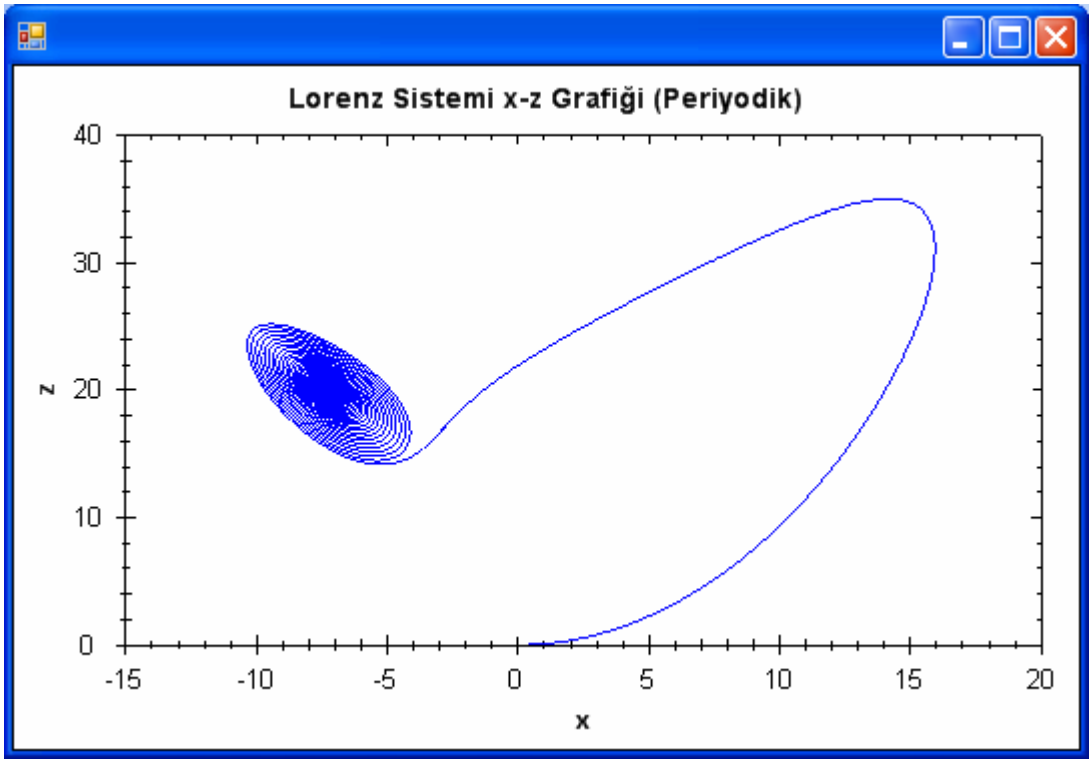


Şekil 2.3 delta= log | x_n - $x_n^{'}$ |

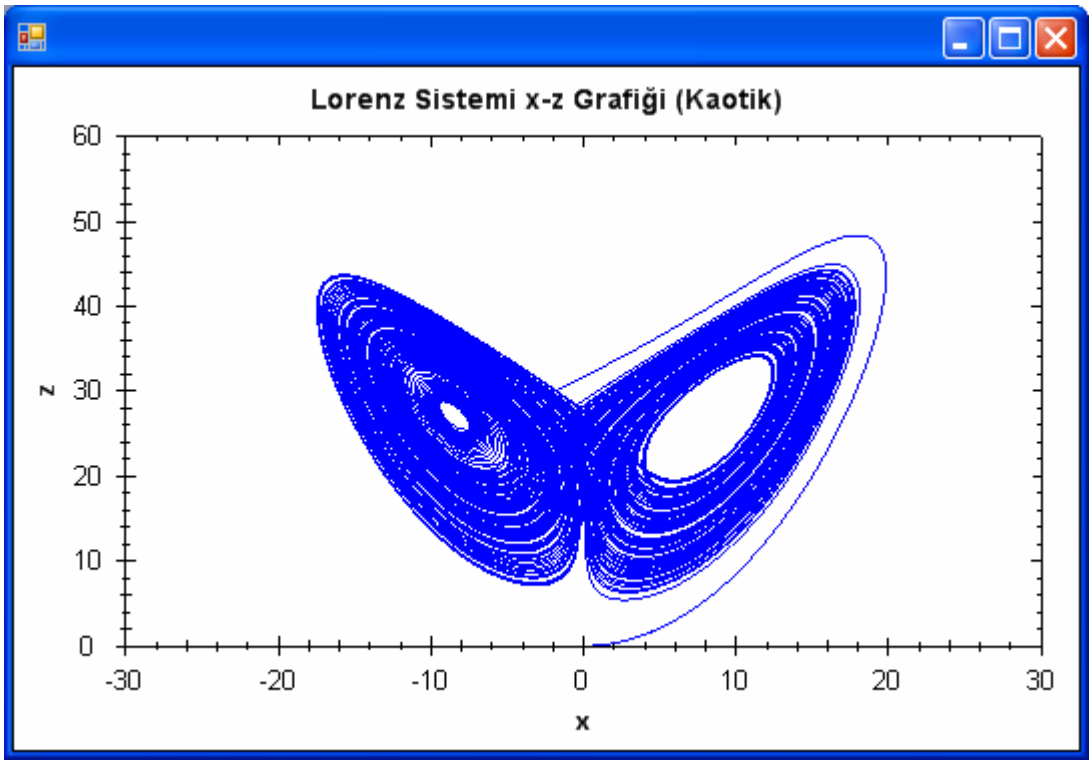
İkinci örneğimiz üç boyutlu sürekli zamanlı kaotik bir sistem olan Lorenz sistemidir. Kaos ile ilgili yapılan ilk sayısal çalışmadır. Edward Lorenz tarafından atmosferdeki havanın akışını modellemek için 1950'li yılların sonunda geliştirmiştir. Sisteme ait dinamikler denklem (2.2)'de verilmiştir.

$$\frac{dx}{dt} = a(y - x)$$
$$\frac{dy}{dt} = (bx - y - xz) \qquad (2.2)$$
$$\frac{dz}{dt} = (xy - cz)$$

Sistemin başlangıç koşulları $-20 \leq x \leq 20$, $-50 \leq y \leq 50$, $-50 \leq z \leq 50$ şeklindedir. Sistem a=10, b=21 ve c=8/3 değerleri için periyodik davranış gösterirken a=10, b=21 ve c=8/3 durumunda kaotik davranış göstermektedir. Sistemin periyodik davranış gösterdiği parametrelerde x-z değerlerinin birbirlerine göre değişimi şekil 2.4'de gösterilmiştir. Sistemin kaotik davranış gösterdiği parametre değerlerinde ise x-z değerlerinin birbirine göre değişimi sırasıyla şekil 2.5'de gösterilmiştir.



Şekil 2.4 a=10, b=21 ve c=8/3 değerleri için x-z grafiği (periyodik)



Şekil 2.5 a=10, b=28 ve c=8/3 değerleri için x-z grafiği (kaotik)

2.3 Kaos Analiz Yöntemleri

Bir sistemde kaos analizi yapılabilmesi için farklı yöntemler vardır [2]. Bu yöntemlerden en sık kullanılanlar:

- Yörüngenin izlenmesi (zaman serileri)
- Faz uzayının incelenmesi (Phase portrait)
- Poincarė haritalama
- Güç spektrumu
- Çatallaşma diyagramı (Bifurcation Diagram)
- Lyapunov üstelleri

Yukarıda verilen yöntemlerden başka yöntemler de vardır fakat fazla kullanışlı değillerdir. Özellikle çekicilerin boyut analizi yapılarak kaos analizleri yapılabilmektedir (Fraktal boyut, Lyapunov boyutu vb.) [13]. Boyut analizi sonucu çekicinin boyutu tamsayı çıkarsa, çekicinin kaotik olmadığı, boyut kesirli sayı çıkarsa çekicinin kaotik olduğu anlaşılmaktadır.

2.3.1 Yörüngenin İzlenmesi

Yörüngenin izlenmesi yöntemi en basit yöntemdir. Sistemin durum değişkenlerinden herhangi biri zaman ekseninde gözlenerek sistemin kaosa girip girmediğine bakılabilir. Bu yöntemler incelenirken dikkat edilmesi gerekenlerden biri sistemin geçici durum davranışının kalktıktan sonra gözlemlemenin yapılmasıdır. Anlatılan yöntemlerin tümünde bu duruma dikkat edilmelidir.

2.3.2 Faz Uzayı

Bu yöntemde, durum değişkenlerinin birbirine göre davranışları çizdirilerek sistemin kaosa girip girmediğine karar verilmektedir. Sistem periyodik ise, durum değişkenleri birbirine göre çizdirildiğinde kapalı bir çevrim görülür. Buna limit çevrimi de (limit cycle) denir ve sistemin periyodik davranış gösterdiği anlamına gelir. Durum değişkenlerinin birbirine göre çizimi düzensiz ve anlamsız bir şekil ise bu tip karmaşık şekiller kaosun olduğu anlamına gelmektedir.

2.3.3 Poincarė Haritalama

Birçok durumda ayrık zamanlı bir sistemi analiz etmek, sürekli zamanlı bir sistemi analiz etmekten daha kolaydır. Poincarė, bu işi başarmak için bir yöntem geliştirmiştir. Aslında bu yöntem, n. dereceden sürekli zamanlı bir dinamik sistemi (n-1). dereceden ayrık zamanlı bir dinamik sisteme dönüştürme işlemidir. Faz uzayında Poincarė yüzeyi diye bilinen bir yüzey seçilir. Bu yüzey üzerinde yörüngenin geçtiği noktalar işaretlenerek bir harita elde edilir. Poincarė yüzeyi seçilirken belirli bir kural yoktur. Tamamen kişinin tecrübesine göre yörüngenin geçtiği bir yüzey seçilir. Özerk olmayan sistemlerin ise Poincarė haritasını elde etmek daha kolaydır. Faz uzayı izlenirken belirli aralıklarla örnek alınarak Poincarė harita elde edilir. Örnekleme süresi ise özerk olmayan sistemi süren büyüklüğün periyodudur. Karmaşık sistemleri daha basit hale getirmek ve kararlılık analiz yapmak için elverişlidir [14].

Periyodik bir davranış Poincarė haritalama yöntemi ile incelenirse, sabit bir nokta elde edilir. Çünkü sistemin periyodu ile aynı zaman dilimlerinde örnekler alınırsa, hep aynı nokta alınacağından tek bir nokta görülür. Sistem periyodumsu, ise kapalı bir çevrim elde edilir. Fakat sistem kaotikse, kapalı olmayan, gelişigüzel bir fraktal şekil oluşur.

2.3.4 Güç Spektrumu

Kaotik sinyaller geniş bantlı sistemlerdir. Dolayısıyla kaotik olan bir sinyalle, kaotik olmayan bir sinyal, güç spektrumuna bakarak ayrılabilir. Eğer sistem kaotikse, güç spektrumunda süreklilik vardır. Sistem kaotik değilse, sadece belli frekanslarda sıçramalar mevcuttur.

2.3.5 Çatallaşma Diyagramı

Bu yöntem, diğer yöntemlerden biraz farklıdır. Düşey eksende sistemin asimptotik çözümlerini, yatay eksende ise kontrol parametresinin değişimini veren çizimlere çatallaşma diyagramı denilir. y ekseninde durum değişkenlerinden birinin davranışı verilmektedir, x ekseninde ise parametrelerden birinin farklı değerleri verilmektedir.

2.3.6 Lyapunov Üstelleri

Geliştirilen yazılımda kaos analizi yapmak için kullanılan yöntem Lyapunov üstelleridir. Bu yöntemin detaylarına ayrıntılı olarak bölüm 3'te değinilmiştir.

3 LYAPUNOV ÜSTELLERİ

Kaos analizinin yapılabilmesi için çeşitli yöntemler geliştirilmiştir. Bu yöntemlerden bazıları daha önce Bölüm 2.3'de açıklanmıştır. Ancak bu yöntemlerin çoğunun analiz işlemini niteliksel yöntemler kullanarak yapması kaosu niceliksel olarak gösterme noktasında yetersiz kalmaktadır. Lyapunov üstelleri (veya üstleri) olarak bilinen kaos analiz yöntemi ise diğer yöntemlerden farklı olarak kaosu niceliksel olarak inceleyen bir analiz yöntemidir.

Bu bölümün giriş kısmında Lyapunov üstellerin amacı açıklanmış, Bölüm 3.2'de üstellerin temel karakteristikleri gösterilmiştir. Daha sonra ayrık ve sürekli zamanlı sistemlerde yazılımın Lyapunov üstellerini nasıl hesaplayacağı açıklanmıştır.

3.1 Giriş

Doğrusal olmayan diferansiyel denklemlerin kararlılığının incelenmesinde değişmez üstellerin kullanılabileceğini ilk olarak 1889 yılında Stockholm Üniversitesinde profesör olan Rus matematikçi Sonya Kovalevskaya (1850–1891) göstermiştir. Kovalevskaya'nın çalışması daha sonra 1892 yılında diğer bir Rus matematikçi olan Alexandr Mikhailovich Lyapunov (1857–1918) tarafından tam olarak geliştirilmiştir. Lyapunov çalışmasında sadece Lyapunov üstelleri ile bir dinamik sistemin (zamanın bir fonksiyonu olarak) yörüngelerinin uzaklaşmasının değişimi ile ilgili düşüncelerinin temellerini açıklamıştır.

Lyapunov üstellerini kullanarak analiz yapma fikrinin çıkış noktası kaotik sistemlerin başlangıç koşullarına olan bağımlılığından kaynaklanmaktadır. Kaotik bir sistem birbirine çok yakın komşu iki başlangıç noktasından başlatıldığında yörüngelerin gittikçe birbirinden uzaklaşması veya yakınlaşması ile kaos analizi yapılabilir. Lyapunov üstelleri komşu yörüngeler aradaki bu mesafeyi ölçen matematiksel bir yöntemdir [3-4].

Lyapunov üstelleri, doğrusal sistemlerde kullanılan özdeğerlere (eigen value) benzetilir. Literatürde de özdeğerlerin doğrusal olmayan sistemlerdeki karşılığı olarak geçmektedir [3]. Sürekli zamanlı ve ayrık zamanlı sistemlerde Lyapunov üstelleri hesaplanabilmektedir. Bunun yanı sıra deney veya benzetim sonuçlarından elde edilen zaman serilerinden de Lyapunuov üstelleri hesaplanabilmektedir [15].

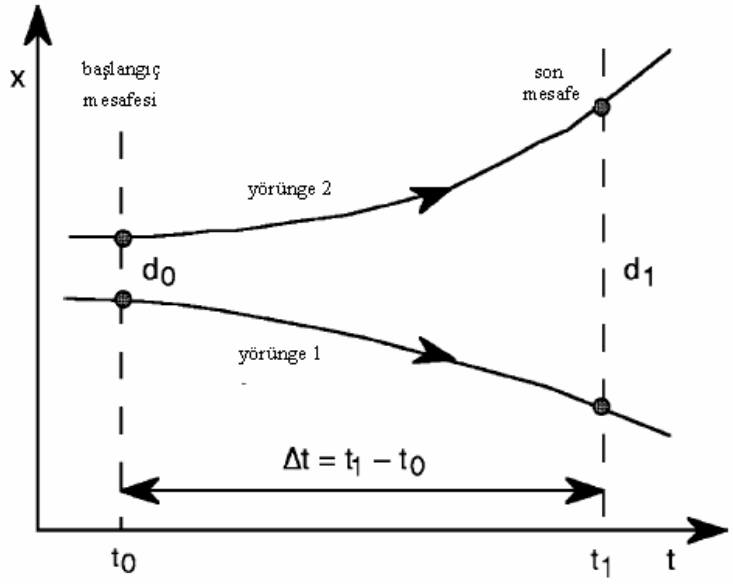
3.2 Lyapunov Üstellerinin Karakteristiği

Lyapunov üstellerinin karakteristiğini basitçe anlatmak için ilk olarak bir boyutlu akıştan (veya diferansiyel denklemlerle tanımlanan sürekli zamanlı yörüngeler) elde edilen birbirinden çok küçük farklı iki komşu yörünge incelenmiştir [16]. Yörüngeler grafiksel olarak Şekil 3.1'de gösterilmiştir. Yörüngelerin zaman periyodu $\Delta t = t_1 - t_0$ ve bu zaman periyodu boyunca aralarındaki farkın artışı ise *dt* ile tanımlanmıştır. Buna göre:

$$dt = d_0 \exp(\lambda \Delta t) \tag{3.1}$$

burada λ üsteli denklem (3.2)'den hesaplanabilir.

$$\lambda = \left(\frac{1}{\Delta t}\right) \ln\left(\frac{dt}{d_0}\right) \tag{3.2}$$



Şekil 3.1 Yörüngelerin grafiksel değişimi

Ayrık zaman aralıkları için (burada *n* Şekil 3.1'de t_0 ve t_1 zamanlarında gösterilen yörüngeler üzerindeki noktaları temsil etmektedir) elde edilen birbirinden çok küçük farklı iki komşu yörünge için zaman periyodu $\Delta n = n_1 - n_0$ ve bu zaman periyodu boyunca aralarındaki farkın artışı ise d_n ile tanımlanmıştır. Buna göre:

$$d_n = d_0 \exp(\lambda \Delta n) \tag{3.3}$$

buradan da λ üsteli denklem (3.4)'deki gibi hesaplanabilir.

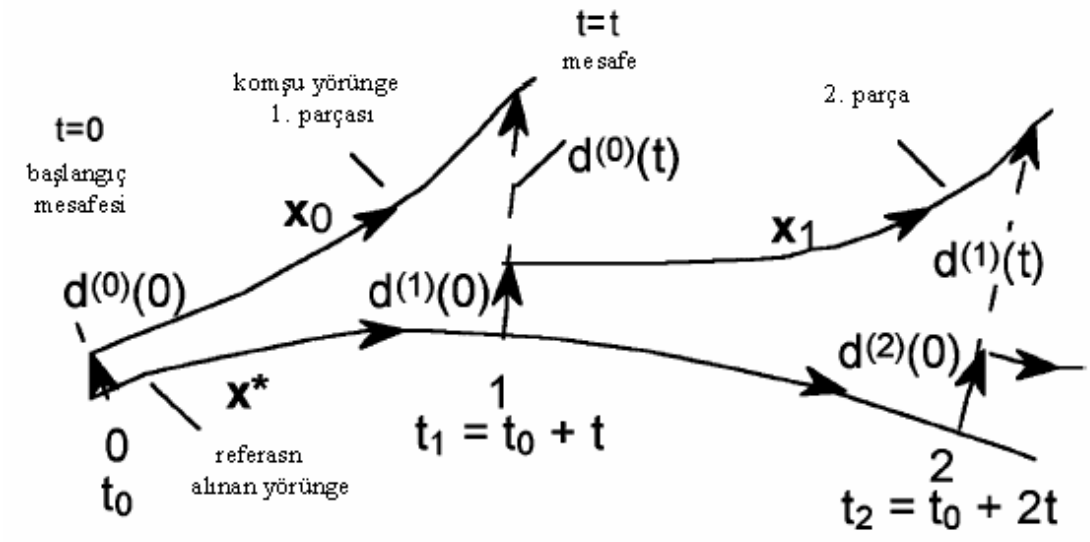
$$\lambda = \left(\frac{1}{\Delta n}\right) \ln\left(\frac{d_n}{d_t}\right) \tag{3.4}$$

şeklinde elde edilebilir.

3.2.1 Ortalama Lyapunov Üsteli

Önceki bölümünde verilen denklemlerle ayrık zamanlı veya sürekli zamanlı herhangi bir sistemde Lyapunov üstellerinin hesaplanmasının ve anlaşılmasının çok basit olduğu görülmektedir. Fakat bu hesaplamalar yeterince doğru değildir. Çünkü bu denklemlerde sadece bir yörüngenin değişimi gözlenmektedir. Sistemin bir yörüngesindeki değişim çok büyük olabileceği gibi diğer bir yörüngesinde değişim çok küçük olabilir. Bu yüzden bu değişimin hesaplanmasına diğer birçok yörüngede dâhil edilmelidir [15,16]. Sonuç olarak ortalama bir değer hesaplanır. Bu değere ortalama Lyapunov üsteli denir.

Ortalama Lyapunov üstelinin tanımı şekil 3.2'de grafiksel olarak gösterilmiştir.



Şekil 3.2 Her bir adımdaki yörünge değişimi

$$\lambda \equiv \bar{\lambda} \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} \lambda^{(j)}$$

$$\lambda = \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} \frac{1}{n^{(j)}} \ln\left(\frac{d^{(j)}(n^{(j)})}{d^{(j)}(0)} \right) \tag{3.5}$$

denklem (3.5)' de kapalı parantezler arasında verilen ifade hesaplamada *N* yörünge parçası boyunca ortalama olarak alınan yörüngelerin seçilmesinin başarısıdır. Bu ifade denklem (3.4) ile karşılaştırıldığından hatanın indirgendiği görülmektedir.

3.2.2 Lyapunov Üstellerinin Doğrudan Verilerden Hesaplanması

Analiz edilen sistemin Lyapunov üstellerini, sistem denklemlerini kullanılarak önceki bölümde verilen formüller aracılığıyla nasıl hesaplanabileceğini gördük. Ancak Lyapunov üstellerinin önemli bir diğer karakteristiği de analiz edilecek sistemin denklemleri açık şekilde bilinmese veya denklemlerin çözümleri yapılamazsa bile analiz işlemini gerçekleştirebilmesidir.

Sistemden elde edilen zaman serisi kaos içermesi durumunda bu veriler faz uzayında yeniden kurularak matematiksel modellerine ve çözümlerine ihtiyaç duymadan doğrudan Lyapunov üstelleri hesaplanabilir [15,16,17,18]. Bu üsteller kullanılarak sistemin kararlılığı incelenebilir.

Bu analiz işlemi sonraki bölümlerde ayrıntılı olarak incelenmiştir.

3.2.3 Lyapunov Üstellerinin Sistem Değişmezleri Olarak Kullanılması

Dinamik değişmezler (basitçe değişmezler) sistemin dinamik davranışlarının nicel tanımlarıdır. Eğer sistem doğrudan gözlemlenebiliyorsa dinamik değişmezlerin değeri doğrudan sistemden elde edilebilir. Dinamik değişmezlerin ölçülmesinin önemi dinamiklerin eşdeğer değişmezlerin ölçümleri ile tanımlanmasıdır [17,18].

Lyapunov üstellerinin diğer önemli bir kullanım alanı ise dinamik sistemlerin bir karakteristik ölçümü (dinamik değişmezi) gibi verilerin yeniden ölçeklenmesi, kaydırılması ve diğer dönüşümlerde kullanılabilmesidir. Buna en güzel örnek zaman serisinden elde edilen garip çekicinin yeniden kurulması verilebilir [16,17,18].

3.2.4 Genel (Global) ve Yerel (Lokal) Lyapunov Üstelleri

Lyapunov üstellerinin sistem değişmezi olarak yeniden ölçekleme, kaydırma ve diğer veri dönüşümlerinde kullanılmasına rağmen üstellerinin hesaplanmasında ortalama değerin doğru analiz yapmak için ne kadar önemli olduğunu görmüştük.

Lyapunov üstellerinin ortalama uzaklaşma hızı olarak yorumlarsak bu yalnız ve yalnız sistemin faz uzayı yerleşiminde kullanılan bir değişmez olduğu görülmektedir. Bu durumda sistemin uzun dönemli zaman periyodunda analiz yapıyorsak Lyapunov üstelinin ortalama değerine ihtiyaç duyarız ki bu değer çoğunlukla Genel (Global) Lyapunov üsteli olarak adlandırılır [16].

Eğer sistemin kısa dönemli zaman periyodunda analizi yapılıyorsa tüm yörüngeleri üstel hesabına katmak gerekmez. Bu durumda hesaplanan üstel ise Yerel (Lokal) Lyapunov üsteli olarak adlandırılır.

3.2.5 Lyapunov Üstellerinin İşaretlerinin Anlamı

Lyapunov üstelinin işareti sistemin dinamiği ile ilgili resmi görmemize yardımcı olmaktadır. Sistemin dinamiği nicelenirken en önemli nokta Lyapunov üstelinin işaretidir [15,16]. Bir boyutlu bir harita için sitemin sahip olduğu tek Lyapunov üstelinin değeri ile sistem analiz edilmektedir. Üstelin değeri pozitif olduğu durumda kaos gözlenirken, sıfır sonucu için marjinal kararalı bir yörünge ve negatif değer için ise periyodik bir yörünge ortaya çıkmaktadır.

Üç boyutlu sürekli zamanlı yayılımlı bir dinamik sistem için ise üstellerin sahip olabileceği birkaç olasılık söz konusudur. Sistemin sahip olduğu üstellerle sisteme ait çekici tanımlanabilmektedir

- Üstellerin işaretleri (+,0,-) ise garip çekici
- (0,0,-) şeklinde ise iki torus
- (0,-,-) durumunda limit çevrim
- Son olarak üsteller (-,-,-) işaretlerine sahiplerse denge noktasıdır.

Sürekli zamanlı dört boyutlu yayılımlı bir dinamik sistem içinse garip çekicinin olası üç tipi vardır. Lyapunov üstellerinin işaretlerinin değişimi:

- (+,+,0,-) hiper kaos gözlenir (sistem birden fazla pozitif Lyapunov üsteli içeriyorsa bu durum hiper kaos olarak adlandırılmaktadır. Rössler sistemi bir hiper kaos çekicisidir).
- (+,0,0,-) kaos gözlenir.
- (+,0,-,-) torus kaos durumu gözlenir

3.2.6 Lyapunov Boyutu

Kaplan ve Yorke [19] bir dinamik sistemin garip çekicisinin karmaşıklığını ölçmek için Lyapunov üstellerini kullanmışlardır. Basit biçimde iki boyutlu harita için Lyapunov boyutu denklem (3.6)'daki gibi verilebilir.

$$D_{L_y} = 1 + \left(\frac{\lambda_1}{|\lambda_2|} \right) \tag{3.6}$$

burada daha büyük üstel olan λ_1 pozitiftir λ_2 ise negatiftir ve $\lambda_1 < |\lambda_2|$ şeklindedir.

Lyapunov boyutunun daha genel tanımı ise aşağıdaki gibidir.

$$D_{L_y} = K + \left(\sum_{j=1}^{K} \frac{\lambda_j}{|\lambda_{K+1}|} \right) \tag{3.7}$$

3.3 Ayrık Zamanlı Sistemlerde Lyapunov Üstellerinin Hesaplanması

Bu bölümde Lyapunov üstellerinin haritalar (ayrık zamanlı sistemler) üzerinde inceleyeceğiz. Ayrık zamanlı sistemlerde Lyapunov üstelleri genel olarak haritanın türevinden elde edilebilir. Burada gösterilen sistem bir boyutlu olduğu için sadece bir Lyapunov üsteli içermektedir. Aynı mantıkla yöntem n boyutlu n Lyapunov üsteline sahip haritalar içinde genelleştirilebilir.

Tekrarlı (iteratif) yapıya sahip bir harita aşağıdaki gibi verilebilir.

$$x_{n+1} = g(x_n) \tag{3.8}$$

burada $g(x_n)$ haritası mevcut x_n değerinden sonra gelen x_{n+1} değerini kestirmek için kullanılır. Bu yüzden ayrık zaman serisi n düzenli aralıklarıyla etiketlenen $\{x_n\}$ değerlerini üretmektedir. Sistem gerekircidir. Bir boyutlu haritalara en güzel örnek Lojistik haritadır.

Bir boyutlu $x_{n+1} = g(x_n)$ haritası için λ Lyapunov üsteli x_0 başlangıç koşulu için $\lambda(x_0)$ gösterilmektedir. Ölçülen ortalama hatanın her tekrardaki artış hızı veya eşdeğer olarak x_0 yakın tekrarlarının süresince bilginin ortalama kaybıdır. Eğer g haritası açık şekilde biliniyorsa

Lyapunuv üstelleri kolaylıkla hesaplanabilir. İki komşu nokta x_0 ve $(x_0+\Delta x_0)$ seçilir. Bir adım sonra ayrıklaşma

$$\Delta x_1 = g(x_0 + \Delta x_0) - g(x_0) \tag{3.9}$$

veya göreceli ayrıklaşma

$$\left(\frac{\Delta x_1}{\Delta x_0}\right) = \left\{\frac{[g(x_0 + \Delta x_0) - g(x_0)]}{\Delta x_0}\right\} \tag{3.10}$$

Burada fonksiyonun ilk türevi kullanılarak

$$\lim_{\Delta x_0 \to 0}\left(\frac{\Delta x_1}{\Delta x_0}\right) = g'(x_0) \tag{3.11}$$

Noktalar arasındaki üstel hata artışı ise

$$\left|\frac{\Delta x_1}{\Delta x_0}\right| = b\lambda \tag{3.12}$$

b logaritmanın tabanı olarak alınabilir.

$$\lambda \log_b(b) = \log_b\left|\frac{\Delta x_1}{\Delta x_0}\right| = \log_b|g'(x_0)| \tag{3.13}$$

veya

$$\lambda = \log_b|g'(x_0)| \tag{3.14}$$

Denklemdeki mutlak değer ifadesi logaritmanın reel ve işaretinin önemsiz olduğunu göstermektedir. Sonuç olarak:

- Eğer $\lambda > 0$ ise harita başlangıç koşullarına duyarlı
- Eğer $\lambda \leq 0$ ise harita başlangıç koşullarına duyarsızdır.

Noktadan noktaya yerel uzaklaşma oranı değiştiğinden dolayı noktaların geniş bir kümesi için genel Lyapunov üsteli aşağıdaki gibi hesaplanır.

$$\lambda \equiv \bar{\lambda} = \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} \log_b \left| g'(x_j) \right| \tag{3.15}$$

Logaritmanın tabanı $b=e$ şeklinde seçilmektedir

Genellikle sistemde bilinmek istenen g haritasının $n.$ adımındaki işlemlerin yapılmasıdır x_0 başlangıç koşulu için

$$x_n = g(x_{n-1}) = g_{(n)}(x_0) = g(g(g(...(g(x_0))...))) \tag{3.16}$$

şeklinde ifade edilir. Çünkü

$$\Delta x_n = g_{(n)}(x_0 + \Delta x) - g_{(n)}(x_0) \tag{3.17}$$

benzer şekilde yerel (lokal) Lyapunov üstelide

$$\lambda(x_0) = \log_b \left| g_{(n)}(x_j)' \right| = \log_b \prod_{j=0}^{N-1} \left| g'(x_j) \right|$$

$$\lambda(x_0) = \sum_{j=0}^{N-1} \log_b \left| g'(x_j) \right| \tag{3.18}$$

ve genel (global) Lyapunov üstelide

$$\lambda(x_0) \equiv \bar{\lambda} = \lim_{N\to\infty} \frac{1}{N} \sum_{j=0}^{N-1} \log_b \left| g'(x_j) \right| \tag{3.19}$$

gösterilmektedir [16].

Ayrık zamanlı sistemler için Lyapunov üstellerinin hesaplanmasında kullanılan algoritmanın yapısı kabaca aşağıdaki gibidir.

Üstelleri hesaplayan algoritmanın sözde kodu aşağıda verilmiştir [20,21].

N: İterasyon sayısı
fark: Komşu yörüngeler arasındaki başlangıçtaki fark
x, x1: Komşu yörüngeler
mesafe: Komşu yörüngeler arasındaki uzaklık
toplam: her bir adımda meydana gelen mesafelerin toplamı
landa: Lyapunov üsteli

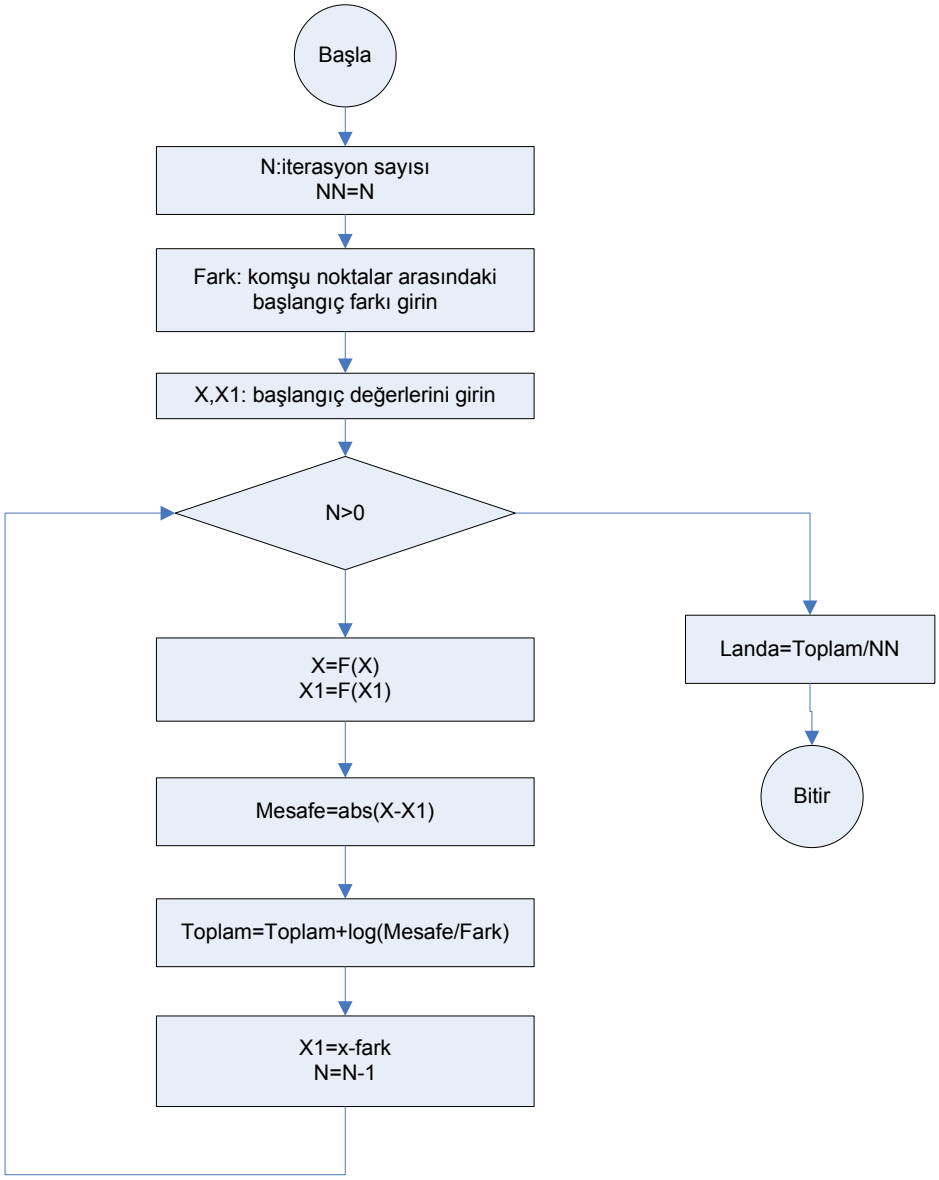
```
tekrarla 1 den N
        x=f(x)
        x1=f(x1)
        mesafe=mutlak değeri al (x-x1)
        toplam=toplam + logaritma al (mesafe / fark)
        x1=x-fark
bitir
landa = toplam / N
```

Üstelleri hesaplayan algoritmanın akış şeması ise şekil 3.3'de verilmiştir.

3.4 Sürekli Zamanlı Sistemlerde Lyapunov Üstellerinin Hesaplanması

Ayrık zamanlı sistemler için Bölüm 3.3'de verilen Lyapunov üstellerinin tanımını bu bölümde sürekli zamanlı sistemler için genişletilecektir. Aralarındaki tek fark haritaların ayrık iterasyonları yerine, diferansiyel denklemin sürekli akışının kullanılmasıdır.

Gerçekleştirilen yazılımda sürekli zamanlı sistemlerin Lyapunov üstellerinin hesaplanması için öncelikle sistemin; zamanın herhangi bir anındaki değerinin diferansiyel denklemlerinden hesaplanması gerekmektedir. Bunun için çeşitli yöntemler mevcuttur. Gerçekleştirilen yazılımda ise bu hesaplamalar Euler yöntemi kullanılarak yapılmıştır. Bu hesaplamaların ardından üstellerin değerini hesaplamak için ayrık zamanlı sistemler için Bölüm 3.3'de verilen algoritma çalıştırılmaktadır.



Şekil 3.3 Algoritmanın akış şeması

4 GELİŞTİRİLEN YAZILIMININ MODELİ

Bir kavram olarak sistem bir veya daha çok amaca veya sonuca ulaşmak üzere aralarında ilişkiler olan fiziksel veya kavramsal birden çok bileşenin oluşturduğu bir bütündür. Bilgisayar sistemleri de insan yaşamını her alanında yoğun bir şekilde etkilemektedir. Bunun sonucu olarak bilgisayar sistemlerinin en önemli parçalarından biri olan yazılım sistemleri ön plana çıkmaktadır. Yazılım sistemlerine bankacılıktan sağlık bilgi sistemlerine, eğitimden eğlence sektörüne, şirket yönetiminden iletişim sistemlerine kadar çok geniş alanlarda rastlanmaktadır.

Bir mühendislik ürünü olan yazılım sistemlerinin başarısı için öncelikle bu sistemlerin modellenmesi gerekmektedir. Bu bölümde ilk olarak modelleme ve yazılım kavramları açıklanacak daha sonra yazılım modelleme dili olan UML (Unified Modeling Language) kullanılarak geliştirilen yazılımın modeli verilecektir.

4.1 Modelleme

Model, gerçek dünyadaki bir olgunun veya sistemin yapı ve işleyişinin, ilgili olduğu bilin sahasının kavram ve kanunlarına bağlı olarak ifade edilmesidir. Model gerçek dünyadaki bir olgunun bir anlatımıdır. Modelleme tüm bilim ve mühendislik dallarında sistemlerin incelenmesinde ve tasarlanmasında kullanılabilir [22,23].

Gerçek hayattaki sistemlerin genellikle doğrusal olmaması, karmaşık hatta kaotik olması sistemi tüm yönleriyle bir defada kavranabilmesini zorlaştırmaktadır. Bu problem sistemlerin parçalar halinde incelenmesi ve tasarlanması ile giderilebilir.

Modellerin anlatmak istedikleri olgu ve sistemleri basitleştirerek belli varsayımlar altında ele almasından dolayı; modeller gerçek sistemin yalınlaştırılmış biçimidir. Ne kadar karmaşık görünseler de gerçeğin eksik bir yorumudur [23].

Modelleme bir çeşit yorumlama olduğu için bir olgu ile ilgili birden fazla model kurulabilir. Bu modeller birbirlerinden farklı olmakla birlikte olgunun belli bazı özelliklerini anlatımda biri diğerine göre daha iyi veya daha kötü olabilmektedir.

Modelleme aslında bir çeşit soyut düşünme yeteneğidir. Sistemin özelliklerinden o anda ilgilendiklerini öne çıkarırken diğerlerinin geriye atan soyut yapılar kurar. Burada önemli olan modelin o anda neleri içerip neleri dışarıda bırakmasının uygun olduğuna karar vermektir. Kısaca modelleme sistemin karmaşıklığını indirgemek için bir filtre görevi görür. Yani modelleme sürecinin başarısı modellemenin soyut düşünme yeteneği ile doğru orantılıdır [22].

Modeller sistem karmaşıklığını yönetilir boyutlara indirgemenin yanı sıra tasarımcılar arasında bir iletişimi biçimi olarak da hizmet verirler. Bir tasarımın genel özelliklerinin açıklanması ve çeşitli seçeneklerinin değerlendirilmesi bir model üzerinde daha etkin olarak yapılabilir. Bunun yapılabilmesi için modellemede ortak bir gösterim biçimin, başka bir deyişle ortak bir modelleme dilinin kullanılması zorunludur [22,23].

4.2 Yazılım

Yazılım katma değer yaratan endüstriyel bir üründür. Kurumlar, firmalar ve bireyler tarafından, etkinliği ve rekabet gücünü artırmak, hizmet ve ürün kalitesini yükseltmek, üretimde kaynak ve zaman tasarrufu sağlamak için satın alınır ve kullanılır. Bilgisayarların her geçen gün daha çok kullanıcı tarafından, daha değişik amaçlar için kullanılması yazılım sektörünün sürekli olarak gelişmesini sağlamaktadır. Bu büyümenin sonucunda da daha yetenekli yazılımlara duyulan gereksinimde artmaktadır [24].

Herhangi bir üretim sürecinde elde edilen ürünün üretiminde kullanılan yöntemlerin maliyeti ve verimliliği ön plana çıkarmaktadır. Yazılımda bir ürün olduğu için üretim sırasında kullanılan tekniklerin verimliliklerinin ölçülmesi bir zorunluluktur.

Ancak yapılan çeşitli araştırmalar sonucunda elde edilen sonuçlar göre yazılım projelerinin %30-%40 kadarı proje bitmeden iptal edilmekte, Yazılım projelerinden sonlandırılabilenler başlangıçta öngörülenin iki katına mal olmaktadır ve sadece bu projelerden %15-%20 kadarı öngörülen zaman ve maliyetler içerisinde bitirilebilmektedir. Bu rakamlar sektörden sektöre ve araştırmadan araştırmaya farklılık göstermekle birlikte, ana hatlarıyla doğrudur. Sonuç olarak yazılım projelerinin öngörülebilen maliyetler içerisinde ve öngörülen zamanda tamamlanabilmesi için ciddi bir proje yönetimi ve mühendislik tasarımı yapılması gerekmektedir [25].

Yazılım Mühendisliğinin ana hedefi, yazılım sistemlerinin mühendislik prensipleri çerçevesinde tasarımı, üretimi ve işletilmesidir. Yazılım mühendisliğinin temelleri, yazılım mühendisliğinin ürettiği ürünlerin niteliklerini anlatan teorik, bilimsel ve matematiksel temellerden öngörülebilir sonuçlar üretmektir. Buradaki temel hedef, kaynakları belirlenmiş bir amaca dönüştürmek için mühendislik tasarımı ve mühendislik biliminin uygulanarak en uygun modellemenin yapılabilmesidir.

Yazılım sistemlerin incelenmesi ve standart olarak kullanılan modelleme dili UML (Unified Modeling Language) olarak adlandırılmaktadır. UML görsel öğeler bakımından zengin bir modelleme dilidir. Görsel öğeler içeren modellerin düz metin ve simgelerin yanında daha

etkili olması UML'nin etkinliğini artırmaktadır. UML modelleme dili 1994 yılına kadar geliştirilen nesneye yönelik modelleme yöntemlerinden sektörde genel beğeni kazanmış olanların harmanlanmış bir biçimidir.

4.3 UML (Unified Modeling Language)

Yazılımda sistem modelleme süreci problemin tanımın yapılması ve bu probleme çözüm oluşturulması aşamasını içerir. Bu iki aşama genellikle sistem çözümleme ve tasarım olarak adlandırılmaktadır. Sistem çözümleme ve tasarım aşamasında sistemin değişik modelleri oluşturulur. Her model sistemin değişik bir açıdan incelenmesini sağlar ve yazılımcının sistemi kavramasında, çözüm seçenekleri oluşturmasında ve müşteriyle ve kendi meslektaşlarıyla görüş alışverişinde bulunmasına yardımcı olur. UML modelleri yalnızca insanlar tarafından kullanılmakla kalmayıp, otomatik kod üreteçleri gibi değişik araçlar tarafından okunabilme özelliğine de sahiptir [26].

Yazılımcının ilk aşamada yapması gereken çalışma kullanıcı gereksinimlerini belirlemesidir. Gereksinin belirleme aşaması, bir yazılım projesindeki en kritik aşamadır. Bu aşamada değişik mesleklerden birçok insan bir sistem oluşturmak için bir araya gelir. Grupta temel olarak müşterinin alan uzmanları ve yazılım işini üstlenen firmanın teknik adamları yer alır. Sistemin gerçekleştirimin yapacak yazılımcılar genellikle geliştirecekleri uygulama hakkında alan uzmanları kadar bilgi ve deneyime sahip değildir. Buna karşılık müşterinin de yazılım konusunda bilgisi sınırlıdır. Genellikle daha önce bir proje geliştirme ortamında birlikte çalışmamış ve birbirini tanımayan bu insanlar arasında ortak bir terminolojinin de eksikliği de göz önüne alınırsa pek çok yanlış anlama ve iletişim başarısızlığı yaşanması kaçınılmazdır. Bu belirsizlik ortamında yazılımcının birincil görevi uygulamanın temel kavramlarını hemen özümsemek, uygulama için hayati önemi olan gereksinimleri diğerlerinden ayırmak ve bu anlayıştan kaynaklanan bir gereksinin raporunu müşteriye sunmaktır [26,27].

Müşteri raporu değerlendirdiğinde belirttiği gereksinimlerin yazılımcı tarafından doğru olarak algılandığını görebilmelidir. Gereksinim Belirleme Raporu sistemin temel yeteneklerini, gereksinimlerin önceliği ve bunların ele alınışı konusunda müşteri ile yazılımcının görüş birliği içerisinde olup olmadığını belirten en önemli belgedir. İyi yazılmış ve müşterinin beğenisini kazanan bir rapor., müşterinin proje konusundaki kaygılarını büyük ölçüde ortadan kaldıracak, alan uzmanları ve yazılımcı arasındaki iletişimi ve işbirliğini güçlendirecek, ortada önemli yanlış anlaşılmaların olmadığı konusunda her iki tarafta da güven oluşmasını sağlayacaktır.

Kötü yazılmış, eksikler ve yanlışlarla dolu bir rapor ise projenin daha ilk aşamada kriz ortamına sürüklenmesine neden olur.

Gereksinim belirleme raporu yazılım ekibi tarafından yazılımcılar için yazılmış bir rapor değildir. Yazılımcı tarafından uç kullanıcının değerlendirilmesi için yazılmıştır ve bu nedenle yazılım mühendisliğinin anlaşılması uzmanlık gerektiren teknikleri bu raporda kullanmamalıdır. Öte yandan, rapor aynı zamanda yazılım geliştirme sürecinin diğer aşamalarında da çok belirleyici roller üstlendiğinden belirli bir teknik disiplin içinde yazılmalıdır. Sonuç olarak rapor katı teknik yöntemlerle, tümüyle serbest bir yaklaşım arasında bir denge içerecek biçimde hazırlanması gereklidir [26].

Yazılım sistemlerinin modellerinde aşağıdaki UML diyagramlarından bir yada birkaçının çizilmesi gerekir. Yaygın olarak kullanılan UML diyagramları

- Kullanıcı Senaryosu Diyagramı (Use-case Diagrams)
- Sınıf Diyagramları (Class Diagrams)
- Nesne Diyagramları (Object Diagrams)
- Ardıl Etkileşim Diyagramları (Sequence Diagrams)
- İşbirliği Diyagramları (Collaboration Diagrams)
- Durum Diyagramları (State Diagrams)
- Etkinlik Diyagramları (Activity Diagrams)
- Bileşen Diyagramları (Component Diagrams)
- Yaygınlaştırma Diyagramları (Deployment Diagrams)

Kullanıcı senaryosu diyagramları (Use-case Diagrams) programın davranışının bir kullanıcı gözüyle incelenmesidir. Gerçek dünyada insanların kullanacağı bir sistemde bu diyagramlar büyük önem taşırlar.

Sınıf diyagramları (class diagrams), gerçek dünyada eşyaları nasıl araba, masa, bilgisayar şeklinde sınıflandırıyorsak yazılımda da birtakım benzer özelliklere ve fiillere sahip grupları gösteren diyagramlardır. Bu gruplar sınıf (class) olarak adlandırılır.

Bir nesne (object), sınıfın (class) bir örneğidir. Nesne diyagramlarında (object diagrams) sınıfın yerine gerçek nesneler kullanılır

Sınıf ve Nesne diyagramları statik bilgiyi modeller. Fakat gerçek zamanlı sistemlerde zaman içinde değişen eylemler bu diyagramlarla gösterilemez. Bu tür zamanla değişen durumları belirtmek için ardıl etkileşim diyagramları (sequence diagrams) kullanılır.

Bir sistemin amacının yerine gelmesi için sistemin bütün parçaları işlerini yerine getirmesi gerekir. Bu işler genellikle birkaç parçanın beraber çalışmasıyla mümkün olabilir. Bu tür ilişkileri göstermek için İşbirliği Diyagramları (collaboration diagrams) gösterilir.

Durum diyagramları (state diagrams), gerçek nesnelerin herhangi bir zaman içindeki durumunu gösteren diyagramlardır. Örneğin Ali nesnesi insan sınıfının gerçek bir örneği olsun. Ali 'nin doğması, büyümesi, gençliği ve ölmesi durum diyagramları ile gösterilir.

Bir nesnesinin durumu zamanla kullanıcı tarafından ya da nesnenin kendi içsel işlevleri tarafından değişebilir. Bu değişim sırasını etkinlik diyagramlarıyla (activity diagrams) gösterilir.

Özellikle birden çok geliştiricinin yürüttüğü projelerde sistemi bileşen dediğimiz parçalara ayırmak, geliştirmeyi kolaylaştırır. Sistemi öyle modellememiz gerekir ki her geliştirici ötekinden bağımsız olarak çalışabilsin. Bu tür modellemeler Bileşen Diyagramlarıyla (component diagrams) yapılır.

Yaygınlaştırma diyagramlarında (deployment diagrams), sistemin fiziksel incelenmesi yapılır. Mesela bilgisayarlar arasındaki bağlantılar, programın kurulacağı makineler ve sistemimizdeki bütün aletler Yaygınlaştırma Diyagramında gösterilir.

4.4 Geliştirilen Yazılımın Modeli

Bu bölümde gerçekleştirilen yazılımın önce gereksinimleri kısaca belirlenmiştir. Daha sonra UML diyagramlarından sadece Kullanıcı Senaryosu Diyagramı (Use-case Diagrams) ve Sınıf Diyagramları (Class Diagrams) verilmiştir. Yazılım sisteminin diğer UML diyagramları yazılım geliştiricilere yönelik olduğu için verilmesine gerek görülmemiştir.

4.4.1 Yazılımın Gereksinimleri

Gerçekleştirilen yazılımda ilk gereksinin kaos analizi yapılacak sistemin girilmesidir. Sistem girişi yapılırken dikkat edilecek nokta sistemin ayrık zamanlımı yoksa sürekli zamanlımı olduğudur. Kaosun hem sürekli zamanlı hem de ayrık zamanlı sistemlerde gözlemlenebilir olmasına rağmen kaos analizi yapabilmek için gerek şartlar bu iki sistem için değişkenlik göstermektedir. Bu nedenle sistemin bir diğer gereksinimi kaos analizi yapılacak sistem tipinin belirlenmesidir.

Sistem tipi belirlendikten sonra ikinci adım sistemin dinamiklerini belirleyen denklem veya denklemlerin girilmesidir. Denklem girişi sırasında yazılımın kullanacağı ayrıştırma

altprogramı (parser routine) denklemleri kolaylıkla çözümleyebilmeli ayrıca sin, cos, e^x, log v.b. gibi matematiksel fonksiyonları hesaplayabilmelidir.

Kaos teorisin en önemli karakteristiklerinden biri başlangıç koşullarına olan duyarlılığıdır. Bu noktada sistemin başlayacağı ilk koşul veya koşullarda yazılımda girilmelidir. Başlangıç koşulu değerinin istenilen hassasiyette girilebilmesi için yazılımın derleyicisinin izin verdiği en üst sınır kullanılmalıdır. Kullanıcı istediği taktirde başlangıç değerlerinin hassasiyeti kendi girebilmelidir.

Lyapunov üstellerinin komşu yörüngeler arasındaki ortalama uzaklaşmayı hesapladığı daha önce gösterilmişti. Bu değerin hesaplanması için birbirinden çok küçük faklı iki başlangıç koşulu alınması gerektiği bilinmektedir. Sistemin başlangıç koşulu bir önceki adımda girilmişti bu adımda ise komşu noktalar arasındaki başlangıç fark değeri sisteme girilmesi gerekmektedir.

Lyapunov üstellerinin doğru bir sistem değişmezi olarak kullanılabilmesi için ortalama bir değer hesaplaması gerektiği bilinmektedir. Bu gereksinimde sistemin toplam kaç adım (iterasyon) devam edeceğinin belirlenmesine ihtiyaç duymaktadır.

Yazılım Lyapunov üstellerinin ortalama değerini hesaplarken her adımda üstellerin değerleri güncellenebileceği gibi kullanıcının girdiği adım sayısında bir de güncellenebilir. Yazılım bu noktada da kullanıcıya seçim yapma imkânı verebilmelidir.

Bir sisteme bir giriş uygulandığı zaman sistem karakteristiğinden dolayı zamanla yok olan geçici bir durum oluşmaktadır. Bu geçici durum, sistemin kararlı durumda vereceği tepkiye etki etmektedir. Bu problemi ortadan kaldırmak için istenilirse üstellerin hesaplanan başlangıçtaki değerleri ortalama Lyapunov üstelinin değerine eklenmeyebilir.

Yukarıda açıklanan sistemin sınır koşulları olan komşu noktalar arasındaki fark, toplam adım sayısı, üstellerin güncelleneceği adım sayısı ve hesaplamadan önce çıkarılacak adım sayısıyla ilgili değerler her defasında girilmesi kullanıcıyı sıkacağından varsayılan değerler belirlenmesi ve istenildiği takdirde kullanıcının bu değerleri değiştirebilmesi gerekmektedir.

Kullanıcının seçimine bağlı olarak Lyapunov üstellerinin değişimi grafiksel olarak da gösterilebilmelidir. Çünkü grafikler karmaşık matematiksel modellerin daha kolay anlaşılmasını sağlamaktadır. Bu doğrultuda kullanıcı grafiğin rengini ve başlığını, x-ekseni etiketi, y-ekseni etiketi gibi özellikleri girilebilmesine izin verilmelidir.

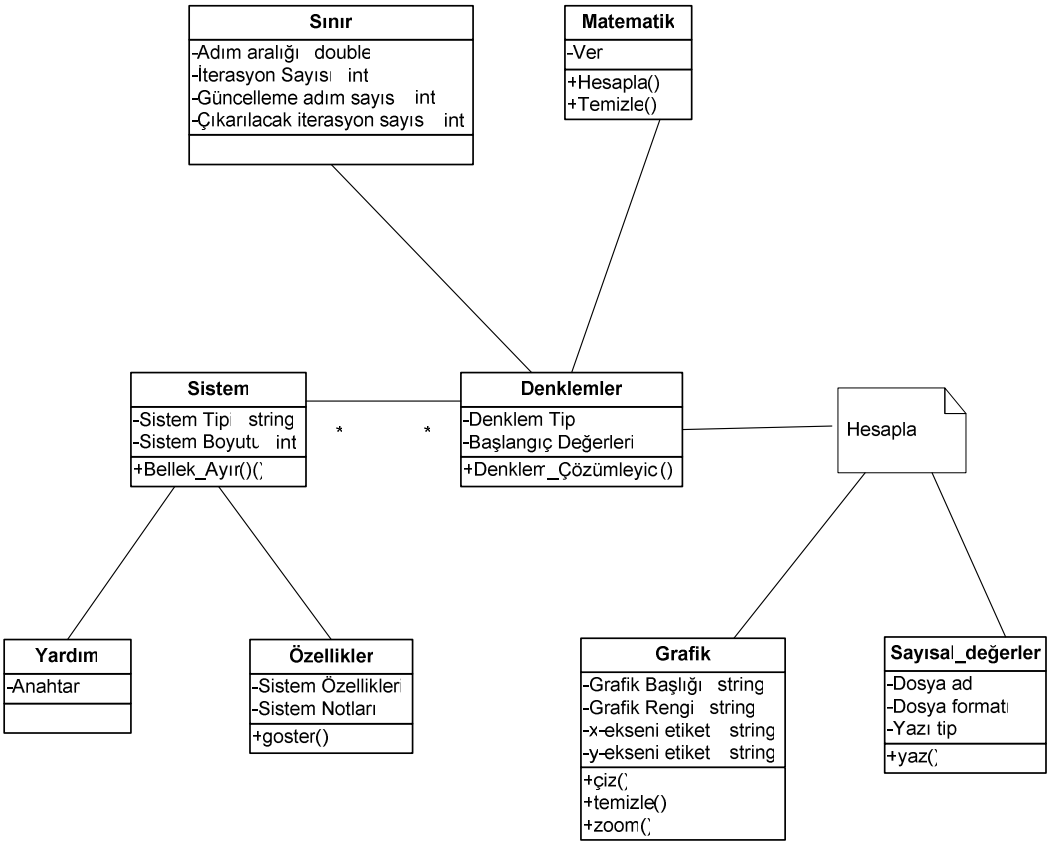
Yine kullanıcı seçimine bağlı olarak Lyapunov üstellerinin her adımda hesaplanan değerleri sayısal olarak gösterilebilmeli ve kullanıcının seçmesi durumunda bu değerler kullanıcının seçeceği dosyaya istediği biçimde kaydedilebilmelidir.

Yazılım ayrıca kullanıcıya girdiği sistemin özelliklerini bir defada gösterebilecek bir arayüz sunmalıdır. Yazılımın sahip olması gereken en önemli gereksinim ise her bir aşamada kullanıcıya zengin yardım menüleri sunmak olmalıdır.

Ayrıca gerçekleştirilen yazılım herkes tarafından kolay kullanılabilir olmasının yanında tasarım mimarisi olarak modüler biçimde tasarlanmalı, kodlar daha sonra yeniden düzenlenebilecek ve geliştirilebilecek biçimde tasarlanmalıdır. Ayrıca programın yazılacağı programlama dili herkes tarafından kolayca elde edilebilir, platformdan bağımsız ve hızlı olmalıdır.

4.4.2 Yazılımın Sınıf Diyagramları

Yazılımın Sınıf diyagramları şekil 4.1'de verilmiştir.



Şekil 4.1 Yazılımın sınıf diygaramları

4.4.3 Yazılımın Kullanıcı Senaryosu

Aşağıda yazılımı kullanacak bir kişinin izleyeceği adımlar gösterilmiştir.

Adım 1 : Sistem tipini giriniz

Adım 2 : Sistemin boyutunu belirleyiniz

Adım 3 : Sisteme ait denklemleri giriniz

Adım 4 : Sisteme ait başlangıç değerlerini giriniz

5-10 arasındaki adımlar kullanıcı seçimine bağlıdır istenilirse 11. adımdan devam edilebilir

Adım 5 : Yazılımın uygulanacağı toplam adım sayısını giriniz

Adım 6 : Komşu noktalar arasındaki uzaklığı giriniz

Adım 7 : Lyapunov üstellerinin güncellenme aralığını giriniz

Adım 8 : Hesaplamalardan önce çıkarılacak adım sayısını giriniz.

Adım 9 : Grafik arayüzü ile ilgili değerleri giriniz

Adım 10: Üstellerin sayısal değerlerini gösteren arayüz ile ilgili değerleri giriniz

Adım 11: Sistemi çalıştınız

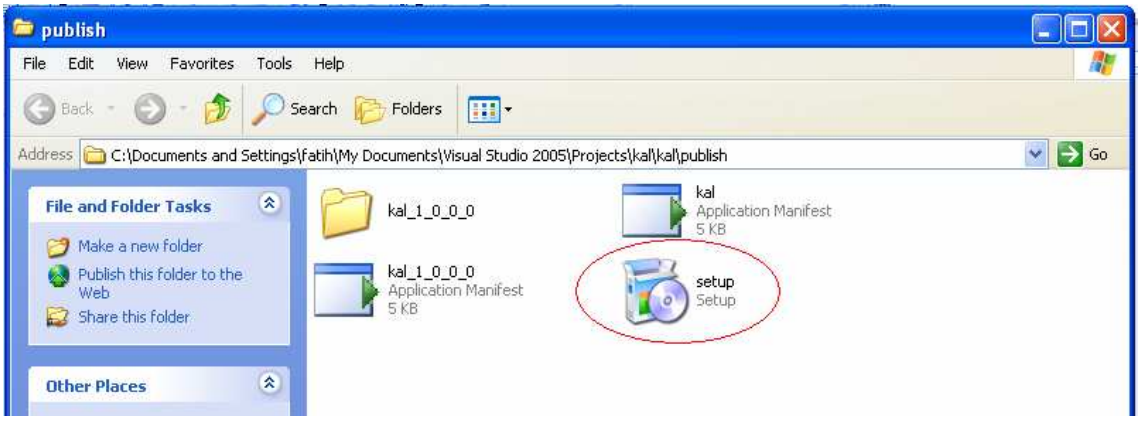
Bu bölümde belirtilen yazılım gereksinimlerinin yazılım üstünde gerçekleştirilmiş biçimleri Bölüm 5 içerisinde gösterilmiştir.

5 GELİŞTİRİLEN YAZILIMININ KULLANIMI

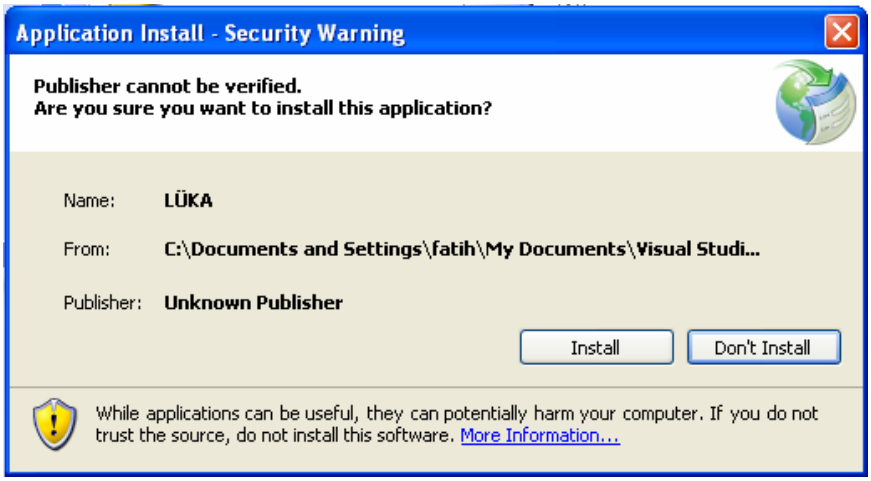
Bu bölümde yazılımın kurulmasından içerdiği her bir parçanın (modülün) açıklanmasına kadar özellikleri ayrıntılı olarak verilmiştir.

5.1 Yazılımın Kurulması

Yazılımın sisteme kurulması için; kurulum CD'si içerisindeki setup dosyasının çalıştırılması yeterlidir. Şekil 5.1'de kurulum CD'sinin içeriği gösterilmiştir.



Şekil 5.1 Kurulum CD'sinin içeriği



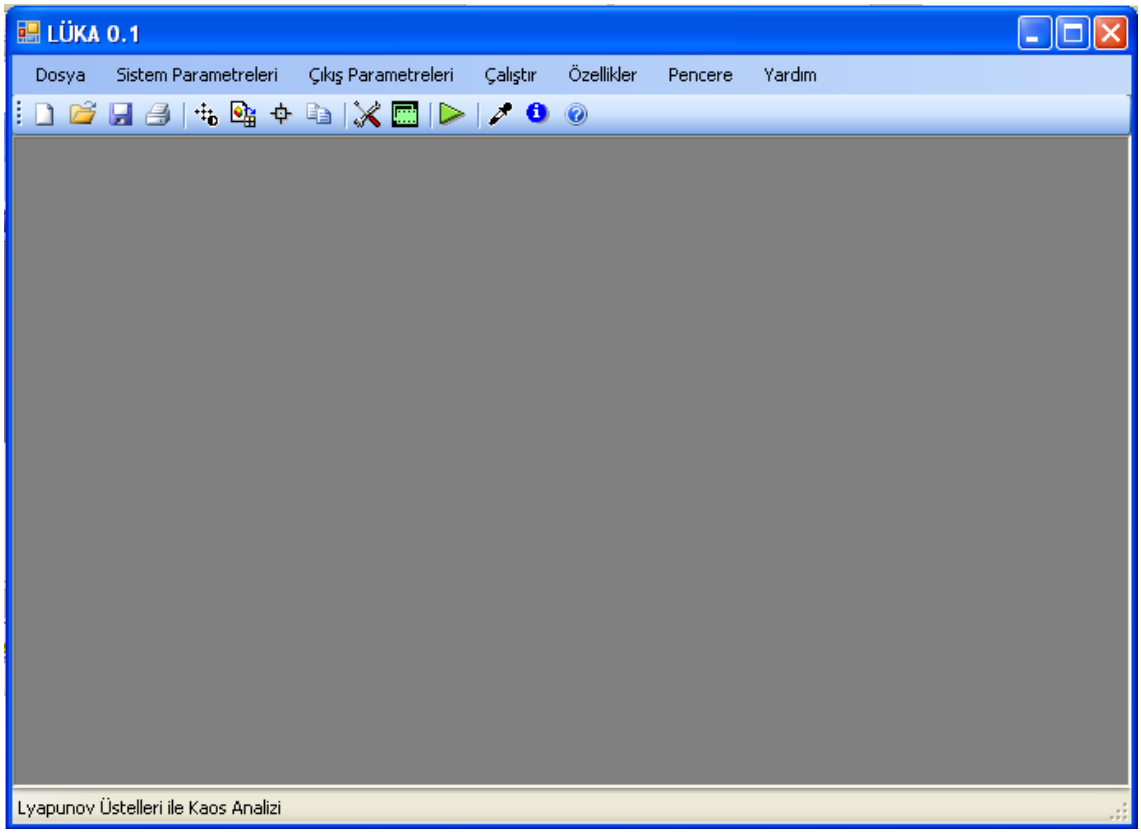
Şekil 5.2 Kurulumun başlangıcı

Kurulum tamamlandıktan sonra yazılıma başlat menüsü→programlar altında erişilebilir.

Şekil 5.3 Başlat menüsü altında yazılımın yeri

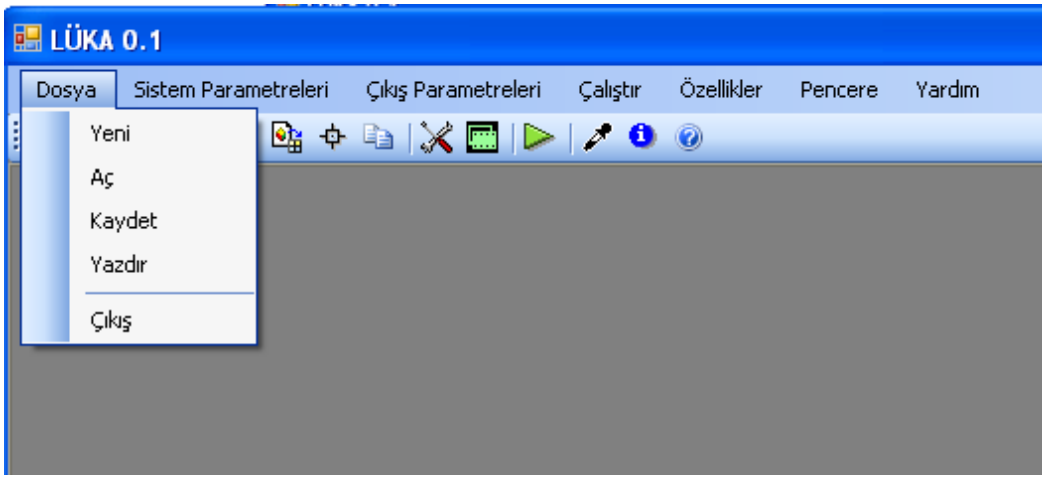
5.2 Yazılımın Kullanımı

Yazılım ilk çalıştırıldığında gelen ana ekran görüntüsü Şekil 5.4'de gösterilmiştir.

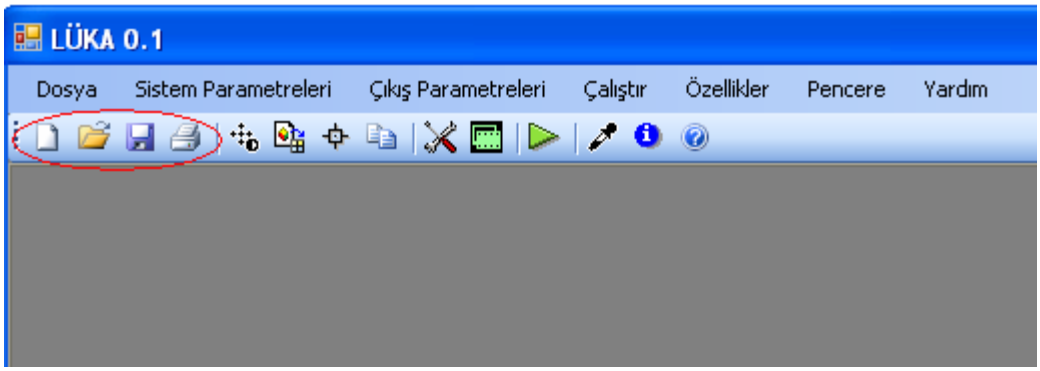


Şekil 5.4 Yazılım ana ekran görüntüsü

Programın Dosya menüsü altında standart bir programda olan yeni bir proje başlatılması, var olan bir projenin açılması, üzerinde analiz yapılan bir projenin kaydedilmesi ve yazdırma işlemleri için menüler hazırlanmıştır (Şekil 5.5 gösterildiği gibi). Ayrıca bu menülere ait kısayollar hızlı erişim kısmında (Şekil 5.6 gösterildiği gibi) oluşturulmuştur.

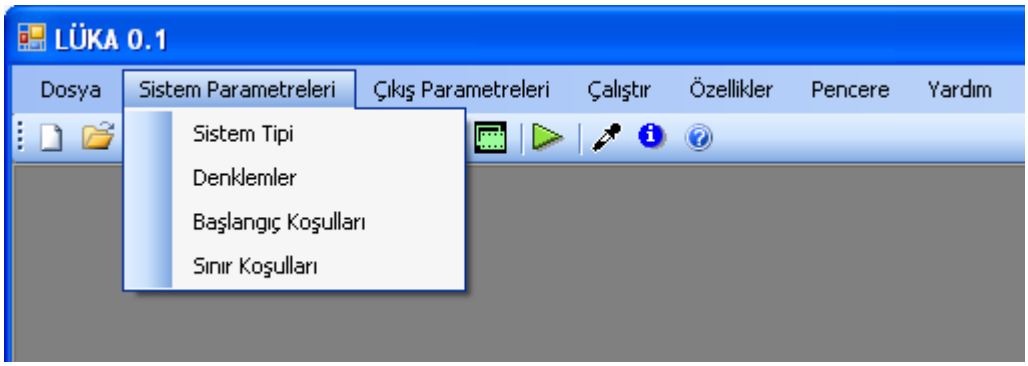


Şekil 5.5 Dosya menüsü

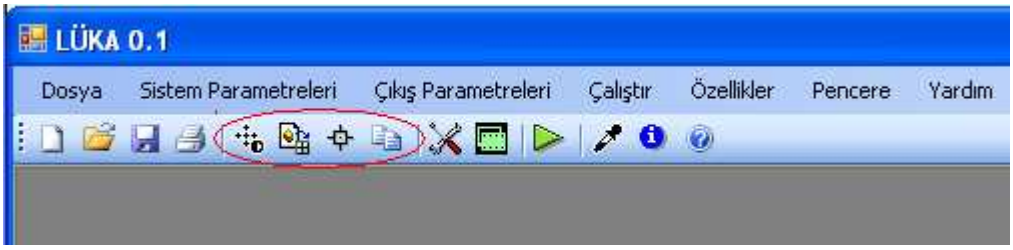


Şekil 5.6 Dosya menüsüne ait kısayollar

Sistem parametreleri menüsü; sistemin tipinin, denklemlerinin, başlangıç koşullarının ve Lyapunov üstelleri hesaplanırken kullanılan sınır koşullarının girilmesi için hazırlanan pencerelere erişim sağlamaktadır (Şekil 5.7'de gösterildiği gibi). Ayrıca bu menülere ait kısayollar hızlı erişim kısmında (Şekil 5.8'de gösterildiği gibi) oluşturulmuştur.

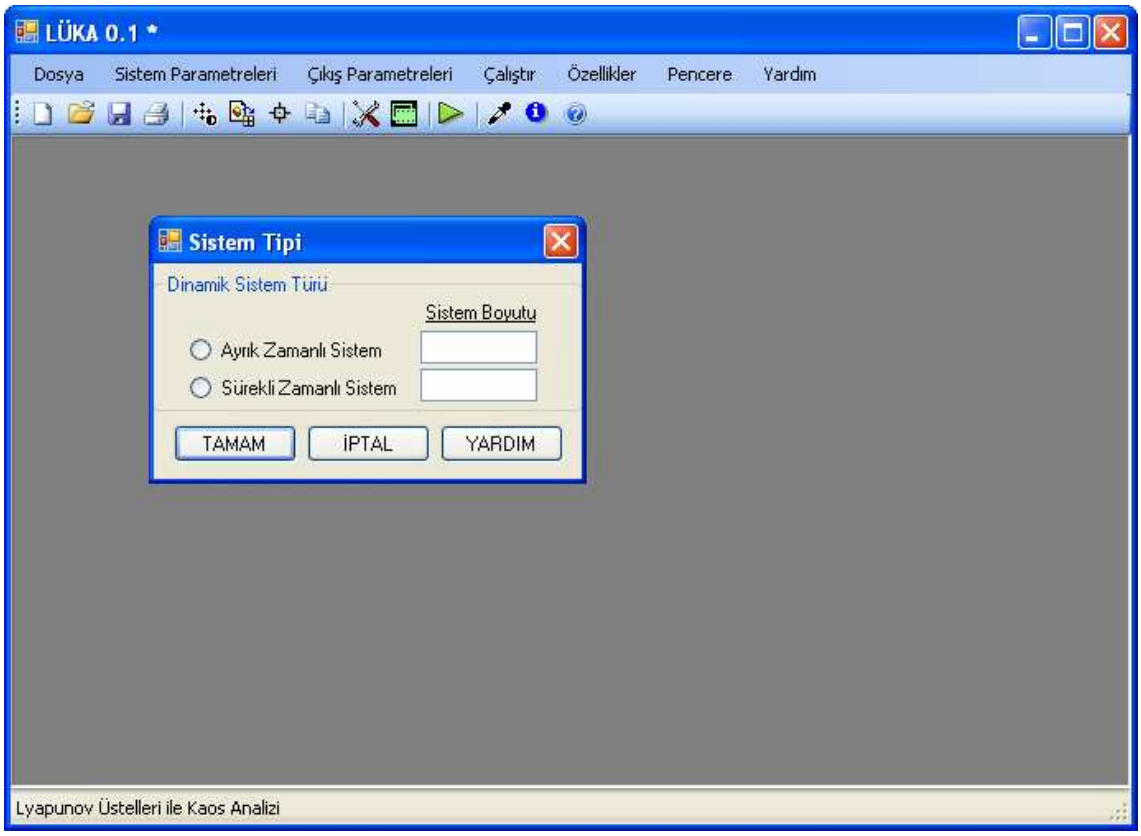


Şekil 5.7 Sistem parametreleri menüsü



Şekil 5.8 Sistem parametreleri menüsüne ait kısayollar

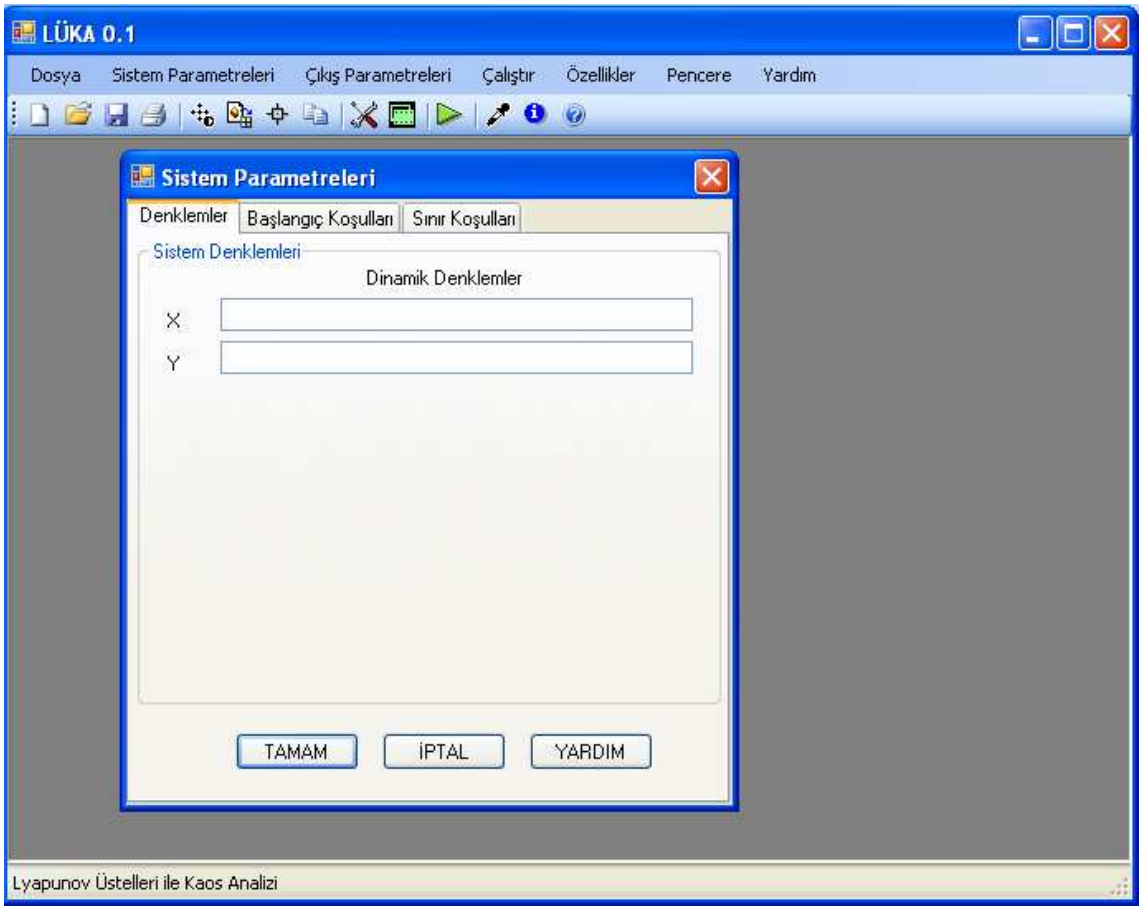
Sistemin tipinin belirlenmesi için hazırlanan pencereye şekil 5.9'da gösterilmiştir. Sisteminiz ayrık zamanlı ise ayrık zamanlı sistem seçeneğini, sürekli zamanlı bir sistem ise sürekli zamanlı sistem seçeneğini seçerek sistemin tipini belirleyebilirsiniz. Sistem tipi belirlenmeden yazılımın diğer menüleri kullanılmak istenildiğinde önce sistem tipinin girilmesi gerektiğini belirten bir uyarı mesajı alınır. Ayrıca yardım butonunu kullanarak ilgili yardım menüsünü aktif hale getirilebilir.



Şekil 5.9 Sistem tipinin girildiği pencere

.Sistem tipi belirlendikten sonra sisteme ait denklem girişleri yapılmalıdır. Örnek olarak iki boyutlu ayrık zamanlı bir sistemin denklem girişin yapıldığı ekran görüntüsü Şekil 5.10'daki gibidir. Denklemler girilmediğinde veya bir problem oluştuğunda yazılım uyarı mesajı

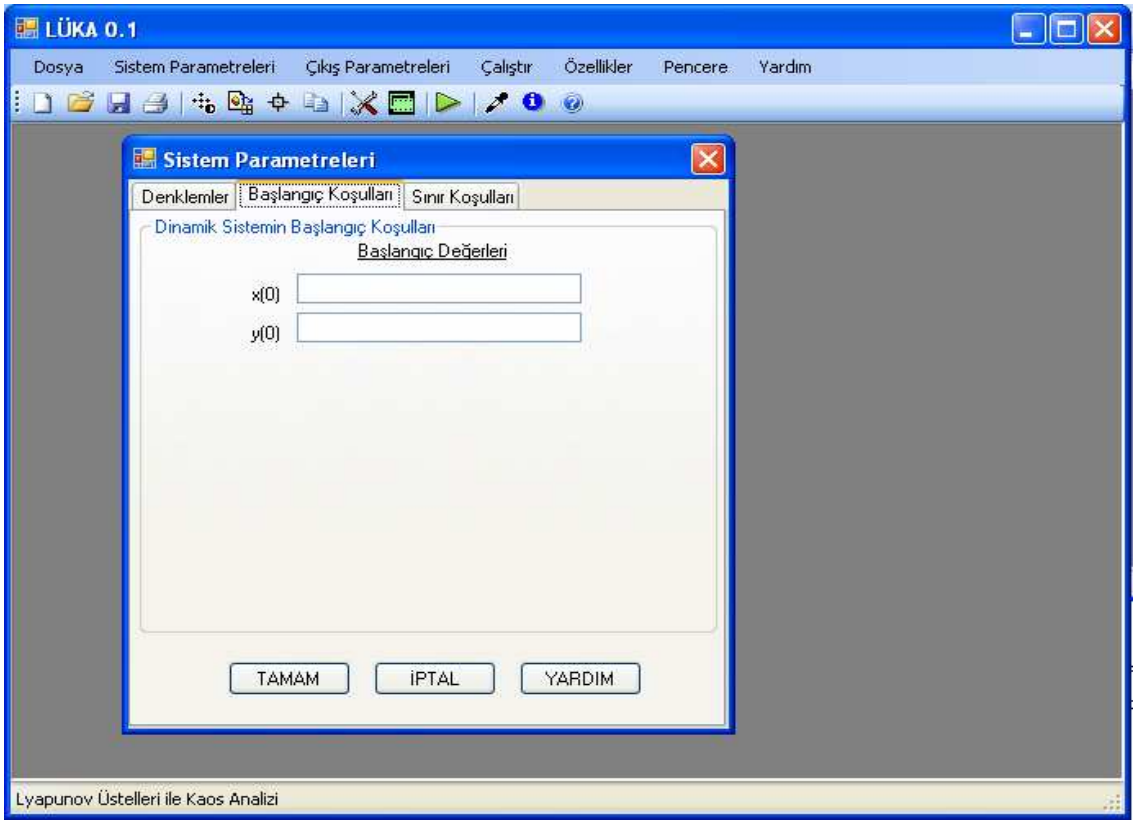
vermektedir. Yine bu pencerede bulunan yardım butonu kullanılarak ilgili yardım menüsünü aktif hale getirilebilir.



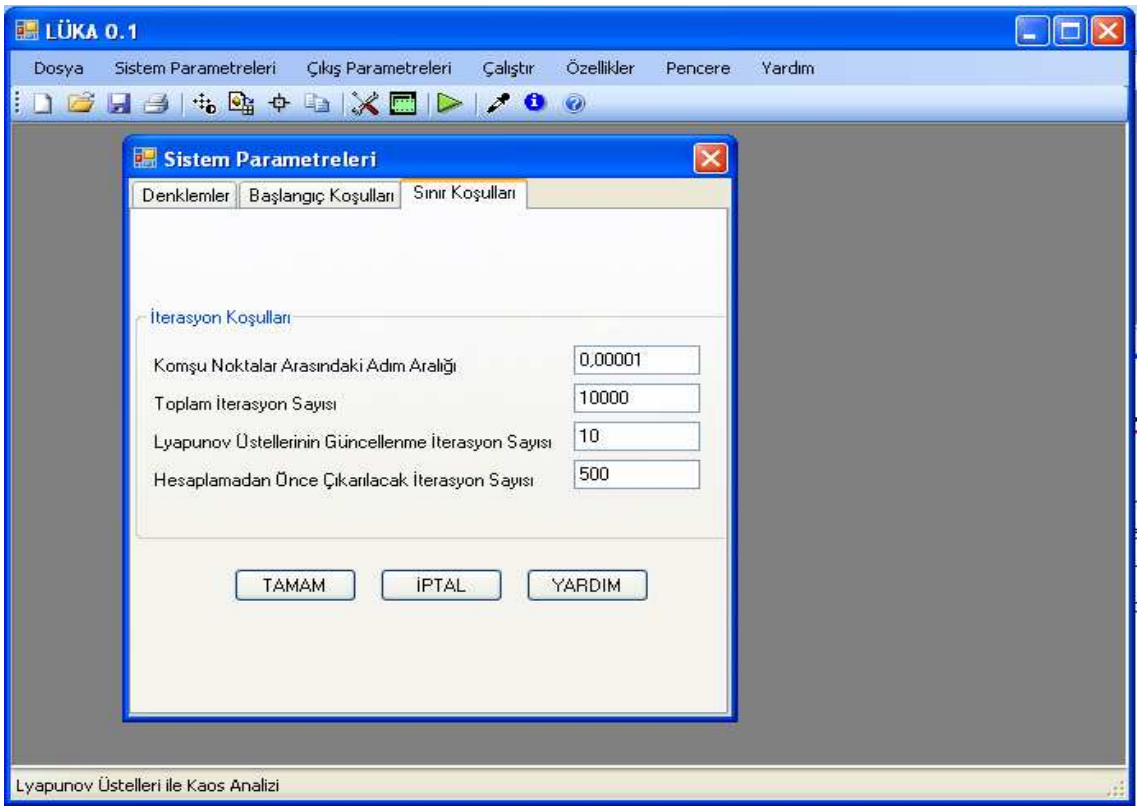
Şekil 5.10 Denklem girişlerinin yapıldığı pencere

Sisteme ait başlangıç koşullarının girildiği pencerenin ekran görüntüsü ise Şekil 5.11'de verilmiştir. Başlangıç değerleri girilmediğinde veya bir problem oluştuğunda yazılım uyarı mesajı vermektedir. Yine bu pencerede bulunan yardım butonu kullanılarak ilgili yardım menüsünü aktif hale getirilebilir.

Sistem parametreleri altındaki son menü ise sınır koşullarıdır. Bu pencerede Lyapunov üstelleri hesaplanırken kullanılacak çeşitli parametrelerin değerleri girilebilmektedir (şekil 5.12'de gösterildiği gibi). Bu parametreler; komşu noktalar arasındaki fark, yazılımın çalışacağı toplam adım sayısı, Lyapunov üstellerinin kaç adımda bir güncelleneceği ve hesaplamalara geçilmeden önce çıkarılacak adım sayısıdır. Bu değerler yazılım tarafından; varsayılan değerleri girilmiştir. Kullanıcı istediği takdirde bu değerleri değiştirebilir. Yine bu pencerede bulunan yardım butonu kullanılarak ilgili yardım menüsünü aktif hale getirilebilir.

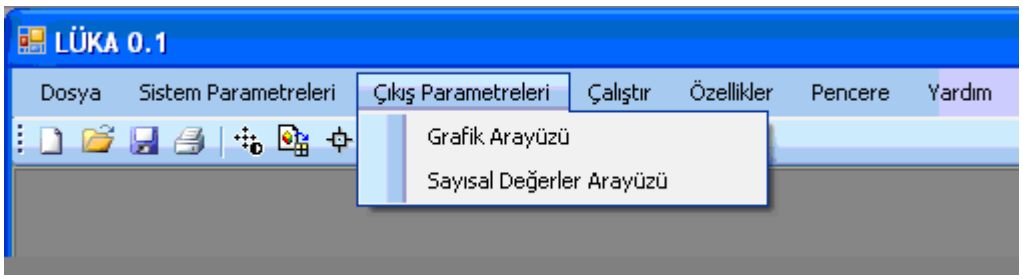


Şekil 5.11 Başlangıç koşullarının girildiği pencere

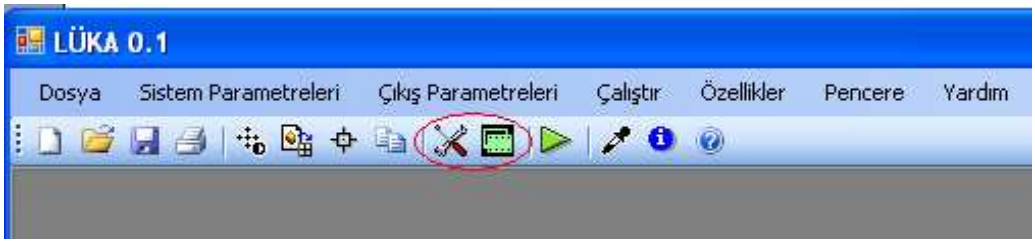


Şekil 5.12 Sınır koşullarının girildiği pencere

Çıkış parametreleri menüsü (Şekil 5.13'deki gibi) sistemin hesaplanan Lyapunov üstellerini göstermek için kullanılan grafik arayüzü ve sayısal değerler arayüzü ile ilgili parametrelerin girildiği kısımdır. Ayrıca bu menülere ait kısayollar hızlı erişim kısmında (Şekil 5.14'deki gibi) oluşturulmuştur.



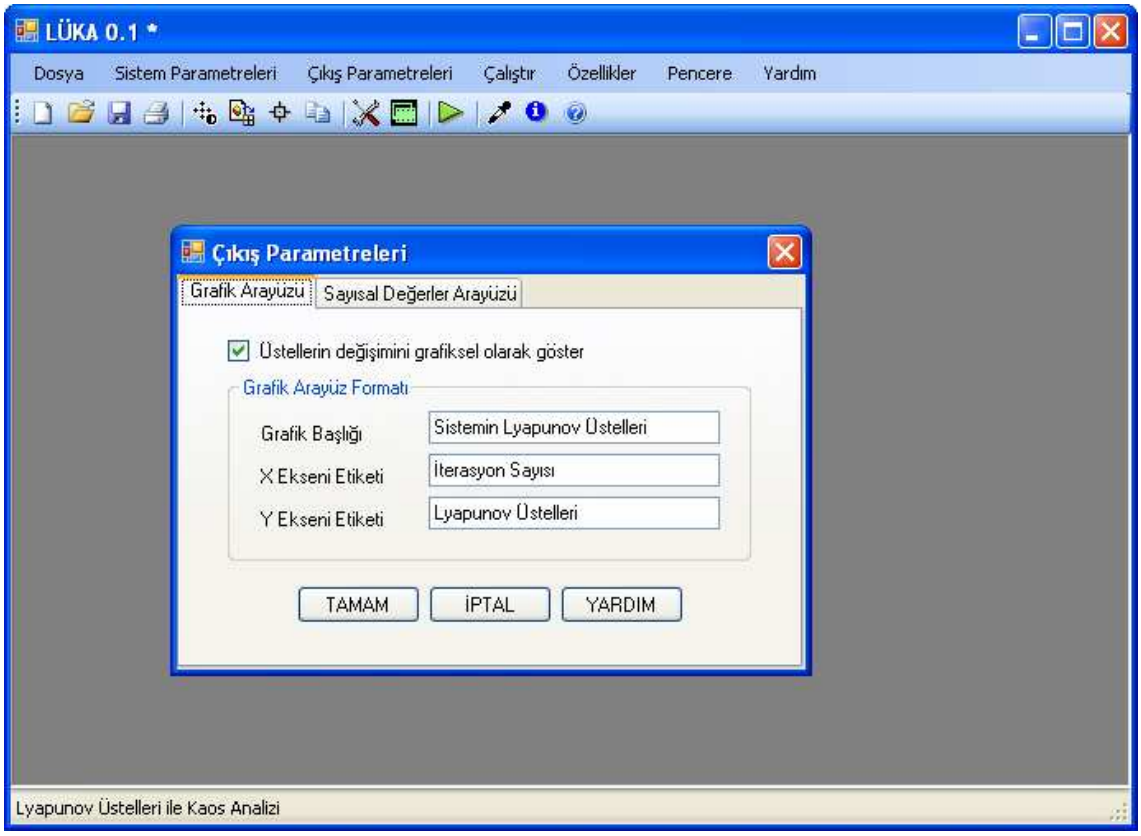
Şekil 5.13 Çıkış parametreleri menüsü



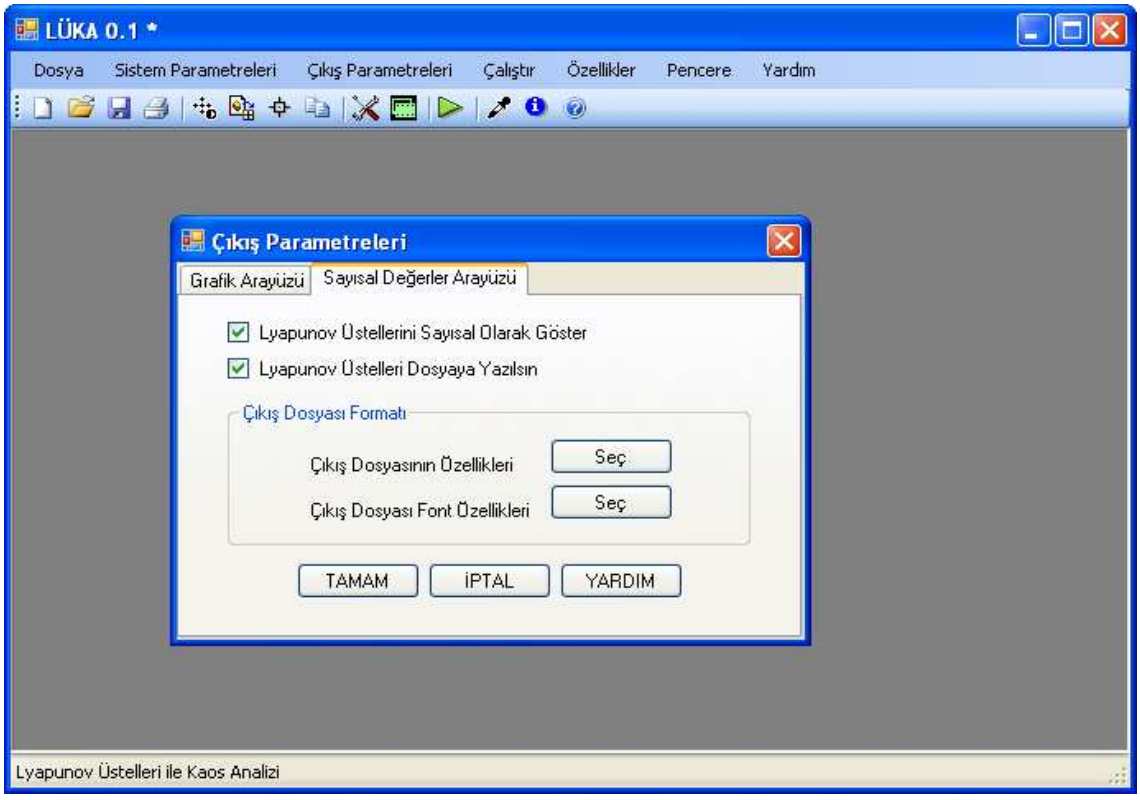
Şekil 5.14 Çıkış parametreleri menüsüne ait kısayollar

Grafik arayüzüne ait ekran görüntüsü Şekil 5.15'de gösterilmiştir. Grafik arayüzünde üstellerin çizilecek grafiğinde x-ekseninin etiketi, y-ekseninin etiketi ve grafik başlığı özellikleri bulunmaktadır. Yazılım tarafından varsayılan değerler bu pencerede girilmiştir. İstenildiği takdirde kullanıcı tarafından bu değerler değiştirilebilir.

Sayısal değerler arayüzüne ait ekran görüntüsü Şekil 5.16'da verilmiştir. Sayısal değerler arayüzünde hesaplanan Lyapunov üstellerinin sayısal değerlerinin gösterilip gösterilmeyeceği ve belirtilen dosyaya yazılıp yazılmayacağı ile ilgili parametrelerin girildiği penceredir. Yazılım tarafından varsayılan değerler bu pencerede girilmiştir. İstenildiği takdirde kullanıcı tarafından bu değerler değiştirilebilir.

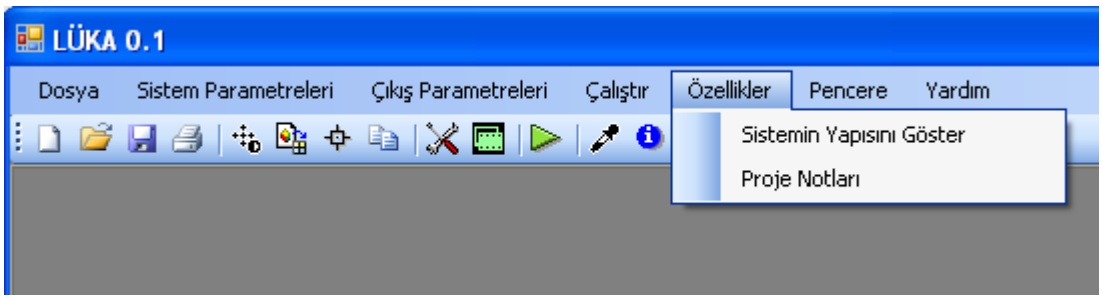


Şekil 5.15 Grafik arayüzü penceresi

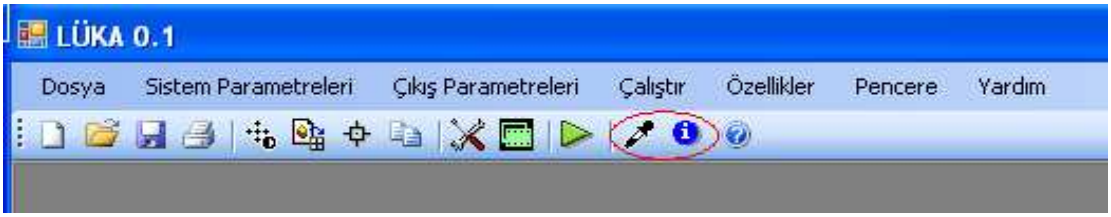


Şekil 5.16 Sayısal değerler arayüzü penceresi

Şekil 5.17‘de özellikler menüsü gösterilmiştir. Özellikler menüsü altında kullanıcının girdiği sistemin yapısını gösteren menü ve yazılımın analizi sırasında kullanıcının notlarını girebileceği bir menü bulunmaktadır. Ayrıca bu menülere ait kısayollar hızlı erişim kısmında (Şekil 5.18’deki gibi) oluşturulmuştur.



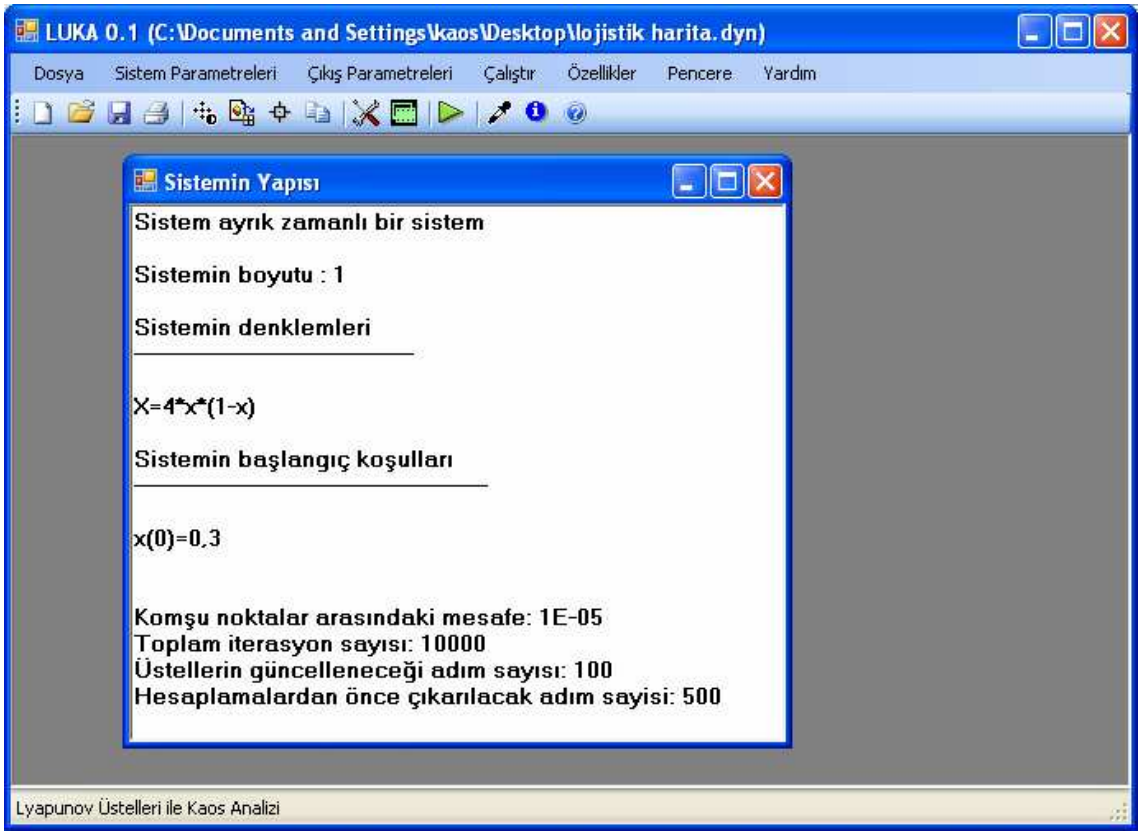
Şekil 5.17 Özellikler menüsü



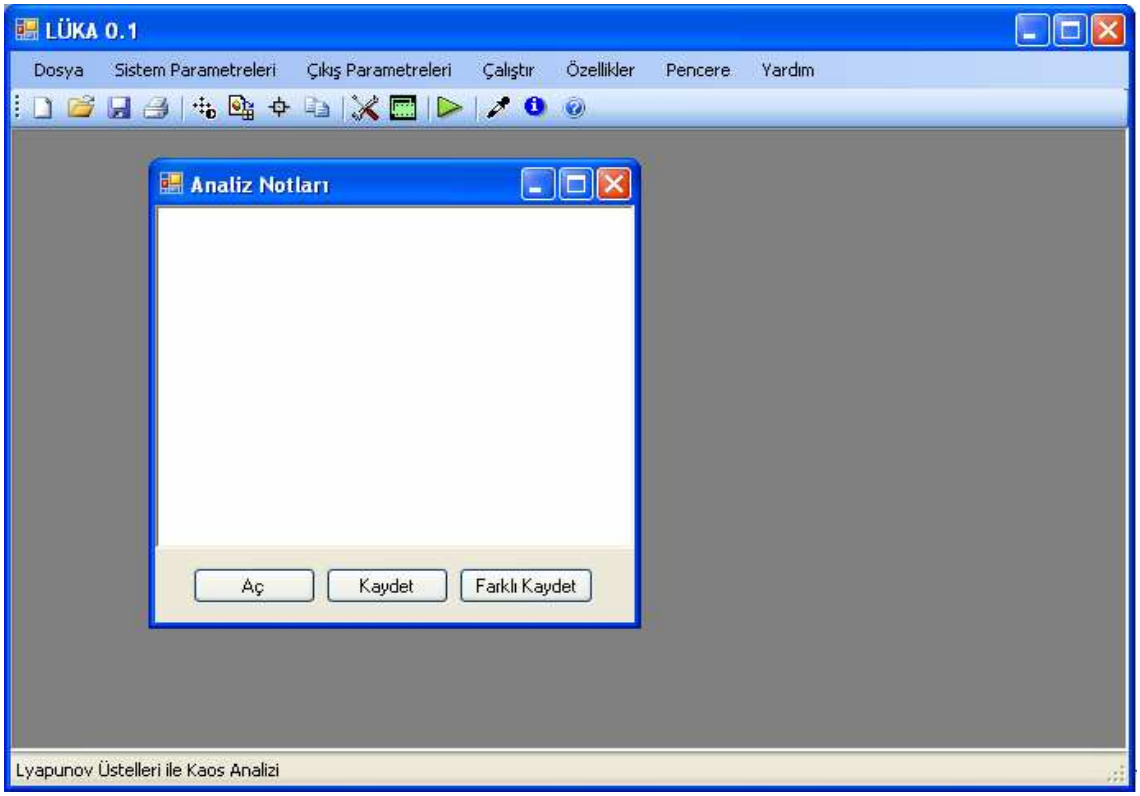
Şekil 5.18 Özellikler menüsüne ait kısayollar

Şekil 5.19‘da kullanıcı tarafından tipi, denklemeleri ve başlangıç koşulları girilen sistemin yapısını veren pencere gösterilmiştir. Bu pencere sistemin tüm özelliklerini aynı yerde görebilme imkânı sağlamaktadır.

Proje notları penceresinde ise kullanıcı sistem analizi sırasında kaydetmek istediği notları yazabildiği gibi yazılımın daha önceki kullanımlarında oluşturulmuş notları da görebilir. Bu pencereye ait ekran görüntüsü Şekil 5.20’de verilmiştir.



Şekil 5.19 Sistemin yapısı penceresi



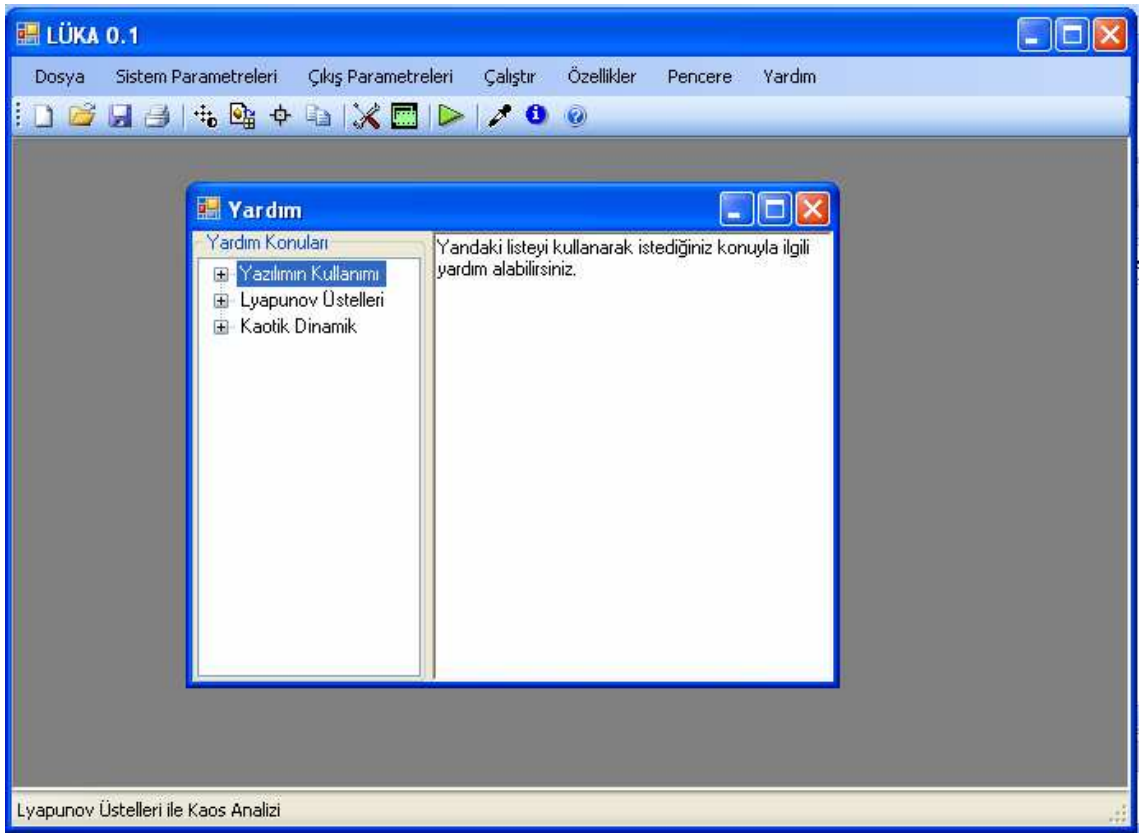
Şekil 5.20 Proje notları penceresi

Pencere menüsü altındaki denetimler kullanılarak aktif pencereler yatay, dikey veya basamaklı şekilde gösterilebilmesine yardımcı olabilir.

Yardım menüsü şekil 5.21'de gösterilmiştir. Bu menü altında yardım konuları, güncelleştirmeleri denetleyen menü ve program hakkındaki menü verilmiştir.

Şekil 5.21 Yardım penceresi

Yardım penceresinin ekran görüntüsü şekil 5.22'de verilmiştir. Yardım listesi kullanılarak ilgili konu hakkında yardım alınabilir.

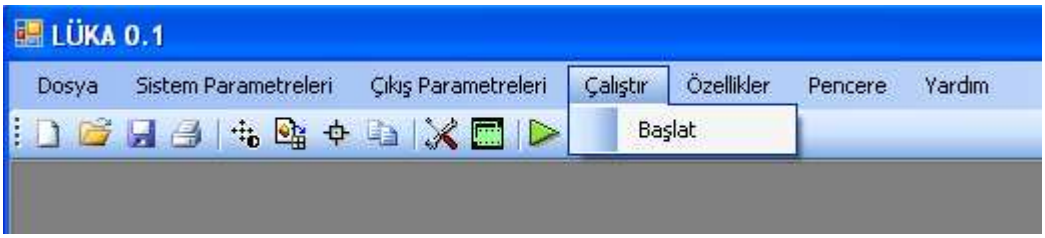


Şekil 5.22 Yardım penceresi

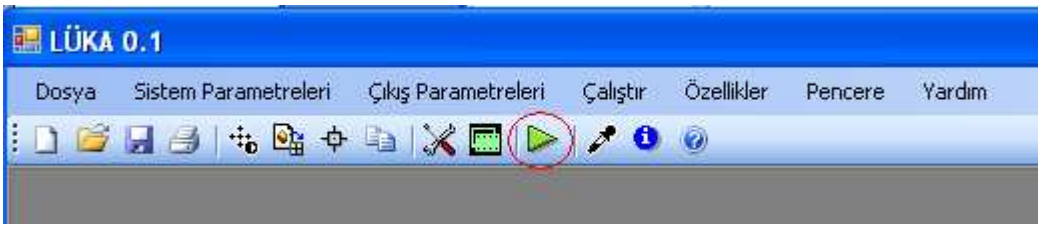
Güncelleştirmelerin denetlendiği pencerenin ekran görüntüsü Şekil 5.23'de verilmiştir.

Şekil 5.23 Güncelleştirmeler penceresi

Yazılımın ihtiyaç duyduğu parametreler girildikten sonra çalıştır menüsü içindeki başlat menüsü (Şekil 5.24'deki gibi) veya hızlı erişimdeki kısayolu kullanılarak (Şekil 5.25'deki gib) yazılım başlatılabilinir.



Şekil 5.24 Başlat menüsü



Şekil 5.25 Başlat menüsüne ait kısayol

6 YAZILIMIN TEST EDİLMESİ

Bu bölümde çeşitli sistemler üzerinde yazılımdan elde edilen sonuçlar gösterilmiştir. Öncelikle kaotik olmayan sistemlerde Lyapunov üstellerinin değişimi gösterilmiş daha sonra ayrık ve sürekli zamanlı çeşitli kaotik sistemler için Lyapunov üstelleri hesaplanmıştır.

6.1 Kaotik Olmayan Sistemde Lyapunov Üstellerinin Hesaplanması

Bir sistemde kaos analizi yapabilmek için sistemin sahip olması gereken bazı koşulların olduğu Bölüm 2.1.3'de gösterilmiştir. Bir sistemde kaos analizi yapabilmek için gerekli olan koşullar

- Sistemde doğrusal olmayan bir elemanın olması
- Sistem sürekli zamanlı bir sistem ise en az 3. dereceden bir sistem olmasıdır. Sistem ayrık zamanlı sistem ise böyle bir şart aranmaz. Ayrık zamanlı sistemler birinci dereceden dahi olsa kaos analizi yapılabilmektedir.

şeklindedir. Kaos analizi yapmak için yukarıda verilen şartlar gerek koşularıdır. Bu şartları sağlayan herhangi bir sistemde kaos analizi yapılabilir ancak mutlaka kaos gözlenebilir denmez. Kaos gözlenebilmesi için mutlaka sistem yörüngesinin başlangıç koşullarına bağımlı olması gerekir (yani en az bir tane pozitif Lyapunov üsteline sahip olmalıdır).

Bu kısımda ayrık zamanlı doğrusal bir sistem ve iki boyutlu sürekli zamanlı bir sistem kullanılarak Lyapunov üstellerinin değişimi hesaplamış iki sisteminde kaos analizi yapmak için gerek koşulları sağlamadığından Lyapunov üstellerinin negatif değerlere sahip olduğu gösterilmiştir.

6.1.1 Ayrık Zamanlı Doğrusal Bir Sistemin Lyapunov Üstelleri

Doğrusal sistemlerde kaos gözlenemeyeceği aşikardır. Sistemin sahip olduğu negatif Lyapunov üstelleri bunun bir göstergesidir. Burada bir boyutlu ayrık zamanlı doğrusal bir sistem kullanılmıştır. Sistemin yapısı denklem (6.1)'de verilmiştir.

$$x_{n+1} = 0.2 * x_n \tag{6.1}$$

Sistemin başlangıç koşulu x_0=0.6 şeklinde belirlenmiştir. Sistemin bu başlangıç değeri için hesaplanan Lyapunov üstelinin değeri ise λ=-0.138894491843064 şeklindedir. Sistem negatif bir üstele sahip olduğu için kaos gözlenmemektedir. Sonuç olarak doğrusal sistemlerde kaos aranmaz.

6.1.2 İki Boyutlu Sürekli Zamanlı Bir Sistemin Lyapunov Üstelleri

Sürekli zamanlı bir sistemde kaos analizi yapılabilmesi için sistemin en az üçüncü dereceden bir sistem olması gerektiği bilinmektedir. Bu nedenle iki boyutlu bir sistemde kaos gözlenemeyeceği aşikardır. Sistemin hesaplanan Lyapunov üstellerinin değerinin negatif olması da bunu desteklemektedir. Sistemin yapısı denklem (6.2)'de verilmiştir.

$$\frac{dx}{dt} = \sin(x)$$
$$\frac{dy}{dx} = y \qquad (6.2)$$

Sistemin başlangıç koşulları x_0=1, y_0=1 şeklinde belirlenmiştir. Üstellerin hesaplanan değerleri λ_1=-7,90483215175878E-05 ve λ_2= -0,596096141068241 şeklindedir. Üstellerin değerlerinden de anlaşıldığı gibi sistem sürekli zamanlı sistemlerde kaos analizi yapmak için gerek koşullarını sağlamadığından (sürekli zamanlı sistemlerde kaos analizi için sistem en az üç boyutlu olmalı koşulu) negatif Lyapunov üstellerine sahiptir ve kaos gözlenemez.

6.2 Ayrık Zamanlı Sistemlerde Lyapunov Üstellerinin Hesaplanması

Bu bölümde yaygın olarak kullanılan çeşitli ayrık zamanlı kaotik sistemlerin Lyapunov üstelleri hesaplanarak kaos analizleri verilmiş ve grafiksel olarak değişimleri gösterilmiştir.

6.2.1 Lojistik Haritaya Ait Lyapunov Üstellleri

Lojistik haritaya ait dinamikler denklem (6.3)'de verilmiştir. Sistemin başlangıç koşulu x_0=0.3 olarak seçilmiştir. Hesaplamalar yapılırken kullanılan sınır parametrelerinin değerleri ise tablo 6.1'de gösterilmiştir.

$$x_{n+1} = a x_n (1 - x_n) \tag{6.3}$$

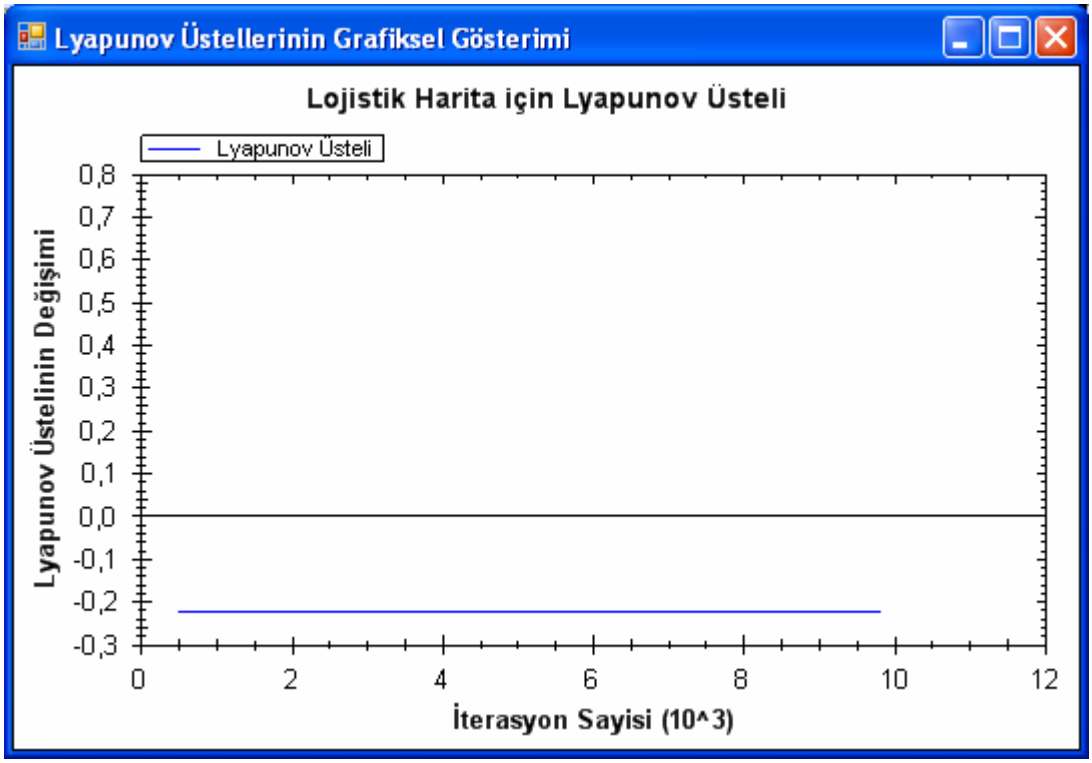
Tablo 6.1 Lojistik harita için sınır parametreleri

Sınır Parametresi	Değeri
Komşu Noktalar Arasındaki Adım Aralığı	0.00001
İterasyon Sayısı	10000
Üstellerin Güncelleneceği Adım Sayısı	10
Hesaplamalardan Önce Çıkarılacak İterasyon Sayısı	500

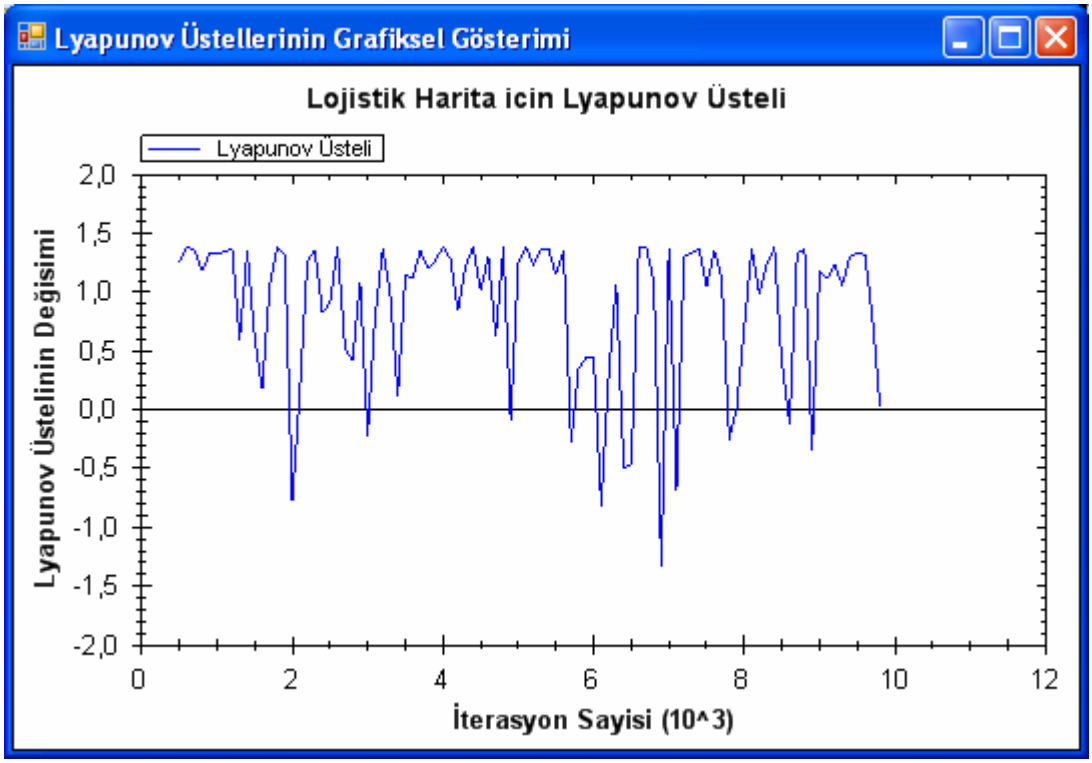
Şekil 6.1'de lojistik haritanın a=2.8 parametresi için Lyapunov üstelinin grafiksel değişimini göstermektedir. Sistemin a=2.8 parametresi için negatif bir Lyapunov üsteline sahip olması bu değerler için sistemde kaos gözlenmediğinin bir göstergesidir.

Şekil 6.2'de ise lojistik haritanın a=4 parametresi için Lyapunov üstelinin grafiksel değişimi gösterilmiştir. Bu parametre değeri için ise sistemin sahip olduğu pozitif Lyapunov üsteli sistemin kaotik bir davranış sergilediğini ortaya koymaktadır.

Tablo 6.2'de ise a parametresinin çeşitli değerleri için sistemin Lyapunov üstellerinin değerleri ve bu değerler için sistemin kaos analizi verilmiştir.



Şekil 6.1 Lojistik haritada a=2.8 parametresi için hesaplanan Lyapunov üsteli



Şekil 6.2 Lojistik haritada a=4 parametresi için hesaplanan Lyapunov üsteli

Tablo 6.2 Lojistik harita için Lyapunov üsteli

a	**Lyapunov Üsteli**	**Kaos Analizi**
2	-0,93374	Kaos yok
2.8	-0.01928	Kaos yok
3.5	-0.75603	Kaos yok
4	0.05669	Kaos var

6.2.2 Karasel (Quadratic) Haritaya Ait Lyapunov Üstelleri

Karasel (quadratic) haritaya ait dinamikler denklem (6.4)'de verilmiştir.

$$x(n+1) = 1 - ax^2 \tag{6.4}$$

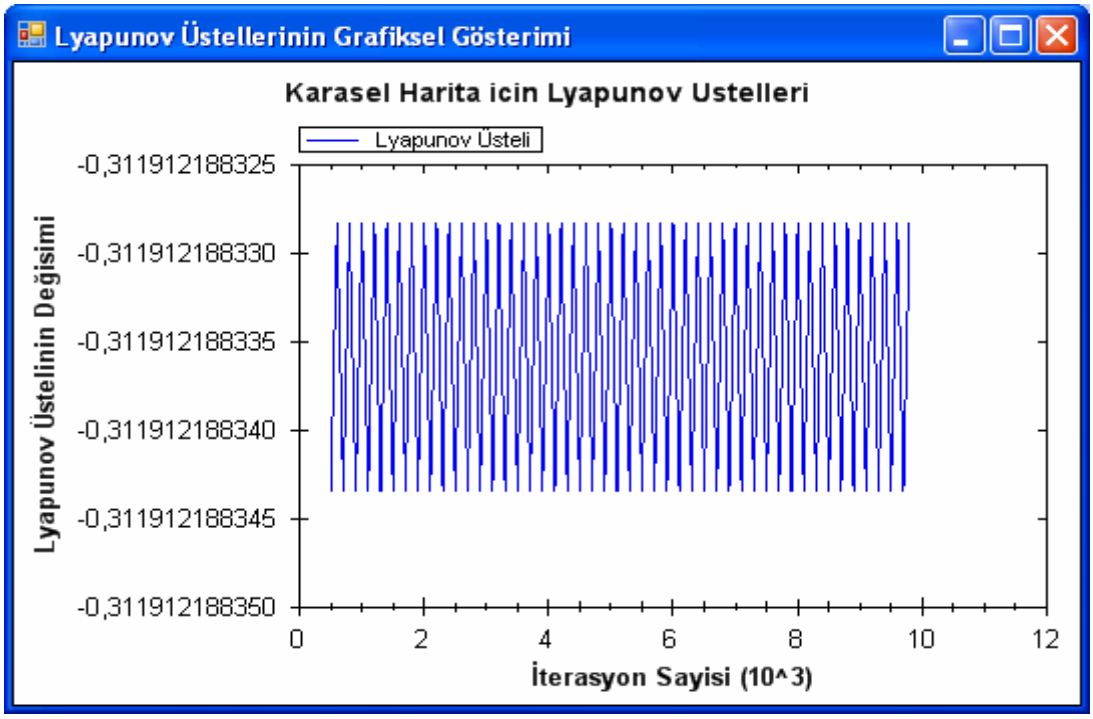
Sistemin başlangıç koşulu x_0=0.4 olarak belirlenmiştir. Lyapunov üstelleri hesaplanırken kullanılan sınır koşulları tablo 6.3'de verilmiştir. Sistem a=0.5 parametresi için

negatif bir Lyapunov üsteline sahiptir. Yani bu değer için sistemde kaos gözlenmez. Bu parametre değerleri için Lyapunov üstelinin grafiksel değişimi şekil 6.3'de gösterilmiştir. Sistem a=2 parametresi için ise pozitif bir Lyapunov üsteline sahiptir. Yani bu parametre değeri için sistem kaos içermektedir. Sistemin kaos içerdiği durumda hesaplanan Lyapunov üstelinin değişimi de şekil 6.4'de gösterilmiştir.

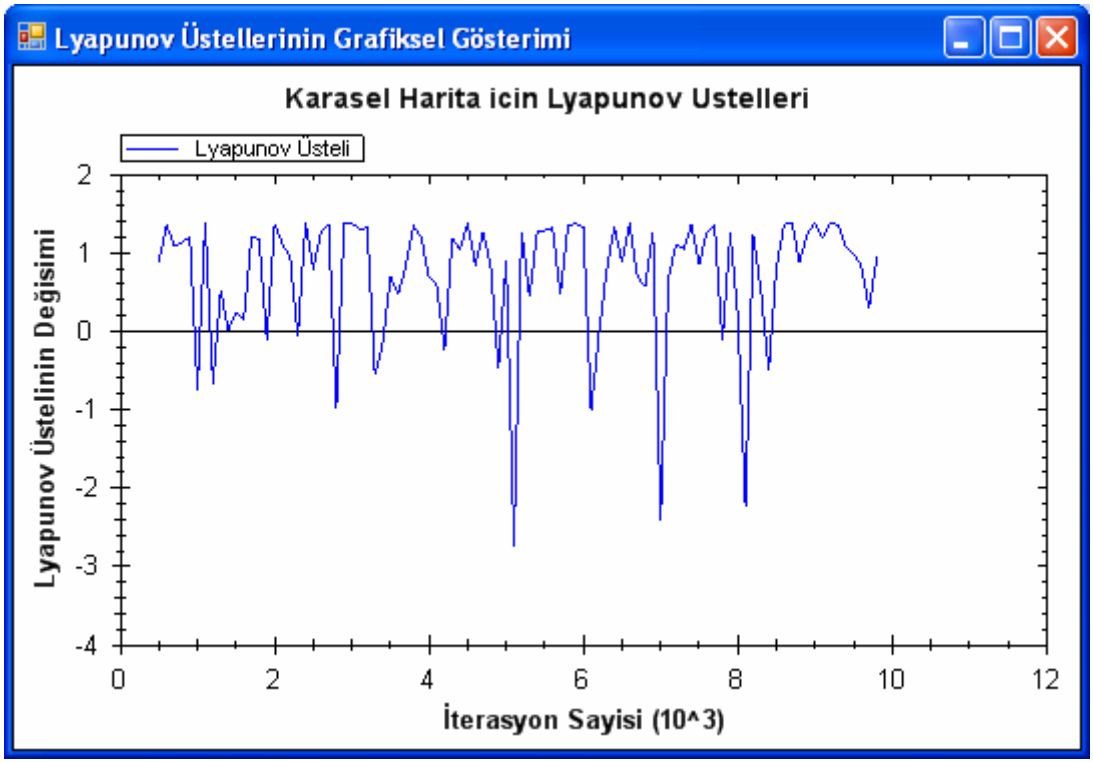
Tablo 6.3 Karasel harita için sınır parametreleri

Sınır Parametresi	Değeri
Komşu Noktalar Arasındaki Adım Aralığı	0.00001
İterasyon Sayısı	10000
Üstellerin Güncelleneceği Adım Sayısı	10
Hesaplamalardan Önce Çıkarılacak İterasyon Sayısı	500

Sistemin a parametresinin çeşitli değerlerine göre hesaplanmış Lyapunov üstelleri ve kaos analizleri tablo 6.4'de verilmiştir.



Şekil 6.3 Karasel haritada a=0.5 parametresi için Lyapunov üsteli



Şekil 6.4 Karasel haritada a=2 parametresi için Lyapunov üsteli

Tablo 6.4 Karasel harita için Lyapunov üsteli

A	**Lyapunov Üsteli**	**Kaos Analizi**
0,5	-0,02691	Kaos yok
0,9	-0,03958	Kaos yok
1,3	-0,03694	Kaos yok
2	0,06225	Kaos var

6.2.3 Henon Haritaya Ait Lyapunov Üstelleri

Henon haritaya ait dinamikler denklem (6.5)'de verilmiştir.

$$x_{n+1} = 1 - ax_n^{\,2} + y_n \qquad (6.5)$$

$$y_{n+1} = bx_n$$

Başlangıç koşulları x_0=0 ve y_0=0 olmak üzere a ve b parametrelerinin değişimine göre sistemin Lyapunov üstelleri ve gösterdiği davranışlar aşağıda verilmiştir. Lyapunov üstelleri hesaplanırken kullanılan sınır koşulları tablo 6.5'de verilmiştir.

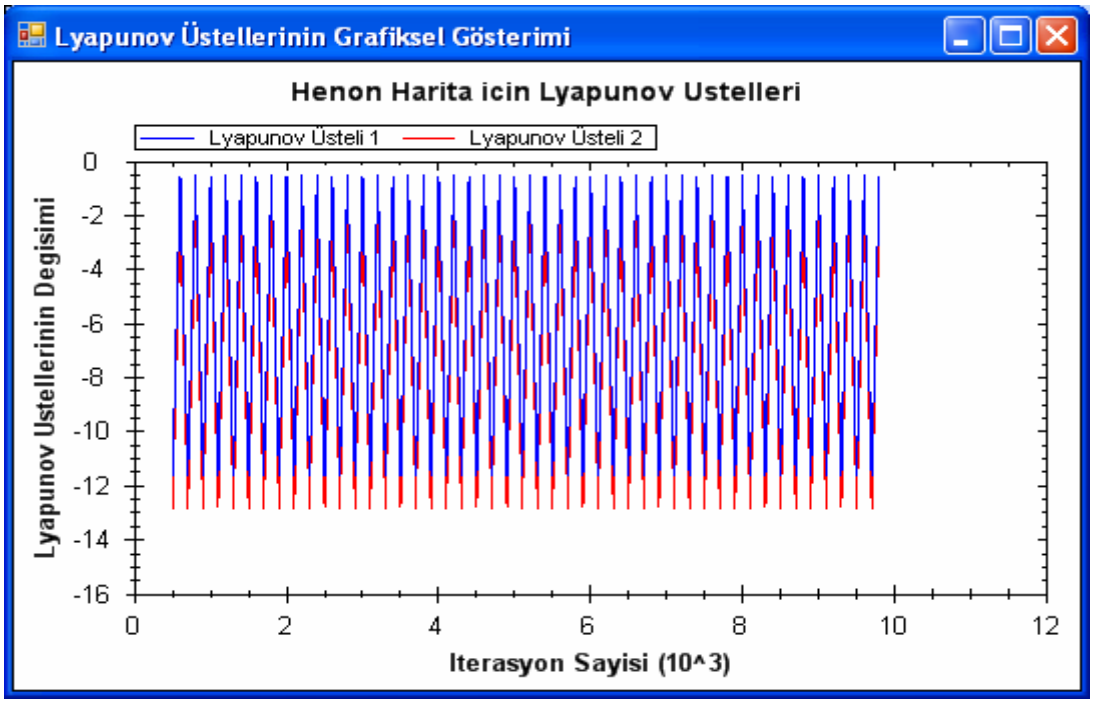
Sistemin $a = 0.9$ ve $b = 0.3$ parametreleri için Lyapunov üstellerinin değişimi Şekil 6.5'de verilmiştir. Sistemin sahip olduğu negatif iki Lyapunov üsteli bu parametre değerleri için kaos gözlenmediğinin bir göstergesidir.

Sistemin $a = 1.4$ ve $b = 0.3$ parametreleri kullanılarak elde edilen Lyapunov üstellerinden birinin pozitif değerli olarak hesaplanması sistemin bu parametre değerleri için kaos içerdiğini göstermektedir. Sistemin kaos halinde Lyapunov üstellerinin grafiksel değişimi de Şekil 6.6'da gösterilmiştir.

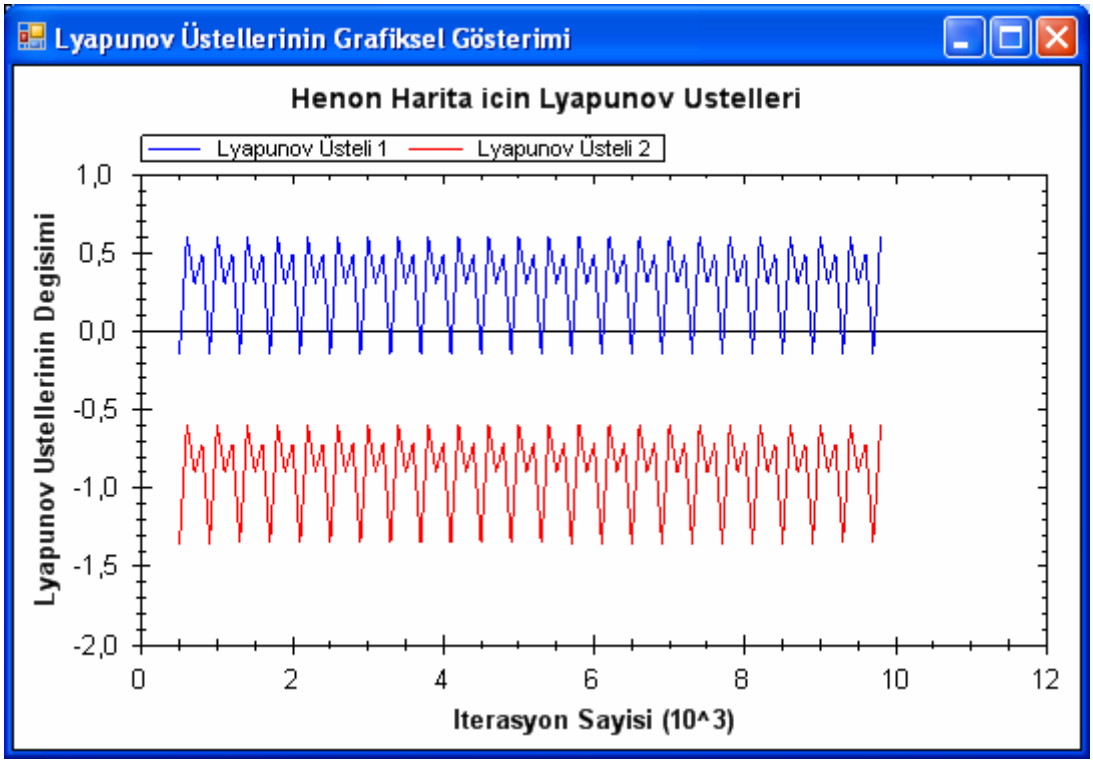
Tablo 6.6'da ise Henon haritada *a* ve *b* parametrelerinin değişik değerleri için hesaplanan Lyapunov üstellerinin değerleri ve kaos analizleri verilmiştir.

Tablo 6.5 Henon harita için sınır parametreleri

Sınır Parametresi	Değeri
Komşu Noktalar Arasındaki Adım Aralığı	0.00001
İterasyon Sayısı	10000
Üstellerin Güncelleneceği Adım Sayısı	100
Hesaplamalardan Önce Çıkarılacak İterasyon Sayısı	500



Şekil 6.5 Henon haritada a=0.9 ve b=0.3 parametreleri için hesaplanan Lyapunov üstelleri



Şekil 6.6 Henon haritada a=1.4 ve b=0.3 parametreleri için hesaplanan Lyapunov üstelleri

Tablo 6.6 Henon harita için Lyapunov üstelleri

a	***b***	**Lyapunov Üsteli -1-**	**Lyapunov Üsteli -2-**	**Kaos Analizi**
0.9	0.3	-0.52392	-0.62783	Kaos yok
1	0.3	-0.03958	-0.14360	Kaos yok
1.4	0.3	0.02707	-0.07682	Kaos var

6.2.4 Ikeda Haritaya Ait Lyapunov Üstelleri

Ikeda haritasına ait dinamikler denklem (6.6)'da verilmiştir.

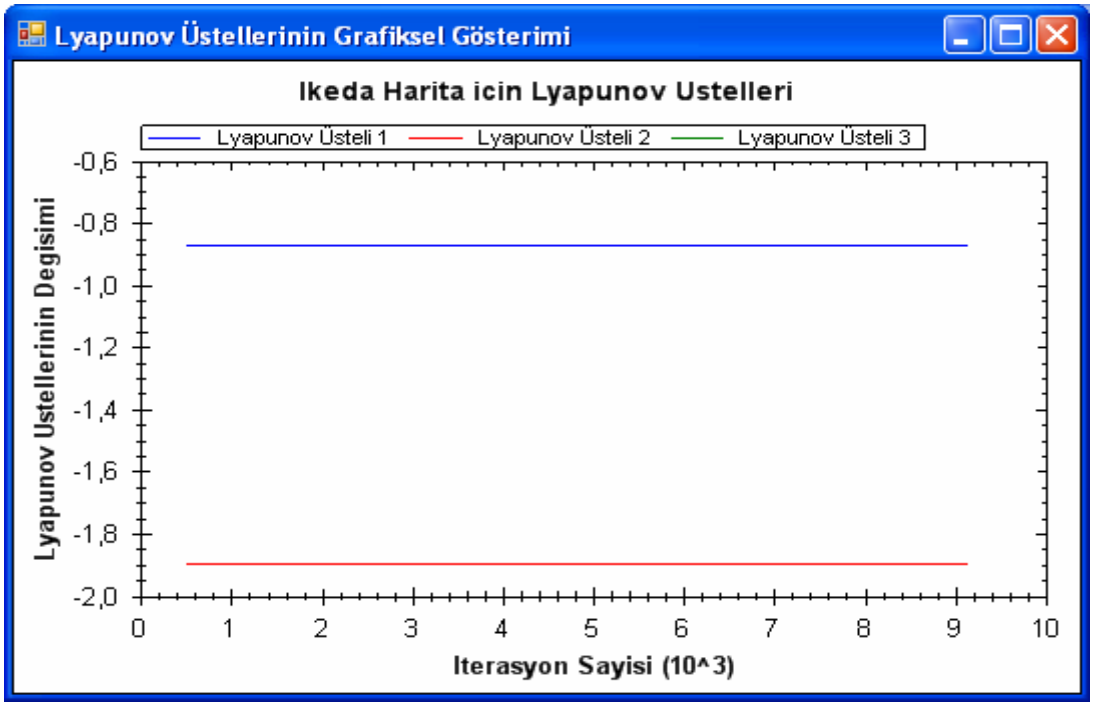
$$\begin{aligned} x(n+1) &= 1 + a(x\cos(z) - y\sin(z)) \\ y(n+1) &= a(x\sin(z) + y\cos(z)) \\ z(n+1) &= 1 \end{aligned} \qquad (6.6)$$

Sistemin Lyapunov üstelleri hesaplanırken x(0)=1, y(0)=1 ve z(0)=1 başlangıç koşulları kullanılmış; sınır parametreleri ise tablo 6.7'deki gibi belirlenmiştir. Ikeda haritanın a=0,3 parametresi için Lyapunov üstellerinin değişimi şekil 6.7'de gösterilmiştir. Ikeda haritanın bu parametre değeri için hesaplanan üstellerinden görülmektedir ki sistem özerk olmayan bir sistem iken özerk biçime dönüştürülmüştür. Çünkü Lyapunov üstellerinden biri hep sıfır çıkmaktadır. Diğer üstellerinin değerlerinin negatif olması bu parametre değerleri için sistemde kaos gözlenmediğini göstermektedir. Haritanın a=0,7 parametresi için pozitif değerli bir Lyapunov üsteline sahip olduğundan a=0,7 parametresi için Ikeda haritasında kaos gözlenmektedir. Bu parametre için üstellerin değişimi ise şekil 6.8'de verilmiştir.

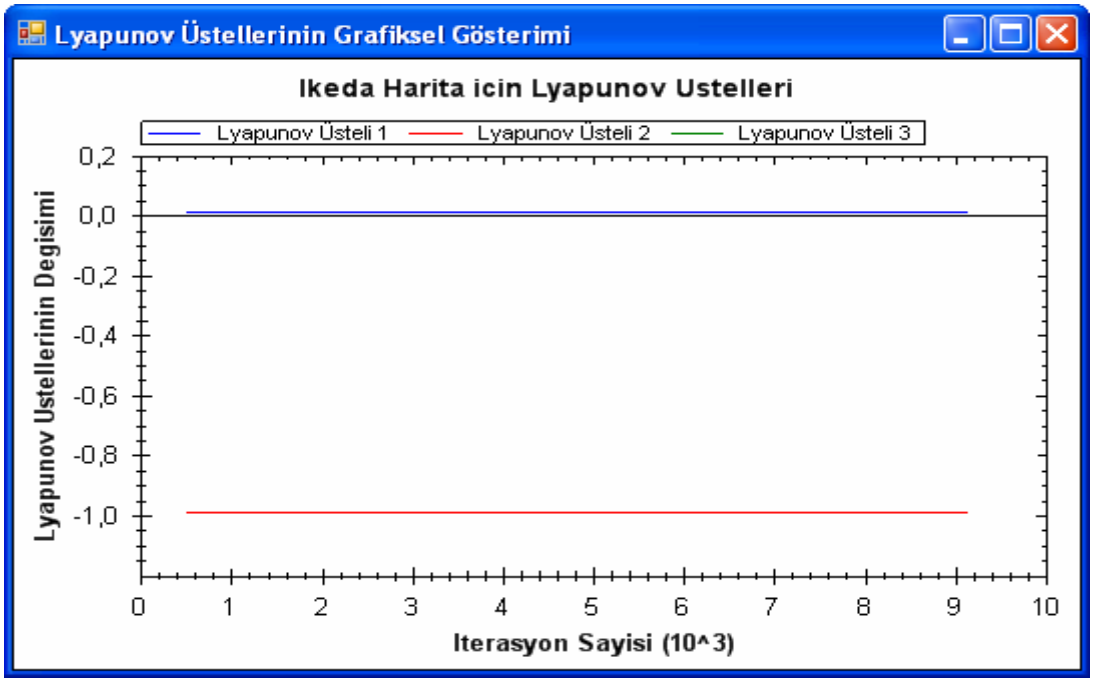
Tablo 6.7 Ikeda harita için sınır parametreleri

Sınır Parametresi	Değeri
Komşu Noktalar Arasındaki Adım Aralığı	0.00001
İterasyon Sayısı	10000
Üstellerin Güncelleneceği Adım Sayısı	10
Hesaplamalardan Önce Çıkarılacak İterasyon Sayısı	500

Tablo 6.8'de ise a parametresinin farklı değerleri için Ikeda haritasından elde edilen Lyapunov üstelleri ve kaos analizi verilmiştir.



Şekil 6.7 Ikeda haritada a=0,3 parametresi için Lyapunov üstellerinin değişimi



Şekil 6.8 Ikeda haritada a=0,7 parametresi için Lyapunov üstellerinin değişimi

Tablo 6.8 Ikeda harita için Lyapunov üstelleri

a	**Lyapunov Üsteli -1-**	**Lyapunov Üsteli -2-**	**Lyapunov Üsteli -3-**	**Kaos Analizi**
0,3	-0,41208	-0,90046	0	Kaos yok
0,6	-0,05526	-0,82637	0	Kaos yok
0,7	0,00649	-0,47060	0	Kaos var

6.3 Sürekli Zamanlı Sistemlerde Lyapunov Üstellerinin Hesaplanması

Bu bölümde yaygın olarak kullanılan çeşitli sürekli zamanlı kaotik sistemlerin Lyapunov üstellerinin değerleri hesaplanarak kaos analizleri yapılmış ve grafiksel olarak değişimleri gösterilmiştir.

6.3.1 Lorenz Sistemine Ait Lyapunov Üstelleri

Lorenz sistemi üç boyutlu sürekli zamanlı bir kaotik sistemdir. Sisteme ait dinamikler denklem (6.7)'de verilmiştir.

$$\frac{dx}{dt} = a(y - x)$$

$$\frac{dy}{dt} = (bx - y - xz) \qquad (6.7)$$

$$\frac{dz}{dt} = (xy - cz)$$

$-20 \leq x \leq 20$, $-50 \leq y \leq 50$ ve $-50 \leq z \leq 50$ olmak koşuluyla, a Prandtl sayısını, b Reynold sayısını göstermektedir. c parametresi ise akışkanın genişliğidir. x yayılan sıvı akışkanının hızı ile orantılı bir değişkendir. y artan ve azalan akışkan elementlerinin arasındaki sıcaklık farkı ile orantılı bir değişkendir. z ise denge noktasındaki sıcaklığın değişimi ile orantılı bir değişkendir.

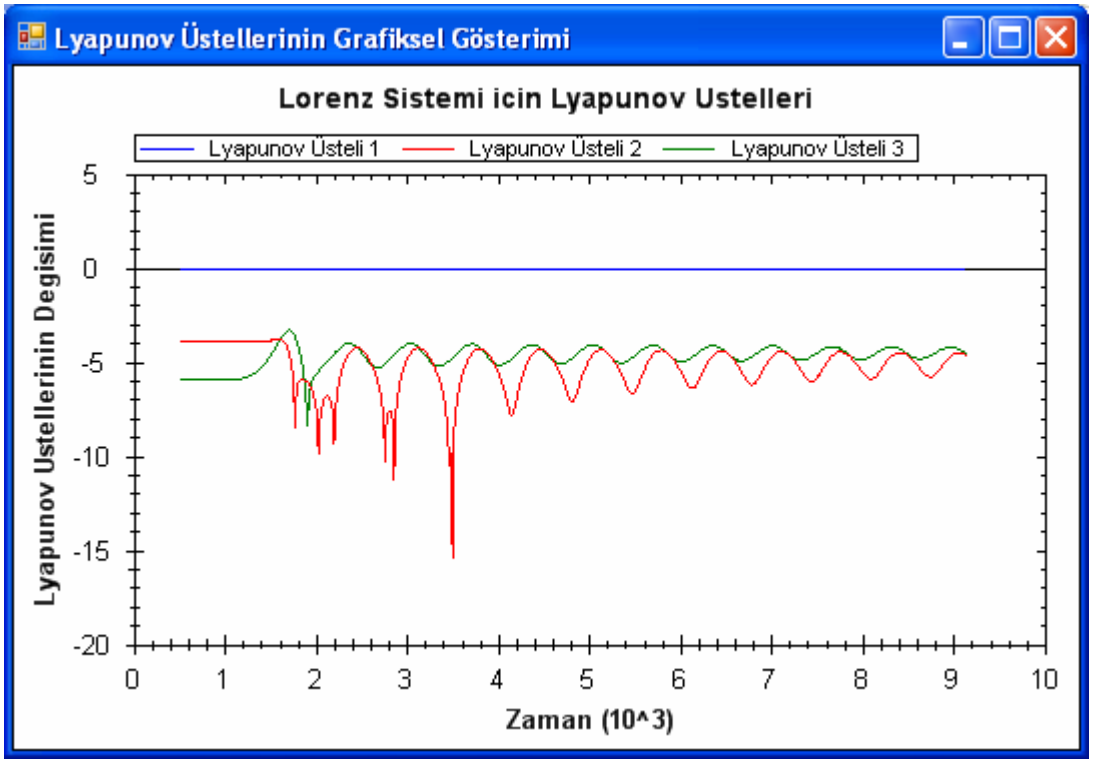
Başlangıç koşulları (x_0=0.1, y_0=0, z_0=0) olmak üzere sisteme ait Lyapunov üstelleri hesaplanırken kullanılan sistem parametreleri tablo 6.9'da verilmiştir. Sistem $a = 10$, $b = 21$ ve $c = 8/3$ değerleri için periyodik davranış göstermektedir. Sistemin periyodik davranış gösterdiği parametre değerleri için hesaplanan Lyapunov üstelleri şekil 6.9'da verilmiştir. Hesaplanan Lyapunov üstellerinin negatif değerli olması sistemin periyodik davranış gösterdiğinin kanıtıdır.

Sistem aynı başlangıç koşulları ve $a = 10$, $b = 28$ ve $c = 8/3$ parametre değerleri için kaotik davranış göstermektedir. bu değerler için hesaplanan Lyapunov üstelleri şekil 6.10'da gösterilmiştir. Lorenz sisteminin $a = 10$, $b = 28$ ve $c = 8/3$ değerleri için pozitif değerli bir Lyapunov üsteline sahip olması bu değerler için sistemin kaosa girdiğini göstermektedir.

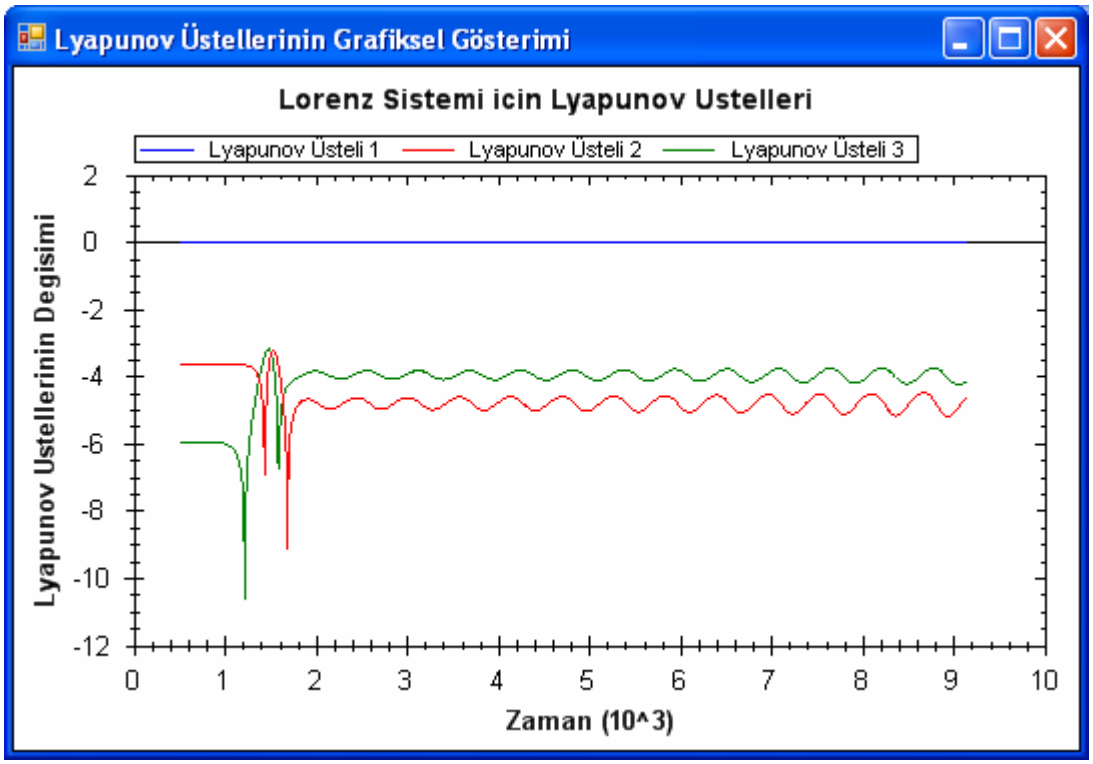
Tablo 6.9 Lorenz sistemi için sınır parametreleri

Sınır Parametresi	Değeri
Komşu Noktalar Arasındaki Adım Aralığı	0.00001
İterasyon Sayısı	10000
Üstellerin Güncelleneceği Adım Sayısı	10
Hesaplamalardan Önce Çıkarılacak İterasyon Sayısı	500

Tablo 6.10'da Lorenz sistemi için *a*, *b* ve *c* parametrelerinin değişik değerleri için hesaplanan Lyapunov üstellerinin değerleri ve kaos analizi verilmiştir.



Şekil 6.9 Lorenz sisteminde a=10, b=21 ve c=8/3 parametreleri için hesaplanan Lyapunov üstelleri



Şekil 6.10 Lorenz sisteminde a=10, b=28 ve c=8/3 parametreleri için hesaplanan Lyapunov üstelleri

Tablo 6.10 Lorenz sistemi için Lyapunov üstelleri

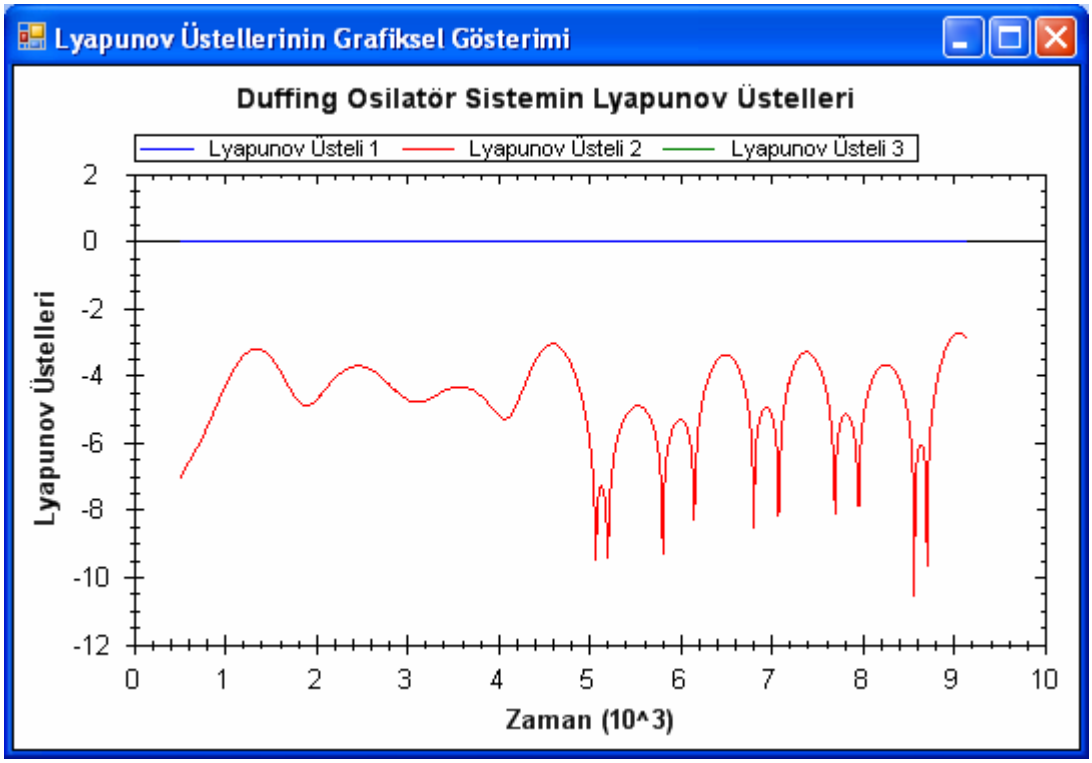
a	***b***	***c***	**Lyapunov Üsteli -1-**	**Lyapunov Üsteli -2-**	**Lyapunov Üsteli -3-**	**Kaos Analizi**
10	18	8/3	-9.196E-07	-0.45485	-0.41640	Kaos yok
10	21	8/3	-1.091E-06	-0.45098	-0.40392	Kaos yok
10	28	8/3	4.069E-06	-0.40304	-0.35874	Kaos var

6.3.2 Duffing Osilatöre Ait Lyapunov Üstelleri

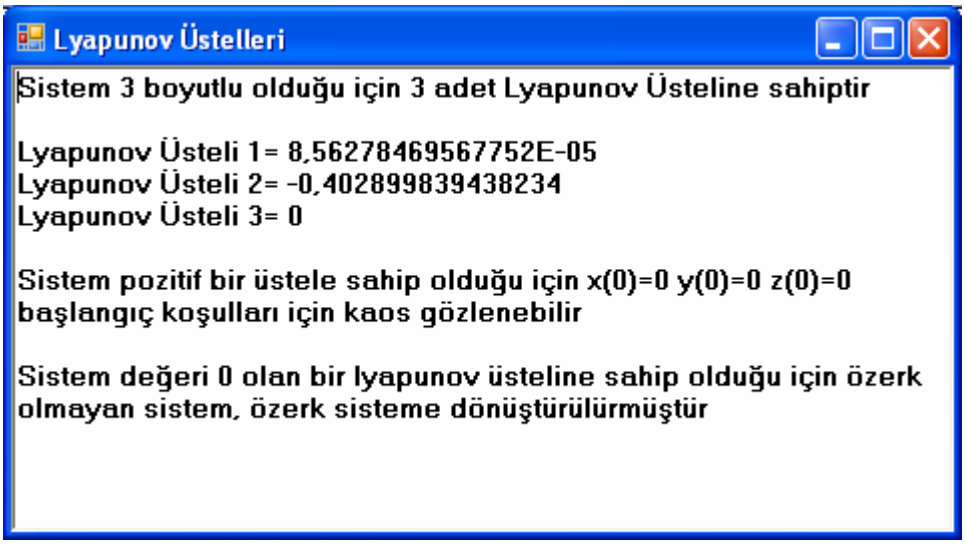
Duffing Osilatör, kaosu anlatan kaynaklarda sıklıkla rastlanan elektriksel kaotik bir sistemdir. Duffing osilatöre ait sistem dinamikleri denklem (6.8)'de verilmiştir.

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = -x - x^3 - ay + b\cos(wz) \qquad (6.8)$$
$$\frac{dz}{dt} = 1$$

$-5 \le x \le 5$, $-10 \le y \le 10$ ve $0 \le z$ olmak koşuluyla, a=0.2, b=36 ve w=0,661 iken sistem kaotik davranış göstermektedir. Osilatörün Lyapunov üstelleri hesaplanırken kullanılan başlangıç koşulları $x_0=0$, $y_0=0$ ve $z_0=0$ şeklinde belirlenmiş ve kaotik davranış gösterdiği parametre değerleri için Lyapunov üstellerinin değişimi şekil 6.11'de verilmiştir. Üstellerden birinin pozitif değerli olması sistemin bu değerler için kaosa girmiş olduğunun bir göstergesidir. Ayrıca osilatöre ait Lyapunov üstellerinin birinin değeri sıfırdır. Bunun anlamı sistem başlangıçta özerk olmayan bir sistemken özerk bir sisteme dönüştürülmüştür. Üstellerin değerleri ve kaos analizi şekil 6.12'de verilmiştir.



Şekil 6.11 Duffing osilatörde a=0.2, b=36 ve w=0,661 parametreleri için hesaplanan Lyapunov üstelleri



Şekil 6.12 Duffing osilatörü için hesaplanan Lyapunov üstelleri ve kaos analizi

6.3.3 Van Der Pol Denklemlerine Ait Lyapunov Üstelleri

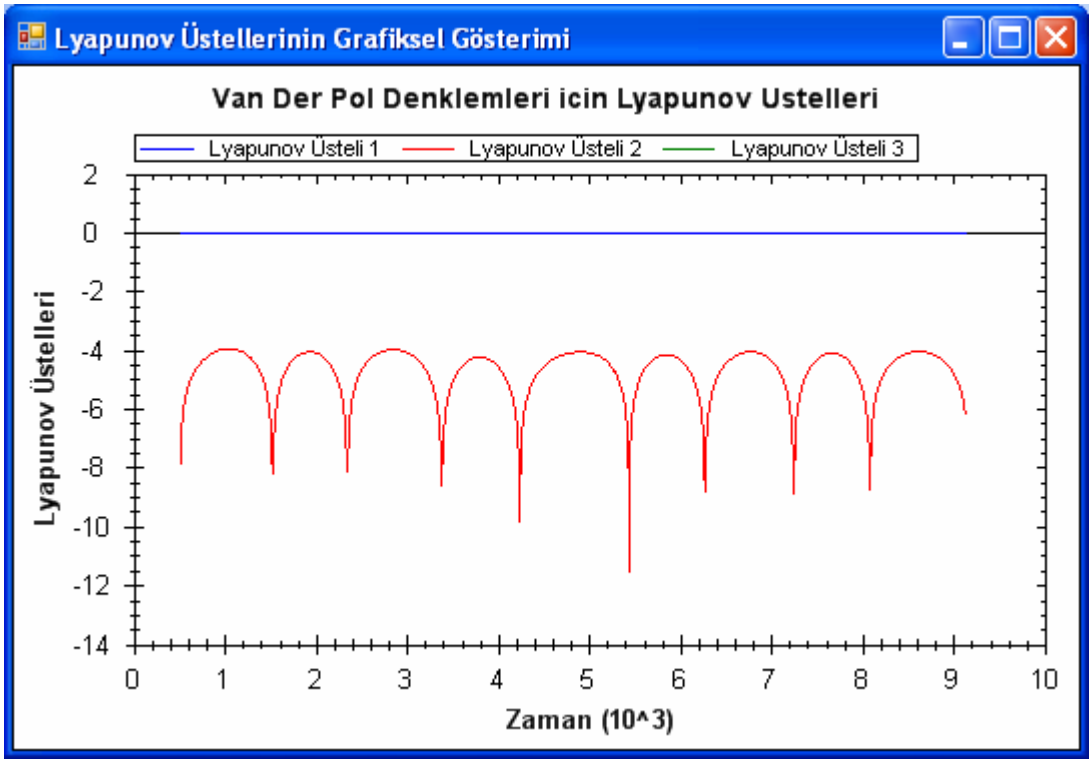
Van Der Pol denklemleri iki boyutlu sürekli zamanlı özerk olmayan bir sistemdir. Sistemin karakteristikleri denklem (6.9)'da verilmiştir.

$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = a*(1-x^2)*y - x^3 + b*\cos(c*t) \quad (6.9)$$

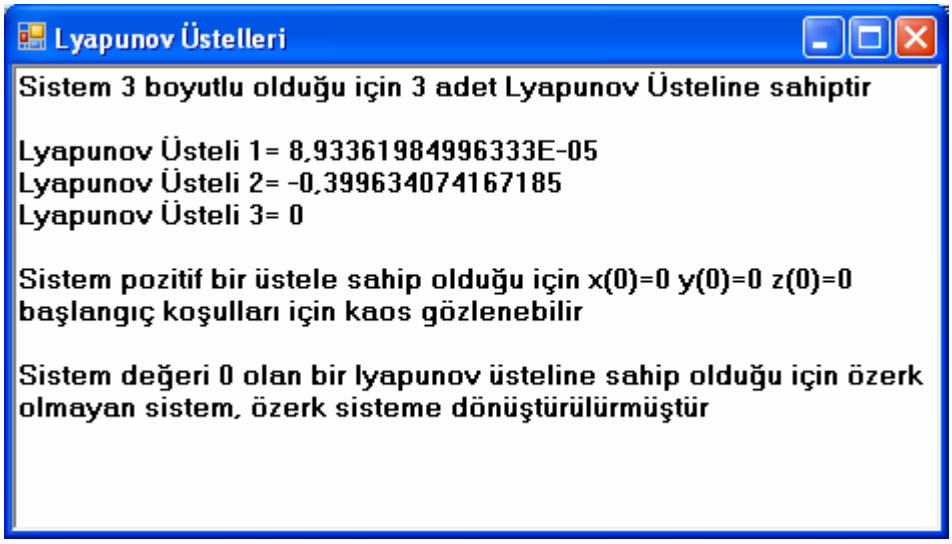
Denklemlerin özerk biçime dönüştürülmüş biçimi ise denklem (6.10)'da verilmiştir.

$$dx = y$$
$$dy = a*(1-x^2)*y - x^3 + b*\cos(c*z) \quad (6.10)$$
$$dz = 1$$

Denklemler x_0=0, y_0=0 ve z_0=0 başlangıç koşulları ve a=0.2, b=5.8 ve c=3 parametreleri için kaos göstermektedir. bu değerler için hesaplanan Lyapunov üstellerinin değişimi şekil 6.13'de üstellerin değerleri ve kaos analizi ise şekil 6.14'de gösterilmiştir.



Şekil 6.13 Van Der Pol denklemi için a=0.2, b=5.8 ve c=3 parametreleri için hesaplanan üsteller



Şekil 6.14 Van Der Pol denklemi için hesaplanan Lyapunov üstelleri ve kaos analizi

6.3.4 Stewart-McCumber Modeline Ait Lyapunov Üstelleri

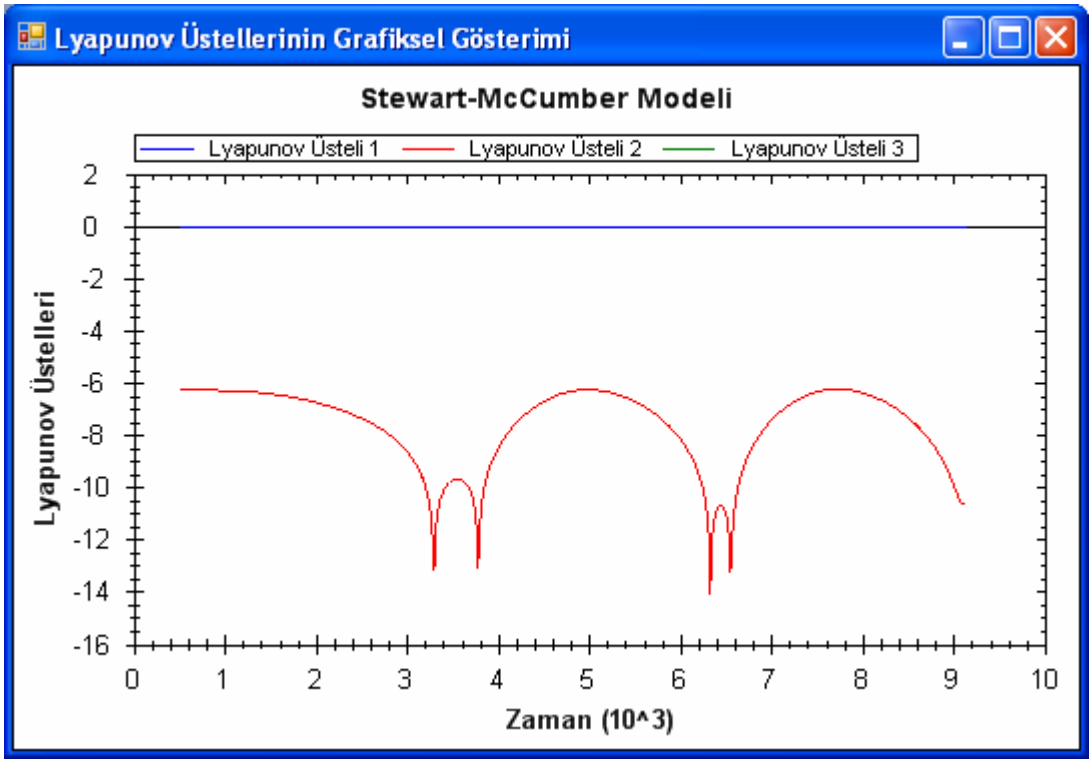
İki boyutlu özerk olmayan Stewart-McCumber modeline ait dinamikler denklem (6.11)'de verilmiştir.

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= 1/a[-y-\sin(x)+b+c*\sin(d*t)] \end{aligned} \tag{6.11}$$

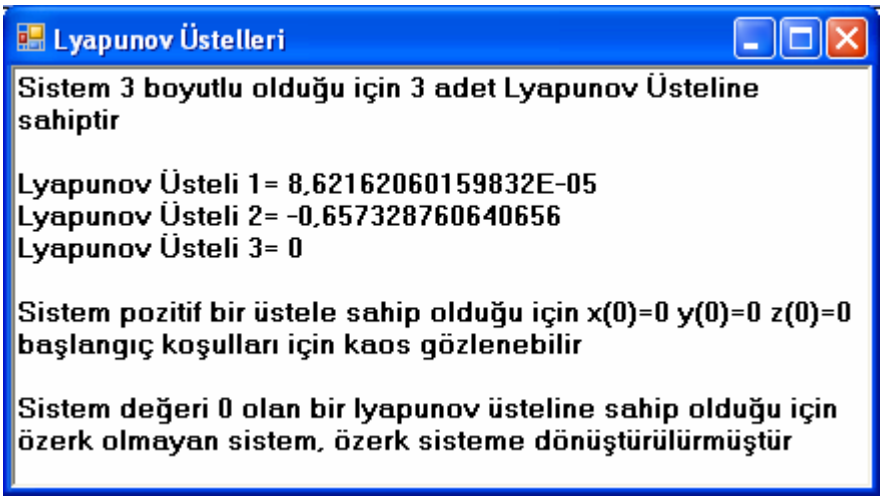
Özer olmayan sistemin özerk hale dönüştürülmesi denklem (6.12)'de verilmiştir.

$$\begin{aligned} dx &= y \\ dy &= (-y-\sin(x)+b+c*\sin(d*z))/a \\ dz &= 1 \end{aligned} \tag{6.12}$$

Sistem (x_0=0, y_0=0, z_0=0) başlangıç koşulları ve a=25, b=1.8 ve c=10.19804 parametreleri için modelde kaos gözlenmektedir. Bu değerler için hesaplanan Lyapunov üstellerinin değişimi şekil 6.15, değerleri ve kaos analizi ise şekil 6.16'da verilmiştir.



Şekil 6.15 Stewart-McCumber modeli için Lyapunov üstellerinin değişimi

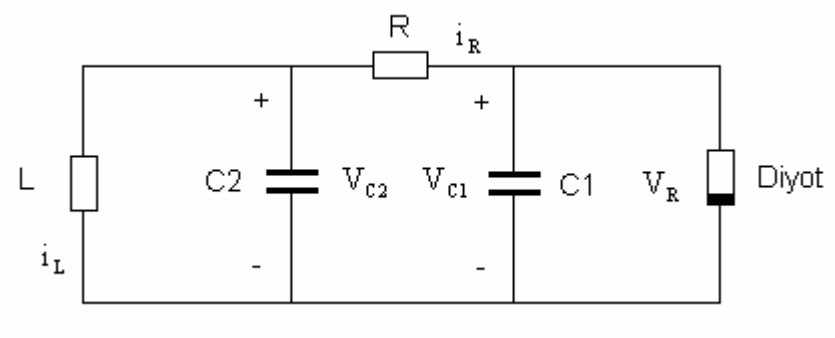


Şekil 6.16 Stewart-McCumber modeli için Lyapunov üstellerinin hesaplanan değerleri ve kaos analizi

6.3.5 Chua Devresine Ait Lyapunov Üstelleri

Elektronik devrelerde kaos ilk defa 1980'li yıllarda gözlenmiştir. Bu devredeki tüm davranış türleri bilgisayar benzetimi yardımıyla görüldükten sonra pratik olarak da gerçekleştirilmiştir ve birebir aynı sonuçlar elde edilmiştir. Bu devre dört tane doğrusal

elemandan ve bir doğrusal olmayan diyottan oluşmuş RLC devresidir. Devrenin şeması şekil 6.17'de verilmiştir.



Şekil 6.17 Chua devresi

Chua devresine ait sistemin dinamikleri denklem (6.13)'de verilmiştir.

$$\frac{dV_{C1}}{dt} = \frac{1}{C_1 R}\left(V_{C2} - V_{C1}\right) - f\left(V_{C1}\right)$$

$$\frac{dV_{C2}}{dt} = \frac{1}{C_2 R}\left(V_{C1} - V_{C2}\right) - i_L \qquad (6.13)$$

$$\frac{di_L}{dt} = -\frac{V_{C2}}{L}$$

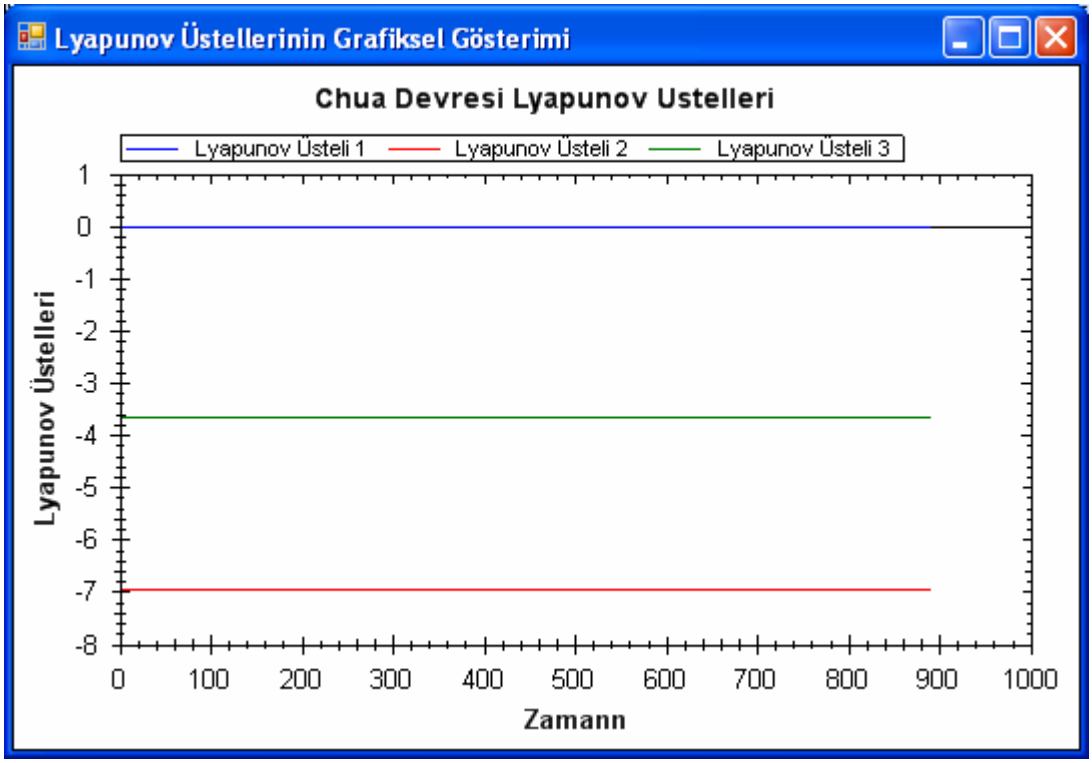
Denklem (6.13)'de gösterilen $f(V_{C1})$ diyotun karakteristiğini temsil etmektedir Bu karakteristik, matematiksel bir biçimde denklem (6.14)'deki gibi ifade edilebilir.

$$f\left(V_{C1}\right) = aV_{C1} + \frac{(b-a)}{2}\left(\left|V_{C1}+1\right| - \left|V_{C1}-1\right|\right) \qquad (6.14)$$

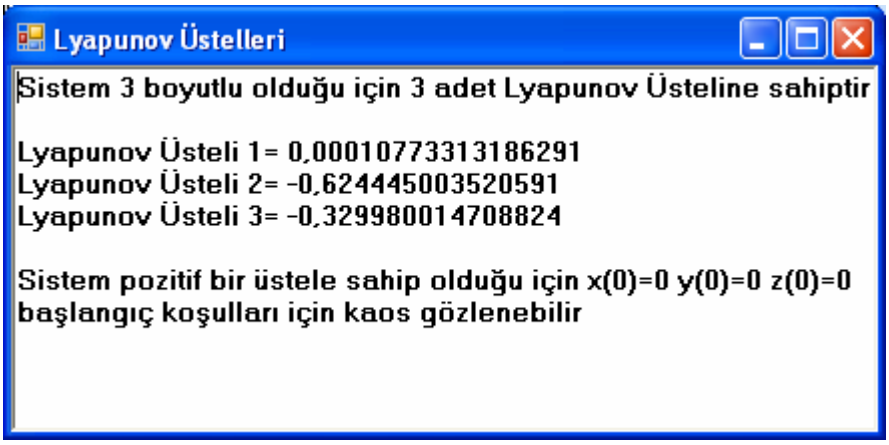
Denklemde $b \le -1 \le a$ şartı aranır.

Devre parametrelerinden $1/C_1R = 15.6$, $1/C_2R = 1$, $a = -5/7$ ve $b = -8/7$ parametreleri sabit tutularak, $1/L$ parametresi değiştirilerek sistemin davranışları incelenebilir. Sistem $1/L = 50$ iken sistem periyodik davranış göstermektedir. $1/L = 25.58$ olduğunda ise sistemin kaotik davranış göstermektedir.

Devrenin kaotik davranış gösterdiği parametrelerde hesaplanan Lyapunov üstellerinin değişimi ve değerleri sırasıyla şekil 6.18 ve şekil 6.19'da gösterilmiştir.



Şekil 6.18 Chua devresine ait Lyapunov üstellerinin değişimi



Şekil 6.19 Chua devresine ait Lyapunov üstellerinin değerleri ve kaos analizi

7 GERÇEKLEŞTİRİLEN YAZILIMIN DEĞERLENDİRİLMESİ VE BENZERLERİYLE KARŞILAŞTIRILMASI

7.1 Giriş

Bilgisayar sistemlerinin en önemli parçalarından biri olan yazılım sistemlerine daha önce Bölüm 5'de genel olarak değinilmiş, tanımı verilerek bir yazılımın modellenme süreci açıklanmıştır. Test aşamasında başarılı sonuçlar elde edilmiş ve kullanıma hazır olan bir yazılımın mühendislik ürünü olarak sunulabilmesi için bir bütün olarak değerlendirilmesi gerekmektedir. Herhangi bir yazılımın değerlendirilmesinde kullanılabilecek çeşitli kriterler bulunmaktadır. Yazılımın başarısı bu kriterleri ne ölçüde sağladığına bağlıdır.

Bu bölümde ilk olarak bu kriterler açıklanmış ve ardından gerçekleştirilen yazılımın bu kriterleri ne ölçüde sağladığı gösterilerek yazılımın başarısı gösterilmiştir. Son olarak literatürde bulunan benzer yazılımlar gösterilerek gerçekleştirilen yazılımın avantajları vurgulanmıştır.

7.2 Yazılım Sistemlerinin Değerlendirilmesinde Kullanılan Genel Kriterler

Giderek daha karmaşık ve daha yetenekli özelliklere bürünen yazılımların kullanım alanları her geçen gün çeşitlenerek artmaktadır. Bu uygulama alanlarından bazıları iş kayıt sistemleri, otomatik banka sistemleri, fabrika kontrol sistemleri, hava tahmin sistemleri, elektronik devre tasarım sistemleri, ofis iletişim sistemleri, tıbbi teşhis sistemleri, uçaklarda kullanılan uçuş yönetim sistemleri, hava trafik kontrol sistemleri, suç kayıt sistemleri gibi örneklendirilebilir [28].

Bu kadar karmaşık alanlarda kullanılan yazılım sistemlerin başarılı olarak kullanılabilmesi için tasarım aşamasında ortaya konulan ihtiyaçları ne ölçüde karşıladığı değerlendirilmelidir. Bu değerlendirme kriterleri aşağıdaki gibi sıralanabilir [28,29].

- Uygulama için uygunluk
- Hız
- Boyut
- Maliyet
- Veri taşınabilirliği
- Kod taşınabilirliği
- Satıcının şöhreti

- Öğrenmenin kolaylığı
- Kullanım kolaylığı
- Bakımın kolaylığı

Gerçekleştirilen yazılım dikkate alınarak bu kriterler değerlendirilirse yazılımının test edilen tüm veriler için başarılı olduğu görülmektedir. Hız bakımından program kabul edilebilir süre içerisinde cevap vermekte, program boyutu olarak günümüz bilgisayarlarında göz ardı edilebilecek düzeyde küçük olması, kullanımı için herhangi bir maliyet gerektirmemesi yazılımın üstün özelliklerinden bazılarıdır. Yazılımın veri taşınabilirliğine izin vermesi ve istenildiği takdirde kod taşınabilirliğine müsaade etmesi ayrıca kolay kullanımı ve basit arayüzü ile belirtilen kriterleri başarılı şekilde sağlamaktadır.

7.3 Yazılım Sisteminin Programlama Dilinin Seçilmesi

Bir yazılım sisteminin başarısını etkileyen en önemli faktörlerden biride yazılım sisteminin kodlandığı programlama dilinin seçimidir. Bilgisayar sistemleri ve buna bağlı olarak programlama dilleri de çok hızlı bir gelişme süreci göstermiştir. Yapısal programlama tekniklerinden nesneye yönelik programlama tekniklerine geçilmesi, programların ancak satır satır yazılıp günlenebildiği editörlerin yerini otomatik kod üreteçlerinin alması çok hızlı olmuştur [25]. Bu değişim giderek artmakla birlikte her geçen gün yeni bir programlama dilinin ve yazılım platformunun ortaya çıkması yazılım sistemlerinin geliştirilirken dikkatli bir seçim yapayı zorunlu hale getirmiştir.

Bir programlama dilinin seçiminde dikkat edilecek ölçütlerden bazıları [28]

- Hızlı geliştirme süreci
- Daha iyi kullanıcı ilişkileri
- Daha esnek sistemler
- Donanım bağımsızlığı
- Veri tabanı sistemlerinden bağımsızlık
- Platformdan bağımsızlık
- Nesne yönelimli programlama tekniklerine uygunluk
- Maliyet
- Hız
- Yaygınlık
- …

şeklinde sıralanabilir. Bu ölçütlere dikkat edilerek gerçekleştirilen bir yazlım sistemi diğerlerine göre daha güçlü olacağı bir gerçektir.

Gerçekleştirilen yazılımda programlama dili olarak C# dili kullanılmıştır. C# dili modern, nesne yönelimli ve aynı zamanda tip güvenliğine büyük önem veren bir dildir. Temel hesaplamaları yapmasının yanında birçok uygulamayı geliştirmek için sağladığı kütüphanelerle yazılım tasarım sürecini kolaylaştırmakta ve kısaltmaktadır. C# dilinin güçlü yönlerinin en başında C++ dilinin başarılı özelliklerini içinde bulundurması gelmektedir.

C# programlama dili; .NET platformu için geliştirilmiş bir dildir. Yazılan kodlarını orta seviye koduna çevrilerek derlemektedir. Bu özelliği sayesinde C# dili ile yazılan bir kod diğer .NET platformuna uyumlu diller tarafından tekrar kullanılabilir. Buda kod taşınabilirliği için güzel bir özelliktir. C# otomatik bellek yönetimi sağlamasının yanında programcının belleği kendisinin de yönetmesini sağlayan bir sistem sunması diğer bir avantajıdır.

Programlama dilinin bu avantajları göz önünde bulundurulursa yazılımın bu noktada da benzerlerinden daha başarılı olduğunun bir göstergesidir.

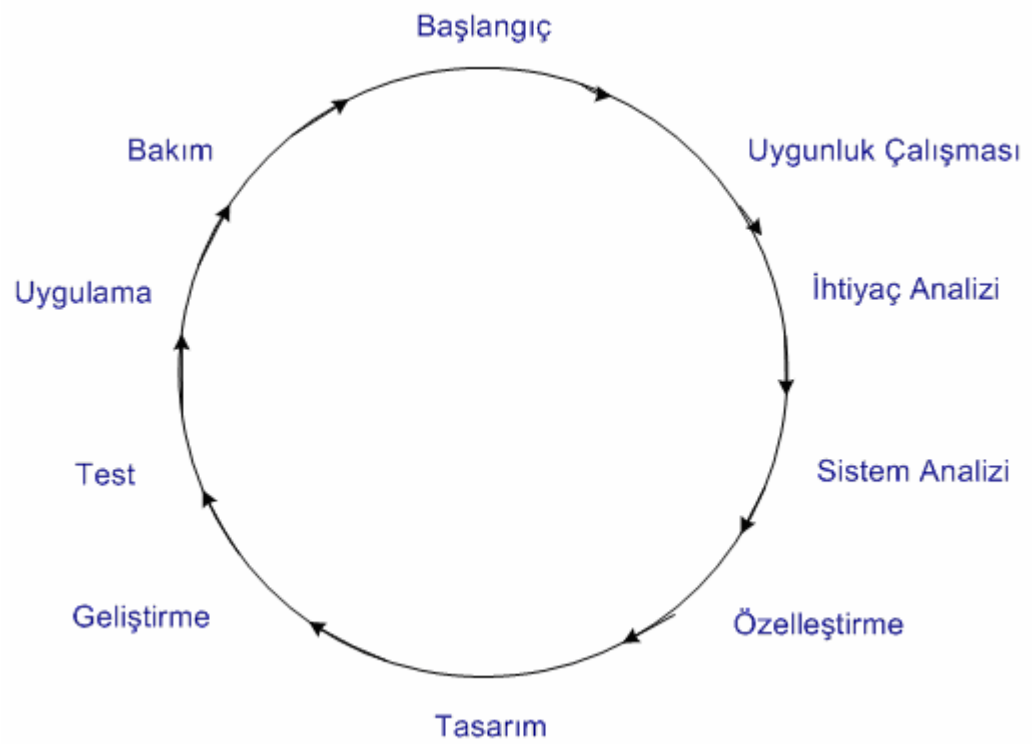
7.3 Yazılımın Tasarım Metodu

Yazılım projelerinin öngörülebilen maliyet ve zamanda tamamlanabilmesi için ciddi bir proje yönetimi ve mühendislik tasarımı yapılması gerektiği daha önce açıklanmıştı. Bir yazılım sistemi değerlendirilirken en önemli ölçütlerden biride yazılımın tasarımı boyunca kullanılan metottur. Yazılım geliştirme proje sürecinde birçok yaklaşım olmasına rağmen pek çoğu günümüz sistemlerinin tasarımı için uygun değildir. Gerçekleştirilen yazılımda ise sürekli bir güncelleme ve bakım içererek eksiklikleri gideren döngüsel tasarım süreci kullanılmıştır. Kullanılan metodun aşamaları şekil 7.1'de gösterilmiştir.

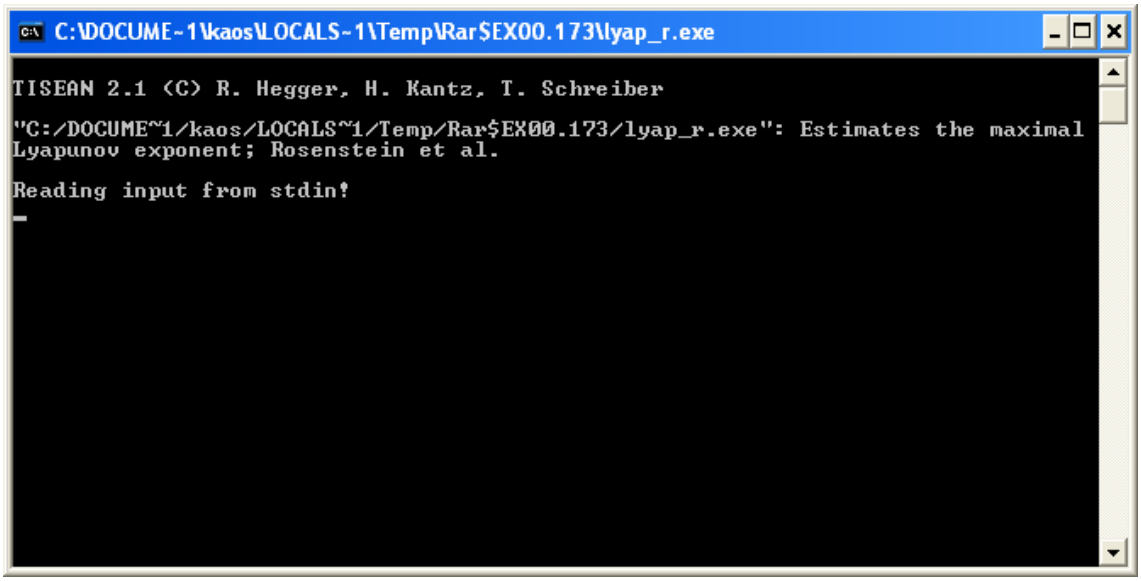
7.4 Gerçekleştirilen Yazılımın Benzerleriyle Karşılaştırılması

Bu bölümde literatürde Lyapunov üstellerini hesaplamak için geçen yazılımlar gösterilmiştir. Bu alandaki ilk örneklere bakıldığında Ms-Dos tabanlı olduğu ve belirli uygulamalar için çalıştığı görülmüştür. Bu tip programlara bazı örnekler şekil 7.2 ve şekil 7.3'de gösterilmiştir. Yakın zamanda geliştirilen yazılımlara bakıldığında ise Windows tabanlı çeşitli yazılımların olduğu görülmüştür. Bu yazılımlardan en başarılısı [32] matlab ortamında gerçekleştirilen LET programıdır. Ancak programda hesaplamalar için denklem girişlerinin problem olması denklemlerin Jacobian matrisinin girişine ihtiyaç duyması matematiksel yönü

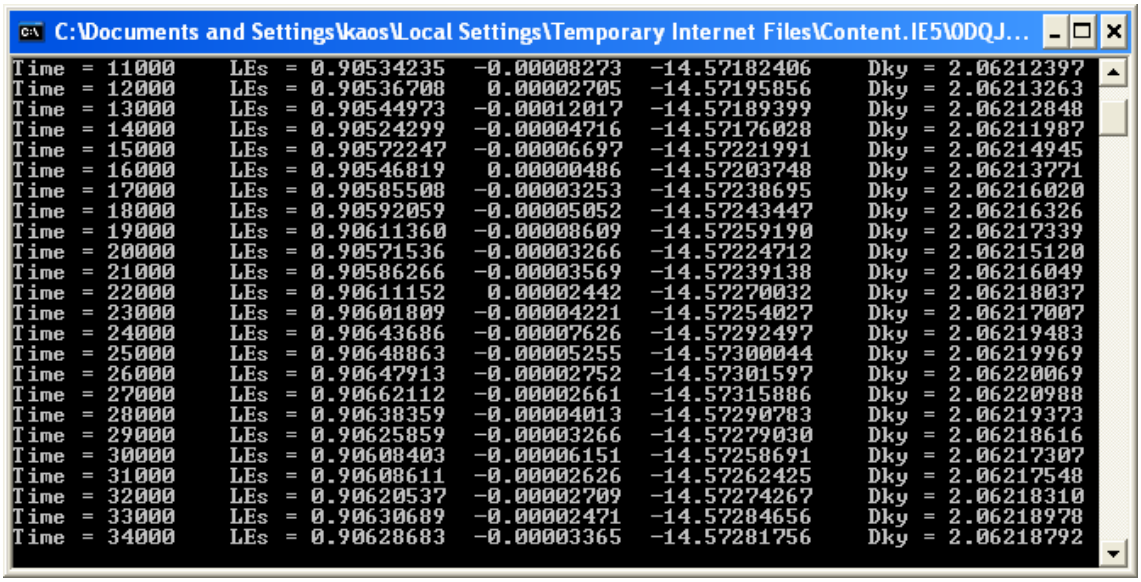
zayıf kullanıcılar için büyük bir problemdir. Kaos analizinin disiplinler arası bir konu olmasından dolayı mevcut programların minimum düzeyde matematik bilgisi gerektirecek şekilde tasarlanmayı bir gereklilik olarak ortaya çıkmaktadır. LET programına ait ekran görüntüsü şekil 7.4'de gösterilmiştir.



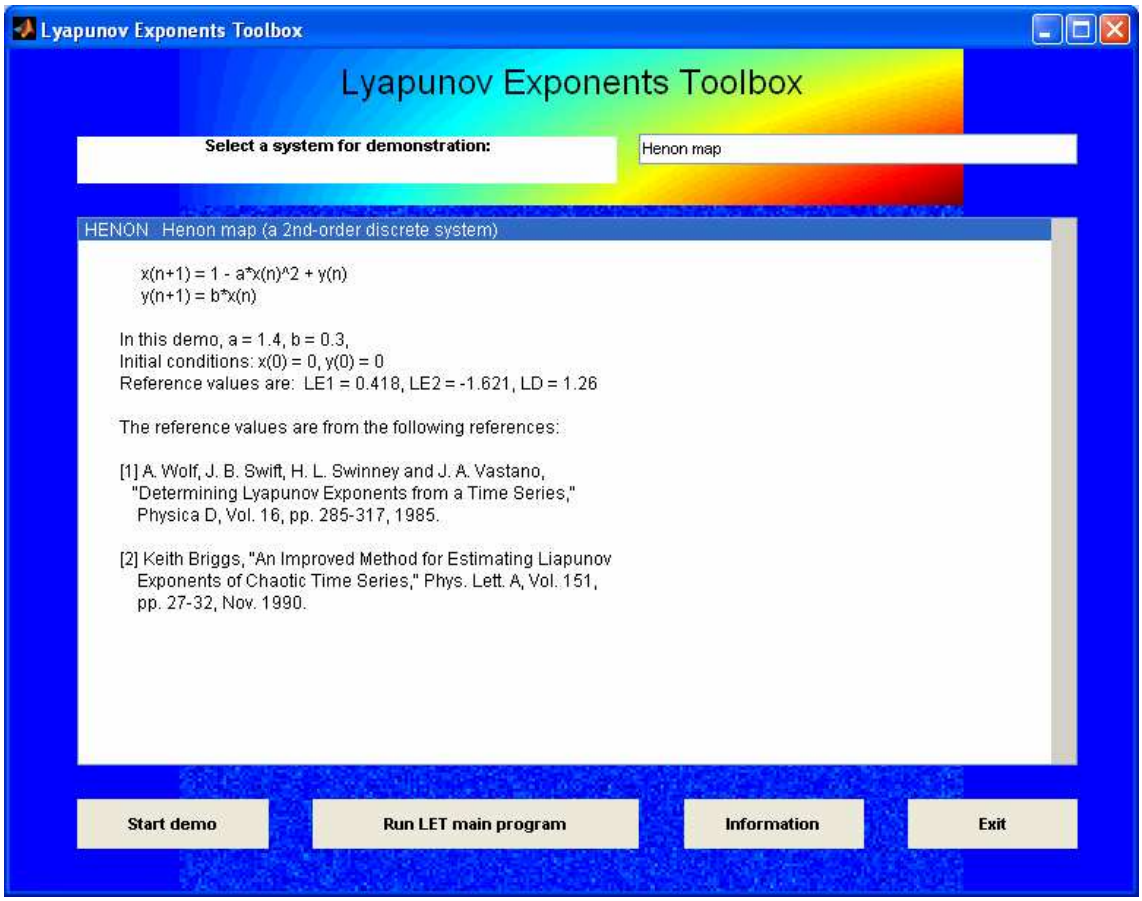
Şekil 7.1 Yazılım geliştirme döngüsü



Şekil 7.2 Lyapunov üstellerini hesaplayan bir program [30]



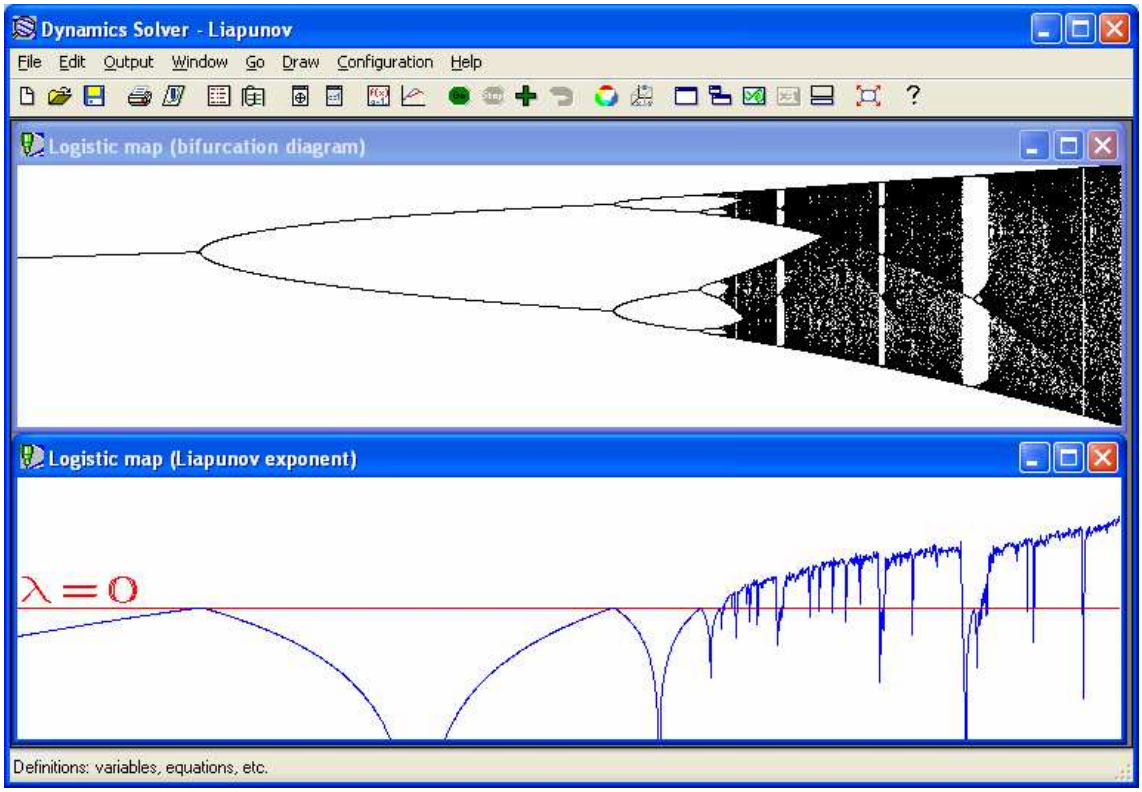
Şekil 7.3 Lyapunov üstellerini hesaplayan bir program [31]



Şekil 7.4 LET (Lyapunov Exponents Toolbox) programı ekran görüntüsü [32]

Lyapunov üstellerini hesaplayan bir diğer program “Dynamics Solver” şekil 7.5’de gösterilmiştir [33]. Gerçekleştirilen program mevcut sistemler için başarılı sonuçlar

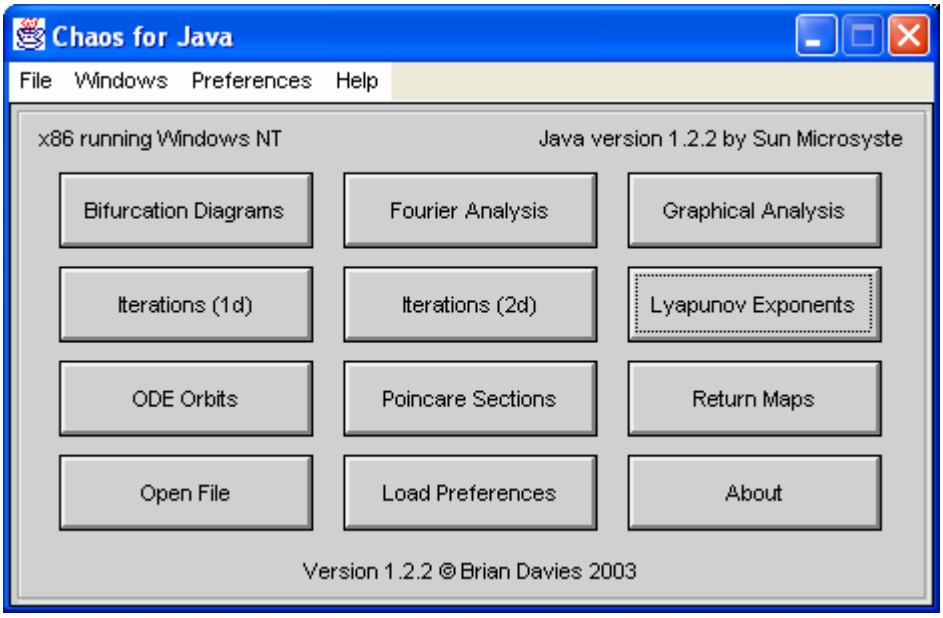
üretmektedir. Ancak program yeni sistemler için çalıştırılmak istendiğinde arayüzünün karmaşık olması ve sistemin amacının sadece Lyapunov üstellerini hesaplamak olmaması programın dezavantajlarından bazılarıdır.



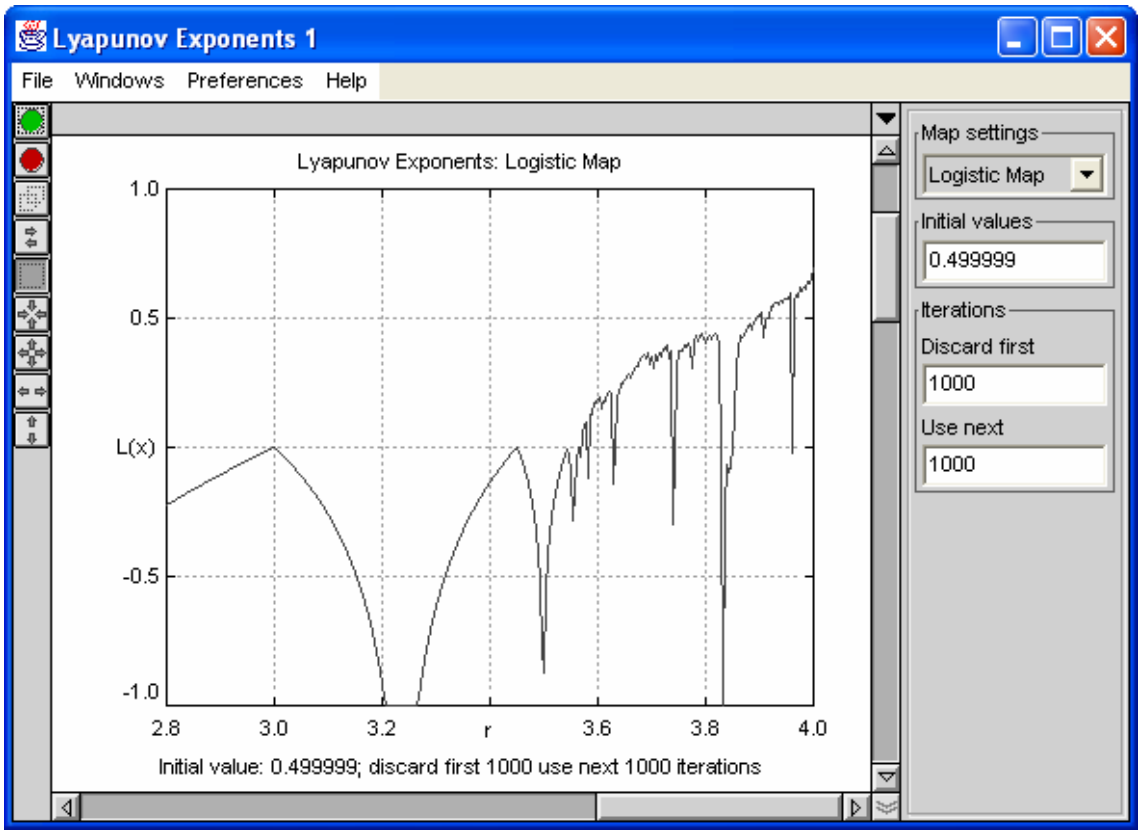
Şekil 7.5 Dynamics Solver programı ekran görüntüsü [33]

Lyapunov üstellerini hesaplayan programlara son örnek java ortamında geliştirilmiş bir programdır. Programın ana ekran görüntüsü şekil 7.6'da lojistik harita için Lyapunov üstellerinin değişimi şekil 7.7'de gösterilmiştir. Bu programın sadece belirli sistemler için çalışıyor olması sistemdeki önemli bir eksikliktir.

Gerçekleştirilen yazılım yukarıda verilen problemleri önemli ölçüde gidermesinden dolayı benzerlerinden başarılı şekilde ayrılmaktadır. Gerçekleştirilen yazılımda sistemin sadece Lyapunov üstellerini hesaplamak gibi tek bir amaca yönelik olarak tasarlanması, matematiksel olarak önemli bir bilgi gerektirmemesi sadece sistem denklemlerinin hesaplamaları yapmak için yeterli olmasının yanında basit arayüzü, hızlı hesaplama kabiliyeti, grafik arayüzü, kolay kullanımı ve zengin yardım menüleri ile benzerlerinin önüne geçmektedir.



Şekil 7.6 "Chaos for Java" programın ana ekran görüntüsü [34]



Şekil 7.7 "Chaos for Java" programının Lyapunov üstelleri hesaplaması [34]

8 SONUÇ VE ÖNERİLER

8.1 Sonuçlar

Bu tez çalışmasında doğrusal olmayan sistemlerde Lyapunov üstellerini hesaplayarak kaos analizi yapan bir yazılım gerçekleştirilmiştir. Yazılımın hem ayrık zamanlı hem de sürekli zamanlı sistemlere uygulanabilmesi, grafik arayüzüyle değişimin grafiksel olarak gözlemlenebilmesi ve ihtiyaç duyulduğunda hesaplanan verilerin başka amaçlar içinde kullanılabilmesi yazılıma güç kazandırmıştır.

Tezde ilk olarak kaos teorisi açıklanmış ve kaos analizinde kullanılan yöntemler kısaca açıklanmıştır. Kaosu gösteren birçok yöntemin bu işlemi sadece nitel hesaplamalar kullanarak yapması önemli bir eksiktir. Gerçekleştirilen yazılımda ise kullanılan Lyapunov üstelleri yöntemi kullanılarak kaos analizini nicel hesaplamalarla yapması geliştirilen yazılımın diğer güçlü bir yanıdır.

Bilimsel araştırmalarda kullanılan yazılımların öneminin gittikçe artmasına paralel olarak kullanıcıların yazılımlardan beklentisi de artmaktadır. Literatürde kaos analizi yapan birçok programın komut tabanlı olması ve yalnızca belirli uygulamalar için geliştirilmesi önemli bir eksiktir. Gerçekleştirilen yazılımın Windows tabanlı olması ve grafiksel olarak zengin bir içeriğe sahip olması yazılımın bir diğer artısıdır.

Birçok yazılım projesinde önemli bir problem yazılım geliştirme döngüsü içerisinde yazılımda herhangi bir değişiklik, bakım ve onarım çalışmasının maliyetinin (zaman veya para) çok yüksek olmasıdır. Gerçekleştirilen yazılımda hem gereksinimlerin belirlenmesi aşamasında hem de kodlama aşamasında kullanılan nesne tabanlı mimari yazılımın gereksinimleri değiştiğinde kolayca yeni ihtiyaçları karşılayacak şekilde uyarlanabilir olması önemli bir faktördür.

Bir bütün olarak değerlendirme yapıldığında yazılımın basit, hızlı ve test aşamasında başarılı olduğu görülmüştür.

8.2 Öneriler

Bir yazılım geliştirme sürecinde yazılımın üç farklı sürümü bulunmaktadır. Bunlar:

- Alfa sürümü: Yazılımı geliştiren ekibin, yazılımı çeşitli veriler üzerinde test ettiği biçimidir.

- Beta sürümü: Yazılım kullanıcılara sunularak, kullanıcıların karşılaştıkları problemlerin ve yazılımın hatalarının belirlenerek giderildiği biçimidir.
- Programın güncel sürümleri (1.0, 2.0,…)

Gerçekleştirilmiş olan yazılımda şu an için alfa sürümünü (LÜKA 0.1) içermektedir. Program birçok kullanıcı tarafından kullanılmalı ve kullanıcılardan gelen bilgiler kullanılarak varsa hatalar giderilmeli, yeni gereksinimler ortaya çıkmışsa giderilmelidir.

Bilimim hızlı ilerlemesi göz önünde bulundurularak yazılımın çok uzun olmayan aralıklarla yeni sürümleri çıkarılmalı veya güncellemeleri yapılmalıdır.

Yazlım sektöründe işlemlerin artık gittikçe web tabanlı olması gerçekleştirilen yazılımın web tabanlı bir sürümünü zorunluluk haline getirmektedir. Bu nedenle gerçekleştirilen yazılımın web tabanlı bir sürümü de gerçekleştirilmelidir.

Gerçekleştirilen yazılımın dili Türkçedir. Yazılımın evrensel olarak kullanılabilmesi için İngilizce bir sürümü de hazırlanmalıdır.

Yazılım Windows tabanlı sistemlere göre geliştirilmiştir. Unix tabanlı sistemler içinde programın sürümleri gerçekleştirilmelidir.

Gerçekleştirilen yazılımda kaos analiz yöntemi olarak sadece Lyapunov üstelleri kullanılmıştır. Lyapunov üstellerinin diğer analiz yöntemlerine göre belirgin üstün yanları olsa da diğer analiz yöntemleri ile kaos analizi yapabilmek için bu yöntemlerde yazılıma dahil edilebilir. Kullanıcı dinamiğini girdiği sistemi istediği yönteme göre analiz edebilir.

Gerçekleştirilen yazılım sadece ayrık ve sürekli zamanlı sistemlerde kaos analizi yapabilmektedir. Benzer mantıkla yazılım zaman serilerinde kaos analizi yapmak içinde genişletilmelidir.

Ayrıca kaos teorisinin kardeş dalı olan fraktalların analizini gerçekleştiren bir yazılımda sisteme eklenerek yazılımın cazibesi artırılabilir.

KAYNAKLAR

1. Yıldırım, C., 1991, Bilim Felsefesi, Remzi Kitapevi, 256 s.
2. Özer, A. B., 2005, Elektriksel Sürücü Sistemlerinde Doğrusal Olmayan Olguların Kaotik Analizi ve Yumuşak Hesaplama Yöntemleri İle Denetimi, Doktora Tezi, Fırat Üniversitesi Fen bilimleri Enstitüsü
3. Strogatz, S.H., 1994, Nonlinear Dynamics and Chaos, Perseus Books Publishing, New York, 498 p.
4. Alligood, K.T., Sauer, T.D. and Yorke, J.A., 1997, Chaos, Springer-Verlag, New York, 603p.
5. Ruelle D., 2004, Raslantı ve Kaos, Tübitak Yayınları, 183 s.
6. Gleick J., 2003, Kaos, Tübitak Yayınları, 412 s.
7. Hilborn, R.C., 2003 (second edition), Chaos and Nonlinear Dynamics, Oxford University Press, New York, 650 p.
8. Baker, G.L. and Gollub, J.P., 1990, Chaotic Dynamics, Cambridge University Press, Cambridge, 179p.
9. Hoppensteadt, F.C., 2000, Analysis and Simulation of Chaotic Systems, Springer-Verlag, New York, second edition, 315p.
10. Kapitaniak, T., 1998, Chaos for Engineers, Springer-Verlag, New York, 140p.
11. Kaynak, O., Efe, Ö., 2000, Yapay Sinir Ağları ve Uygulamaları, Boğaziçi Üniversitesi, 141 s.
12. Gündüz G., Anaksimenes'te ve Günümüzde Kaos Anlayışı, İstanbul Kültür Üniversitesi Yayınları, 2004
13. Castillo, O. and Melin, P., 2001, Soft computing for Control of Non-linear Dynamical Systems, Physica-Verlag, New York, 221p.
14. Banerjee, S. and Verghese, G.C., (Editors), 2001, Nonlinear Phenomena in Power Electronics, IEEE Pres, New York, 440p.
15. Wolfe A., Swift J.B., Swinney H.L., and Vastano J. A., Determining Lyapunov Exponents From a Time Series. *Physica D* 16:285-317, 1985
16. Kinsner, W., Characterizing Chaos Though Lyapunov Metrics, IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Rewiews Vol 36 No:2 March 2006, 141-151 p.
17. Small M., Applied Nonlinear Time Series Analysis: Application Physics, Physiology and Finance, World Scientific Publishing Company, 2003

18. Kantz H. Schreiber T.,1997, Nonlineat Time Series Analysis, Cambridge Nonlinear Science 7, 304 p.
19. Kaplan, J. L., and Yorke, J. A., Chaotic behavior of multidimensional difference equations," in Functional Differential Equations and Approximation of Fixed Points, H.-O. Peitgen and H. O. Walther, Eds. New York: Springer-Verlag, 1979, pp. 204–227.
20. Steeb, W.H., 1999, the Nonlinear Workbook, World Scientific Publishing, Farrer Road, Singapore, 585p.
21. Parker, T.S. and Chua L.O., 1989, Practical Numerical Algorithms for Chaotic Systems, Springer-Verlag, New York, 348p.
22. Öztürk F., Özbek L., 2004, Matematiksel Modelleme ve Simülasyon, Gazi Kitabevi,
23. Law, A. M., Kelten, W. D., 2000, Simulation Modeling and Analysis, Mcgraw-Hill, 760 p.
24. Arifoğlu A., 2001, Yazılım Mühendisliği, SAS Bilişim 361 s.
25. Baransel, C., Mumcuoğlu, A., 2003, Web Tabanlı Üç Katmanlı Yazılım Mimarileri, SAS Bilişim, 364s.
26. Grady, B., 1999, The Unified Modeling Language User Guide, Addison-Wesley, 482 p.
27. Güngören B., 2005, UML ile Nesne Tabanlı Çözümleme ve Tasarım, Seçkin Yayınevi, 222 s.
28. Holloway S., 1990, Fourth-Generation Systems Their Scope, Application and Methods of Evaluation, Chapman and Hall, 374p.
29. Domenico F., 1983, Measurement and Tuning of Computer Systems, Prentice -Hall, 523 p.
30. Hegger R., Kantz H., Schreiber T., Nonlinear Time Series Analysis, 2000 http://www.mpipks-dresden.mpg.de/~tisean/TISEAN_2.1/index.html
31. Sprott J., C., 2005, http://sprott.physics.wisc.edu/chaos/lespec.htm
32. Siu, S. W. K,. 1998, Lyapunov Exponent Toolbox (LET), ftp://ftp.mathworks.com/pub/contrib/v5/misc/let.
33. Aguirregabiria, J., M., 1992, Dynamics Solver, http://tp.lc.ehu.es/jma/ds/ds.html
34. Davies B., 1999, Chaos for Java, http://wwwmaths.anu.edu.au/~briand/chaos/software.html
35. Korsch, H.J. and Jodl, H.-J., 1999, Chaos, Springer-Verlag, Berlin, second edition, 311p.
36. Hale, J.K. and Koçak, H., 1991, Dynamics and Bifurcations, Springer-Verlag, New York, 568p.

37. Devaney, R.L., 2003 (second edition), Chaotic Dynamical Systems, Westview Press, Colorado, 335 p.
38. Baykal, N. ve Beyan, T., 2004, Bulanık Mantık Uzman Sistemler ve Denetleyiciler, Bıçaklar Kitapevi, Ankara, 508s.
39. Melin P., Castilo O., 2002, Modelling Simulation and Control of Non-Linear Dynamical Systems, Taylor and Francis, 249 p.
40. Walter, J. M.,. 1998, The Nonlinear Pendulum Project:PHSI 362 Project, University of Ogata..
41. Ruelle R., Deterministic Chaos: The Science and The Friction, Proc. Roy.Soc. Lon. 427A pp. 87-114, 1990
42. Mathews H. Fink K., 2004, Numerical Metods Using Matlab, Prentice Hall, 680 p.
43. http://www.chaoskit.com/

ÖZGEÇMİŞ

1983-Elazığ doğumlu olan Fatih ÖZKAYNAK ilk, orta ve lise öğrenimini Elazığ'da tamamladı. 2001 yılında girdiği Fırat Üniversitesi Bilgisayar Mühendisliği Bölümünden 2005 yılında mezun oldu. Aynı yıl Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Ana Bilim dalında yüksek lisans yapma hakkı kazanmıştır.