Pamukkale Univ Muh Bilim Derg, 27(2), 234-243, 2021

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Yeni bir Julia tabanlı sistem tanımlama dili ve benzetim ortamı: JuSDL

A novel Julia based system description language and simulation environment: JuSDL

Zekeriya SARI[1*], *Serkan GÜNEL*[2]

[1,2]Elektrik-Elektronik Mühendisliği Bölümü, Mühendislik Fakültesi, Dokuz Eylül Üniversitesi, İzmir, Türkiye.
zekeriya.sari@deu.edu.tr, serkan.gunel@deu.edu.tr

Geliş Tarihi/Received: 20.12.2019 Düzeltme Tarihi/Revision: 19.03.2020 doi: 10.5505/pajes.2020.03591
Kabul Tarihi/Accepted: 28.03.2020 Araştırma Makalesi/Research Article

Öz

Bu çalışmada, Julia programlama dili tabanlı bir tanımlayıcı sistem dili ve amaca yönelik hızlı ve etkili sistem benzetimlerine ve çevrimiçi ve çevrimdışı çözümlemelerine olanak sağlayan bir benzetim ortamı geliştirilmiştir. Geliştirilen benzetim ortamında ayrık zamanlı ya da sürekli zamanlı, statik ya da dinamik sistemlerin benzetimleri mümkündür. Özellikle, adi, rastgele adi, rassal, cebirsel, gecikmeli türev denklemleri ve ayrık fark denklemleri gibi çok farklı denklem türleri ile modellenen dinamik sistemlerin benzetimi yapılabilmektedir. Benzetim sırasında modelin bağlantıları üzerinden akan veri çevrimiçi ve çevrimdışı olarak işlenebilmekte ve özelleşmiş çözümlemeler yapılabilmektedir. Bu çözümlemelerin, standart Julia kütüphanesi ya da çeşitli Julia paketleri kullanılarak kolaylıkla tanımlanabilecek eklentiler ile de zenginleştirilmesi mümkündür. Benzetim model bileşenlerinin bireysel ve örnekleme zaman aralıklarında eşzamanlı ve paralel evrilmesi ile yapılır. Bileşenlerin birbirinden bağımsız evrilmesi farklı matematiksel denklemler ile ifade edilen bileşenlerden oluşan modellerin benzetimine olanak sağlarken; bileşenlerin eşzamanlı ve paralel evrilmesi ise benzetim hızını artırmaktadır.

Anahtar kelimeler: Sistem benzetimi, Sistem modelleme, Dinamik sistemler, Julia programlama dili.

Abstract

In this study, a Julia programming language based system description language and simulation environment that enables fast and effective system simulations together with online and offline data analysis is introduced. In the simulation environment developed, it is possible to simulate discrete time or continuous time, static or dynamical systems. In particular, it is possible to simulate dynamical systems modeled by different types of equations, such as the ordinary differential, random ordinary differential, stochastic differential, differential-algebraic, delayed differential equations, and discrete-time difference equations. During the simulation, the data flowing through the links of the model can be processed online and offline, and specialized analysis can be performed. These analyzes can also be enriched with plugins that can be easily defined using the standard Julia library or various Julia packages. The simulation is performed by evolving the model components individually and parallelly between sampling time intervals. The independent evolution of the components allows the simulation of the models consisting of the components represented by different mathematical equations, while the parallel evolution of components increases the simulation speed.

Keywords: System simulation, System modeling, Dynamical systems, Julia programming language.

1 Giriş

Sayısal benzetimler temel olarak yapılan modelleme sonucunda elde edilen matematiksel denklemlerin çözümlenmesi ve/veya bu çözümlemeler sonunda elde edilen verinin bilgisayar yardımı ile işlenmesi olarak ifade edilebilir. İlgilenilen sistemin özelliklerinin ve yapılan modellemenin durumuna göre elde edilen matematiksel denklemler adi, rassal, zaman gecikmeli türev ya da fark denklemleri olabilir. Yapılan benzetimin hızlı olması, farklı türde çözümleme araçları sunması da etkili bir benzetim ortamından beklenen diğer tipik özelliklerdir.

Sistemlerin sayısal benzetimleri için farklı benzetim ortamları geliştirilmiştir. [1]'de sürekli-zamanlı dinamik sistemlerin modellenmesi için geliştirilen sistem modelleme dilinin benzetim ve modelleme imkânları [2] ve [3]'te yapılan çalışmalar ile genişletilmiştir. Hibrit bağ yapıları ile modellenme veya ayrık/sürekli zamanlı bileşenler içerme gibi özel modellemelere sahip sistemlerin benzetimleri için de özelleşmiş araçlar geliştirilmiştir [4],[5]. Benzetim sırasında sistemlerde görülebilecek yapısal değişiklikler özel olarak ele alınmıştır [6],[7]. Grafiksel kullanıcı ara yüzleri ile kullanıcılara hızlı ve kolay modelleme, benzetim ve çözümleme yapma olanağı sunulan [8] ve [9]'da verilen araçlar için özelleşmiş eklentiler geliştirilmiştir [10],[11]. Dinamik sistemlerin çeşitli özelliklerinin detaylı çözümlemelerine olanak sağlayan tümleşik kütüphaneler de mevcuttur [12],[13].

Bu benzetim ortamları genellikle yalnızca adi türevsel denklemler veya türevsel cebirsel denklemler ile ifade edilen modellerin benzetimine olanak sağlayabilmektedir. Karşılaşılabilecek matematiksel modellerin çeşitliliği düşünüldüğünde bu durum kısıtlayıcıdır [14]. Ayrıca, mevcut benzetim ortamlarının birçoğu etkili bir benzetim ortamından beklenen hız gerekliliğini karşılamak amacıyla kullanılan paralel hesaplama teknikleri gibi modern benzetim tekniklerinden de uzaktır.

Bu çalışmada, Julia programlama dili tabanlı bir tanımlayıcı sistem dili ve benzetim ortamı JuSDL geliştirilmiştir[15]. Amaç çok sayıda sistem bileşenlerinden oluşan karmaşık sistem ağlarının kolaylıkla modellenebilmesi ve hızlı, etkili benzetimlerinin yapılabilmesini sağlamaktır. Bu amaç doğrultusunda, yüksek performanslı sayısal çözümleme ve hesaplamalı bilim için tasarlanmış, açık kaynak kodlu, yüksek seviyeli, genel amaçlı dinamik programlama dili olan Julia programlama dili kullanılmıştır. Julia programlama dili,

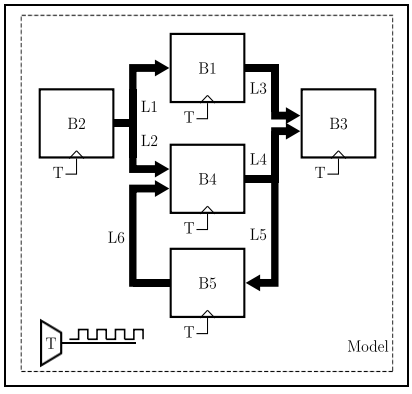
* Yazışılan yazar/Corresponding author

dinamik bir dil olmasına karşın, LLVM (Low Level Virtual Machine) üzerine geliştirilen JIT(Just-in-Time) derleyicisi sayesinde C programlama dili gibi statik dillerin yüksek hız performanslarına ulaşabilmektedir [16]-[18]. İzlek(thread) ve süreç(process) seviyesinde farklı paralelleme tekniklerini desteklemektedir. Julia'nın standart kütüphanesine ek olarak, Julia topluluğu tarafından veri bilimi, bilimsel hesaplama vb. farklı alanlar için geliştirilmiş çok sayıda özelleşmiş paket de kullanıma açıktır. Julia'nın yüksek hız performansı ve paralelleme desteği, geliştirilmek istenen benzetim ortamının hızlı ve etkili benzetim yapabilme gereksinimini karşılaması açısından önemlidir. Metaprogramlama desteği aracılığıyla Julia'nın yazım kuralları amaca yönelik olarak geliştirilebilmektedir. Bu çalışmada önerilen benzetim ortamının çözümleme olanakları, standart Julia kütüphanesi ya da çok sayıdaki özelleşmiş Julia paketleri kullanılarak kolaylıkla tanımlanabilen yeni eklentiler ile genişletilebilir. Benzetim ortamında ayrık ya da sürekli zamanlı, statik ya da dinamik sistemlerin birlikte çözümlemeleri mümkündür. Özellikle, ayrık fark denklemleri ve adi, rastgele adi, rassal, cebirsel, gecikmeli türev denklemleri gibi çok farklı denklem türleri ile modellenen dinamik sistemlerin benzetimi yapılabilmektedir. Benzerlerinin aksine, model bir bütün halinde benzetim süresi boyunca tek seferde evrilmez. Bunun yerine, model bileşenleri bireysel ve örnekleme zaman aralıklarında evrilir. Bileşenlerin bireysel evrilmesi farklı matematiksel modeller ile ifade edilen bileşenlerden oluşan sistemlerin benzetimine olanak sağlarken; bileşenlerin eşzamanlı ve paralel evrilmesi ise benzetim hızını artırmaktadır.

2 JuSDL ile modelleme ve benzetim

2.1 Modelleme

Geliştirilen benzetim ortamında işaret-akışı yaklaşımı benimsenmiştir [19]. Bu yaklaşımda model bileşenlerden ve onları birbirine bağlayan bağlantılardan oluşur (Şekil 1).



Şekil 1. Bir örnek modelin blok şeması. B1, ..., B5 model bileşenleri L1, ..., L6 model bağlantılarıdır. T modelin zaman referansıdır. Bileşenlerin altında gösterilen T bağlantısı bileşenlerin tetikleme bağlantılarıdır.

Figure 1. Block diagram of an example model. B1, ..,. B5 are the components and L1, ..., L6 are the connections of the model. T is the common time reference of the model. The links T shown under the components are the trigger links of the components.

Bileşenler veri işleyen birimlerdir ve bilginin nasıl işleneceği bileşen davranışı tarafından belirlenir. Bileşen davranışı bileşenin modellenmesinde kullanılan fiziksel büyüklüklerin uyması gereken fiziksel yasalar sonucu elde edilen matematiksel denklemler ile tanımlanır. Sistemin doğasına ve yapılan modellemeye göre bu denklemler, sürekli ya da ayrık zaman değişkene sahip olma, türevli ifade içerip içermeme vb. çeşitlilikler gösterebilir. Model bileşenlerinin birbirleri ile etkileşimi giriş ve çıkış bağlantıları üzerinden gerçekleştirilir. Bağlantıların üzerindeki veri akışı tek yönlüdür. Bileşenler giriş bağlantısına veri yazan diğer bileşenler tarafından sürülür.

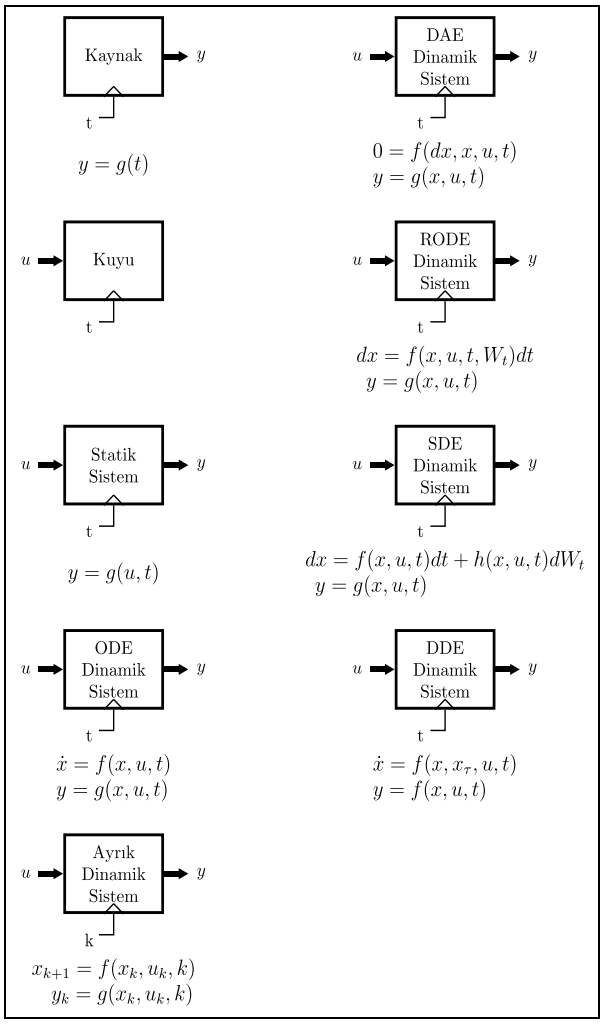
Model benzetimi bileşenlerin eşzamanlı olarak evrilmesi ile gerçekleştirilir. Bu amaçla model bileşenleri için bir ortak zaman referansı kullanılmıştır. Zaman referansından benzetim adım aralıklarında üretilen zaman tetiklemesi bileşenlerin tetikleme bağlantılarına yazılır. Tetikleme bağlantılarından zaman tetiklemesi alan her bir bileşen giriş bağlantısından veri okur, kendi matematiksel modeline göre çıkışını hesaplar ve çıkış bağlantısına yazar.

2.1.1 Bileşenler

Geliştirilen benzetim ortamındaki bileşen türleri çıkış ve durum denklemleri ile birlikte Şekil 2'de gösterilmektedir. Bileşenler kaynaklar, kuyular ve sistemler olarak gruplandırılabilir. Kaynaklar zamanın bir fonksiyonu şeklinde işaret üreten bileşenlerdir. Zaman tetiklemesi alan bir kaynak bileşeni çıkış denklemine göre çıkışını hesaplar ve çıkış bağlantısına yazar. Çıkışları yalnızca zamana bağlı olduğundan giriş bağlantıları yoktur. Kuyular ise veri işleme birimleridir. Genel amaçları benzetim sırasında bağlantılar üzerinden akan verinin çevrimiçi olarak işlenmesini sağlamaktır. Zaman tetiklemesi alan bir kuyu bileşeni girişindeki veriyi okur ve işler. Örneğin, veri bir grafiksel arayüzde çizilerek görselleştirilebilir, konsola basılarak veri akışı gözlemlenebilir veya veri dosyalarına yazılarak saklanabilir. Bir kuyu bileşeninin veri işleme yeteneği, standart Julia kütüphanesi ya da farklı Julia paketleri kullanılarak kolaylıkla tanımlanıp kuyu bileşenine eklenebilen yeni eklentiler ile amaca yönelik olarak zenginleştirilebilir. Örneğin, veri üzerinden çeşitli değişmezler, spektral özellikler, istatistiksel bilgiler vb. türetilebilir, parametre kestirimi yapılabilir, çeşitli işaret işleme teknikleri uygulanabilir. Geliştirilen benzetim ortamı, belirli çözümlemeler için tanımlanmış kullanıma hazır eklentilerinin, tanımlanacak yeni eklentiler ile artırılmasına izin verecek esneklikte tasarlanmıştır.

Statik sistemlerde çıkış girişe ve zamana bağlı olduğundan bir statik sistem çıkış denklemi ile ifade edilir. Zaman tetiklemesi alan bir statik sistem girişindeki veriyi okur, çıkış denklemine göre çıkışını hesaplar ve çıkış bağlantısına yazar. Dinamik sistemlerde ise sistem davranışı durumlar ile karakterize edilir ve çıkış girişe, önceki duruma ve zamana bağlıdır. Dolayısıyla, bir dinamik sistem çıkış denklemiyle birlikte bir durum denklemi ile ifade edilir. Zaman tetiklemesi alan bir dinamik sistem girişindeki veriyi okur, durum denklemine göre yeni durumunu hesaplar, hesaplanan yeni durum için çıkışını hesaplayıp çıkış bağlantısına yazar. Şekil 2'de gösterildiği gibi geliştirilen benzetim ortamında bir dinamik sistemin durum denklemi adi türevsel(ODE), türevsel cebirsel(DAE), rastgele adi türevsel(RODE), rassal türevsel(SDE) ve gecikmeli(DDE) türevsel denklem biçiminde verilebilir.

Mevcut benzetim ortamlarında, özel olarak kodlama yapılmaksızın, yalnızca adi türevsel denklem ya da türevsel cebirsel denklem durum denklemi ile ifade edilen dinamik sistemlerin benzetimi mümkündür [1]-[13]. Dolayısıyla, gürültü çözümlemesi, gecikme çözümlemesi veya sistem parametrelerinin rastgele değişmesi gibi çözümlemeler bu benzetim ortamlarında kullanıcı tarafından duruma özel olarak kodlanmalıdır. JuSDL bu kısıtlamayı ortadan kaldırmaktadır.



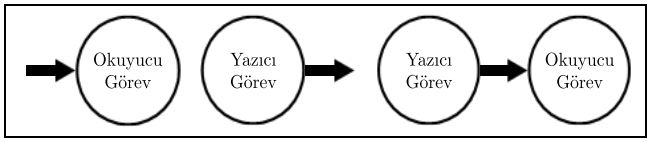
Şekil 2. Bileşen türleri ve ilgili durum ve çıkış denklemleri. u, x, y sırasıyla sistem girişini, durumunu ve çıkışını, t sürekli zamanı, k ayrık zamanı, W_t gürültüye karşılık gelen rassal süreci ve x_τ ise zamanda τ kadar geciktirilmiş x işaretini göstermektedir. g çıkış fonksiyonunu f (ve h) durum fonksiyonlarını göstermektedir.

Figure 2. Component types together with their state and output equations. u, x, y are the input, state and output of the system, respectively. t is continous time and k is descrete time. W_t is the stochastic process corresponding to the noise. x_τ is the state x delayed by τ seconds in time. g and f (and h) are the output and state function(s).

2.1.2 Bağlantılar

Bağlantılar Julia programlama dilindeki kanallar(channel) üzerine inşa edilmiştir. Bağlantılara yazılan her veri bu kanallara gönderilir; bağlantılardan okunan her veri ise bu kanallardan alınır. Kanallar üzerinden veri akışının sağlanabilmesi için kanallara bağlı aktif Julia görevleri (task) gereklidir. Julia görevleri, hesaplamaların işletim sisteminin görev zamanlayıcısı ile doğrudan iletişime geçmeden esnek bir şekilde askıya alınmasını ve sürdürülmesini sağlayan bir kontrol akışı özelliğidir [16]. Görevler arasındaki iletişim ve veri alış-verişi bağlı oldukları Julia kanalları üzerinden gerçekleştirilir.

Şekil 3'te yalnızca okunabilir, yalnızca yazılabilir ve hem yazılabilir hem okunabilir kanal elde etmek için açılması gereken görevler sembolik olarak gösterilmiştir. Yazıcı ve okuyucu görev sırasıyla kanala veri yazan ve veri okuyan görevdir. Kanalın bir ucundan veri okuyabilmek için kanalın diğer ucunda yazıcı görev; bir ucundan veri yazabilmek için ise kanalın diğer ucunda okuyucu görev gereklidir. Eğer okuma sırasında kanala henüz bir veri yazılmamışsa okuma gerçekleşmez ve veri yazılana kadar beklenir. Benzer şekilde, kanala veri yazma işlemi sırasında henüz kanaldan veri okunmamışsa yazma işlemi gerçekleşmez ve veri okunana kadar beklenir. Benimsenen modelleme yaklaşımında bir bileşen diğer bileşeni sürer. Bileşenleri birbirine bağlayan bağlantılara süren bileşenler veri yazar; sürülen bileşenler ise bu bağlantılardan veri okur. Dolayısı ile model bağlantılarının hem okunabilir hem yazılabilir olması gereklidir. Bu durum, modelde bir ucu herhangi bir bileşene bağlanmamış bir bağlantının olmamasını gerektirir. Aksi takdirde benzetim sırasında bir bileşene bağlanmayan bir kanal üzerinden veri okuma veya yazma sırasında benzetim sonlanmaz. Benzetim sırasında bağlantılar amaca yönelik istenildiği gibi düzenlenebilir, bağlantının kazancı değiştirilebilir, yeni bağlantı eklenebilir ya da var olan bir bağlantı koparılabilir. Başka bir deyişle sistem dinamik olarak değişebilir. Bu da, birden fazla düğümden oluşan sistem ağlarında ağ topolojisinin ve bağlantı şiddetlerinin ağın davranışını nasıl etkilediğinin gözlemlenebilmesi anlamında çeşitli çalışmaların yapılabilmesi veya arızaların modellenmesi imkânını sağlar.



(a) (b) (c)

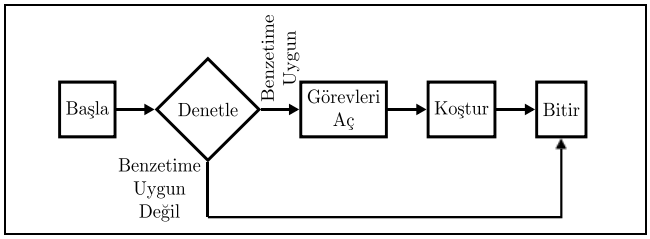
Şekil 3. (a): Okunabilir bağlantı. (b): Yazılabilir bağlantı. (c): Hem yazılabilir hem okunabilir bağlantı.

Figure 3. (a) Readable connection (b) Writable connection (c) Readable and writable connection.

2.2 Benzetim

Benzetimi yapılacak bir model birbirine bağlı bileşenlerden ve bir zaman referansından oluşur. Zaman referansı, model bileşenlerinin çıkışlarını örneklemek ve bileşenleri tetiklemek için kullanılır. Benzetim, zaman referansının örnekleme zaman aralıklarında ürettiği zaman darbeleriyle bileşenleri tetiklemesi ile yapılır. Bileşenler zaman referansının her tetiklemesinde kendilerini evirerek çıkışlarını hesaplar ve çıkış bağlantılarına yazarlar.

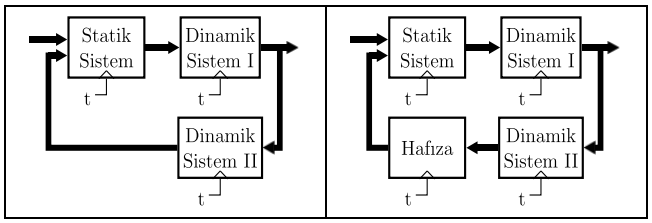
Benzetim aşamaları Şekil 4'teki akış şemasında gösterilmiştir. Benzetimin tüm aşamalarının gerçekleştirilmesi, denetlenmesi ve raporlanması herhangi bir kullanıcı müdahalesi gerektirmeden otomatik olarak yapılmaktadır.



Şekil 4. Benzetim akış şeması.

Figure 4. Flowchart of the simulation process.

İlk aşamada model bir veya iki ucu bağlanmamış bağlantıların olup olmadığı anlamında denetlenir. Eğer modelde en az bir ucu bağlanmamış bir bağlantı tespit edilirse benzetim bu aşamada sonlandırılır. Modelin benzetime uygun olmadığı diğer bir durum cebirsel döngülerin var olduğu durumdur. Örneğin, hemen her geri beslemeli sistem modeli cebirsel döngülere örnek teşkil etmektedir. Bir cebirsel döngü çıkışları girişlerine bağlı olarak hesaplanan bir veya birden fazla bileşenden oluşan bir kapalı çevrimdir (Şekil 5). Benzetim sırasında döngüdeki bileşenlerin hiçbiri döngüyü kıracak şekilde çıkış üretemediği için benzetim sonlanmaz. Bu tip bir problem modelin cebirsel döngü içermeyecek şekilde yeniden düzenlenmesiyle, döngünün ileribesleme(feedforward) cebirsel denkleminin çözülmesi ile veya döngü üzerinde döngüyü kıracak şekilde belirli başlangıç koşuluna sahip bir hafıza bileşenin eklenmesiyle kırılabilir. Denetleme adımının modelin cebirsel döngüler içerip içermemesinin tespiti ve eğer varsa bu döngülerin(ileribesleme cebirsel denklemlerinin çözülmesi veya hafıza bileşenlerinin kullanılması yoluyla) kırılması anlamında genişletilmesi mümkündür [20].



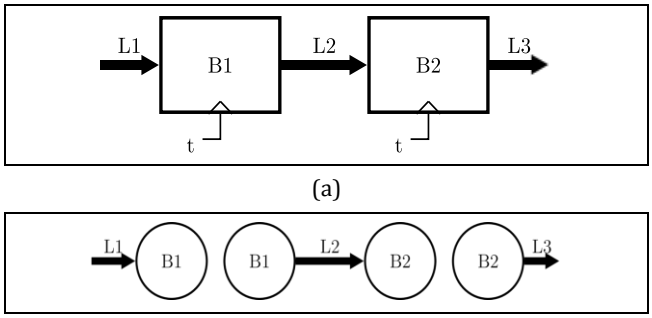
(a) (b)

Şekil 5. (a) 'Statik Sistem', 'Dinamik Sistem I' ve 'Dinamik Sistem II' bileşenlerinden oluşan bir cebirsel döngü. (b): Cebirsel döngünün bir hafıza bileşeni ile kırılması.

Figure 5. (a) An algebraic loop consisting of 'Static Sistem', 'Dinamik Sistem I' and 'Dinamik Sistem II'. (b) Breaking the algebraic loop with a memory component.

Denetleme aşamasının olumlu sonuçlanması durumunda model bağlantılarından veri akışının sağlanabilmesi için gerekli görevler açılır. Bu noktada her bir bağlantıya bir yazıcı ve bir okuyucu görev bağlanır. Örneğin, Şekil 6a'da B1, B2 bileşenlerinden ve L1, L2, L3 bağlantılarından oluşan örnek bir model bölümü gösterilmiştir. Benzetim süresince tetikleme bağlantısından zaman tetiklemesi alan B1 bileşeni L1 bağlantısından veri okur, çıkışını hesaplar ve L2 bağlantısına yazar. Benzer şekilde tetikleme bağlantısından zaman tetiklemesi alan B2 bileşeni L2 bağlantısından veri okur, çıkışını hesaplar ve L3 bağlantısına yazar. B1 ve B2 bileşenlerinin L1, L2 ve L3 bağlantılarına bağladığı görevler Şekil 5b'de gösterilmiştir. B1 bileşeni L1 bağlantısından veri okuduğu için L1 bağlantısına okuyucu görev, L2 bağlantısına veri yazdığı için L2 bağlantısına yazıcı görev açılır. Benzer şekilde, B2 bileşeni L2 bağlantısından veri okuduğu için L2 bağlantısına okuyucu görev, L3 bağlantısına veri yazdığı için L3 bağlantısına yazıcı görev açılır. L2 bağlantısına hem yazıcı hem okuyucu görevi açıldığından bu bağlantı üzerinden B1 bileşeninden B2 bileşenine veri akışı sağlanmış olur. Görev açma aşamasında açılan görevlerin benzetim boyunca aktif olduklarının kontrolü ve bir hata ile karşılaşılması durumunda ilgili hatanın raporlanması için bir görev yöneticisi oluşturulur.

Görev açma aşamasını koşturma aşaması takip eder. Görev açma aşamasında bileşenlere karşılık açılan görevler bileşenlerin tetikleme bağlantıları üzerinden zaman tetiklemelerini almalarını beklerler. Bu tetiklemeler koşturma aşamasında model zaman referansı tarafından benzetimin örnekleme aralıklarında üretilir. Örneklemelerin eşit veya birbirinden bağımsız zaman aralıklarında yapılabilmesi mümkündür. Üretilen tetiklemeler bileşenlerin tetikleme bağlantılarına iletilir. Zaman tetiklemesi alan bir bileşene karşılık gelen görev ilgili bileşenin girişinden veri okumak, bileşenin çıkışını hesaplamak ve çıkış bağlantısına yazmak olarak tanımlanmıştır. Koşturma aşaması zaman referansının başlangıç zamanında başlar ve benzetim örnekleme zaman aralıklarında devam ettirilerek zaman referansının bitiş zamanında sonlandırılır.



(a)

(b)

Şekil 6. Bağlantılara görev açılması. (a): B1, B2 bileşenlerinden ve L1, L2, L3 bağlantılarından oluşan model bölümü. (b): (a)'da verilen bağlantılar için açılan görevler.

Figure 6. Launching tasks for the connections. (a): A part of a simple model consisting of B1, B2 components and L1, L2, L3 connections. (b): Tasks launched corresponding to the links in (a).

Örnekleme minimum zaman aralığı benzetimi yapılacak sistemlerin zaman sabitleri dikkate alınarak belirlenmelidir. Birbirinden çok farklı şekilde evrilen sistemler birbirine bağlanabileceğinden genel geçer bir adım aralığı hesaplama algoritması belirlemek güçtür. Benzetim örnekleme zaman aralıkları kullanıcı tarafından belirlense de, bir örnekleme anından diğerine doğru sistemler evrilirken sistemlerin çözücüleri uyarlamalı (adaptive) adım aralıkları atmaktadır. Dolayısıyla, kullanıcı tarafından belirlenen örnekleme zaman aralıkları sistemlerin eşit zaman aralıkları ile evrilmesine değil, sadece sistemlerin çıkışlarının belirtilen anlarda örneklenmesine zorlamaktadır.

Matematiksel modellerinin gerektirdiği hesaplama karmaşıklığına bağlı olarak bileşenlerin evrilmeleri birbirinden farklı süreler alabilir. Bileşenlerin evrilmesi sırasında benzetimin eşzamanlılığının bozulması bağlantılar modellenirken Julia kanallarının kullanılması ile ortadan kaldırılmıştır.

Koşturma aşamasının tamamlanmasından sonra görev açma aşamasında açılan görevler kapatılır ve benzetim sonlandırılır.

Mevcut benzetim ortamlarında sistemler her ne kadar alt bileşenlerine ayrılsa da benzetim için modeli bir bütün halinde ifade eden tek bir tümleşik matematiksel denklem elde edilir ve bu denklem tüm benzetim süresi için tek seferde çözülür [19]. Geliştirilen benzetim ortamında ise sistemler yine alt bileşenlere ayrılır fakat tek bir tümleşik denklem elde edilmez. Bunun yerine alt bileşenler kendi matematiksel denklemlerini çözerek bireysel evrilir. Ayrıca, bileşenler tek seferde evrilmez; benzetim örnekleme aralıklarında paralel olarak evrilir ki bu da farklı türde matematiksel denklemlerle ifade edilen bileşenler içeren sistemlerin(örneğin hem sürekli hem de ayrık zamanlı

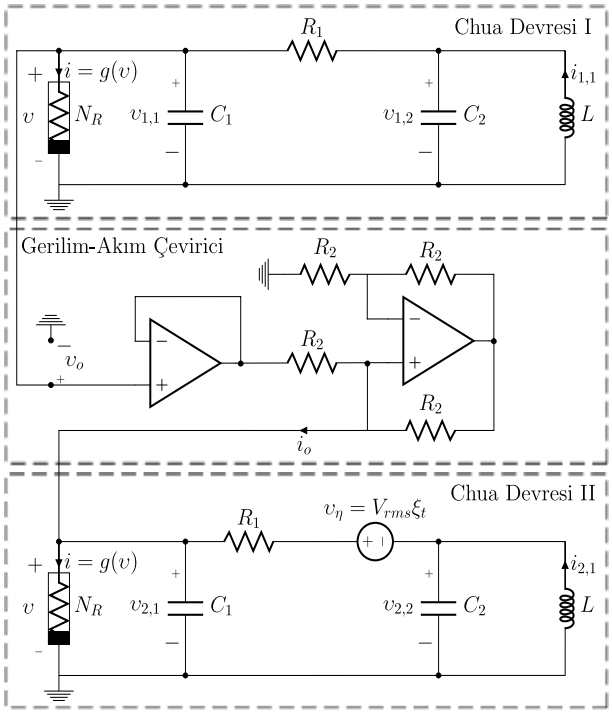
davranışlar gösteren hibrit sistemler [21]) gerçek zamanlı benzetimlerini mümkün kılabilmektedir. Gerekli olduğunda paralel işlemeden feragat edilerek sistemin tek bir bileşen olarak tanımlanabilmesi her zaman mümkündür.

Benzetim boyunca modelin bağlantıları üzerinde akan sürekli zamanlı işaretler benzetimin örnekleme anlarında örneklenir. Örneklenen değerler örnekleme zaman aralığı boyunca aradeğerlenerek bileşenlerin evrilmesi sağlanır. Örnekleme zaman aralıkları benzetim sonuçlarının hassasiyetini doğrudan etkileyen bir faktördür.

3 Açıklayıcı örnekler

3.1 Farklı türden denklemler içeren model benzetimi

JuSDL ile farklı tipte matematiksel denklemler ile verilen bileşenlerin birlikte benzetimleri mümkündür. Örneğin klasik adi diferansiyel denklem ve gürültü modellemek için rassal diferansiyel denklem içeren bir model Şekil 7'de verilmiştir.



Şekil 7. Birbirine bağlı iki Chua devresi.

Figure 7. Two coupled Chua circuits.

İkinci Chua devresinde v_η, R_1 direnci üzerindeki termal gürültüyü modelleyen gerilim kaynağıdır ve etkin değeri $V_{rms} = \sqrt{4kR_1TB}$ olarak bulunur[22]. Burada k Boltzmann sabiti, $T > 0$ Kelvin biriminden sıcaklık ve B bant genişliğidir. Birinci Chua devresinde ise R_1 direnci üzerindeki termal gürültü ihmal edilmiştir. İki devre birbirine bir gerilim-akım çevirici devresi ile bağlanmış ve birinci Chua devresinin C_1 sığacı üzerindeki gerilim akıma çevrilip ikinci Chua devresinin C_1 sığacı üzerinde akan akıma bağlanmıştır. N_R

$$g(v) = \begin{cases} m_0 v + m_1 - m_0 & -B_p \geq v \\ m_0 v & -B_p \leq v \leq B_p \\ m_0 v + m_0 - m_1 & B_p \leq v \end{cases} \quad (1)$$

akım-gerilim karakteristiğine sahip doğrusal olmayan direnç, ve ξ_t sıfır ortalamalı birim varyanslı beyaz Gauss gürültüsüdür. Şekil 7'de verilen sistem

$$\begin{array}{lll} x_j = v_{j,1}/B_p & y_j = v_{j,2}/B_p & z_j = i_{j,1}\,R/B_p \\ \tau = t\,/(RC_2) & a = m_1 R & b = m_0 R \qquad j = 1,2 \\ \alpha = C_2/C_1 & \beta = C_2 R^2/L & \epsilon = 1/(R_1 C_1) \end{array} \quad (2)$$

dönüşümleri yapılırsa, birinci Chua devresi

$$\begin{aligned} \dot{x}_1 &= \alpha(x_1 - y_1 - f(x_1)) \\ \dot{y}_1 &= x_1 - y_1 + z_1 \\ \dot{z}_1 &= -\beta y_1 \end{aligned} \quad (3)$$

ile verilen adi türevsel denklemle[23], ikinci Chua devresi

$$\begin{aligned} dx_2 &= \alpha(x_2 - y_2 - f(x_2) + \epsilon x_1)d\tau - \eta dW_\tau \\ dy_2 &= (x_2 - y_2 + z_2)d\tau + \eta dW_\tau \\ dz_2 &= -\beta y_2 d\tau + 0\, dW_\tau \end{aligned} \quad (4)$$

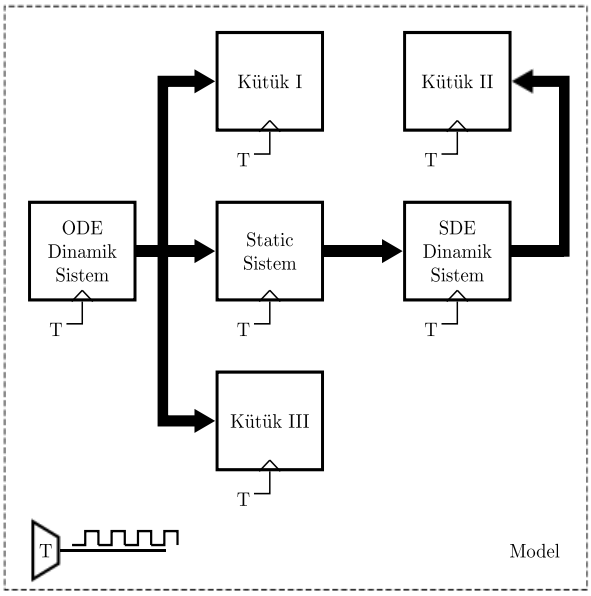
ile verilen rassal türevsel denklemle[24] ve gerilim-akım çevirici devresi

$$i_o = \epsilon v_o \quad (5)$$

ile verilen cebirsel denklemle modellenebilir. Burada, x_1, y_1, z_1 ve x_2, y_2, z_2, sırasıyla, birinci ve ikinci Chua sisteminin durum değişkenlerini gösterirken i_o ve v_o gerilim-akım çevirici sistemin çıkış ve giriş değişkenini göstermektedir. $f(x) = bx + 1/2(a-b)(|x+1| - |x-1|)$, $\eta = V_{rms}/RC_1$ ve W_τ, ξ_τ beyaz gürültüsüne karşılık gelen Wiener rassal sürecini göstermektedir.

Şekil 7'deki devre modeli adi ve rassal türevsel denklemler ve cebirsel denklem olmak üzere farklı türden matematiksel denklemler ile verilen bileşenlere sahiptir. Model bileşenlerinin bireysel evrilmesi bu türden farklı türde matematiksel denklemler ile ifade edilen bileşenlere sahip bir modelin benzetimini mümkün kılmaktadır.

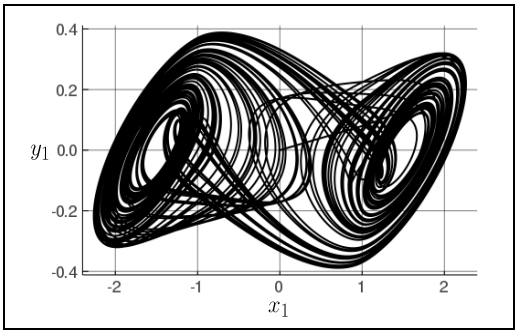
JuSDL metin tabanlı bir kullanıcı arayüzüne sahiptir. Bir metin dosyasında JuSDL uygulama programlama arayüzü (API) ile model kodlanır ve benzetimi yapılır. Şekil 7'deki sistemin işaret akışı yaklaşımına göre oluşturulmuş modeli Şekil 8'de ve geliştirilen benzetim ortamında yazılan program Betik 1'de verilmiştir.



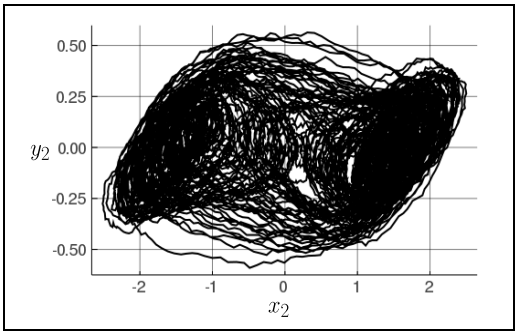
Şekil 8. Şekil 7'de verilen devrenin modeli.

Figure 8. The *model corresponding to the circuit in Figure 7.*

Sırasıyla, Eşitlik 3 ve 4'te verilen türevsel denkleme karşılık gelen 'ODE Dinamik Sistem' ve 'SDE Dinamik Sistem' ve Eşitlik 5'te cebirsel denkleme karşılık gelen 'Statik Sistem' oluşturulmuştur. Dinamik sistemlerin çıkışlarının saklanması amacıyla 'Kütük I' ve 'Kütük II' oluşturulmuştur. 'ODE Dinamik Sistem'in en büyük Lyapunov üstelini[25] sistemin birinci çıkışı üzerinde akan veriden çevrimiçi olarak hesaplanıp saklanması amacıyla 'Kütük III' oluşturulmuştur. Benzetim sırasında modelin bağlantıları üzerinde akan veriden herhangi bir değer türetmeyecekleri için 'Kütük I' ve 'Kütük II'ye bir eklenti eklenmemiştir. 'Kütük III'e ise en büyük Lyapunov üstelinin hesaplanabilmesi için uygun eklenti eklenmiştir. Bileşenler oluşturulduktan sonra birbirine uygun şekilde bağlanıp model oluşturulmuş ve benzetim yapılmıştır. Benzetim sonunda 'Kütük I' ve 'Kütük II'de saklanan veri Şekil 9'da çizilmiş ve 'Kütük III'de saklanan en büyük Lyapunov üstelleri 2.62, 0.90, 1.83, 0.93 2.47 olarak okunmuştur. Şekil 9b'de verilen yörüngede direnç üzerindeki termal gürültünün varlığı ayırt edilebilmektedir.



(a)



(b)

Şekil 9. Şekil 7'de verilen devrenin benzetim sonuçları. (a): 'ODE Dinamik Sistem' in $x_1 - y_1$ yörüngesi. (b): 'SDE Dinamik Sistem'in $x_2 - y_2$ yörüngesi.

Figure 9. Simulations results of the circuit in Figure 7. (a) $x_1 - y_1$ trajectory of 'ODE Dinamik Sistem'. (b) $x_2 - y_2$ trajectory of 'SDE Dinamik Sistem'.

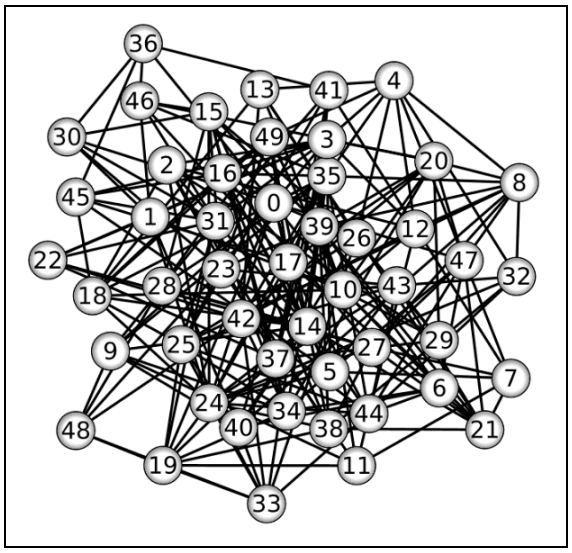
Geliştirilen yazılımda tanımlayıcı sistem dili kullanılmıştır. Model bir bütün halinde tek seferde tüm bileşenlerinin davranışlarını içerecek şekilde oluşturulmak yerine, Betik 1'de verilen programda gösterildiği gibi, bileşenlerinin ayrı ayrı oluşturulması ve onların birbirine nasıl bağlandıklarının belirtilmesi ile oluşturulmuştur. Bileşenlerin çıkışlarının bağlı olduğu diğer bileşenler takip edilerek model yapısı çıkarılabilir. Model oluşturulması sırasında bileşenler model üzerindeki veri akışı yönünde verilme zorunluluğu olmadan gelişigüzel sırada verilebilir. Bu durum çok sayıda bileşenden oluşan ve karmaşık topolojiye sahip olan büyük sistem ağlarının modellerinin oluşturulmasında kolaylık sağlar.

Betik 1. Şekil 8'de verilen sistemin benzetimi için geliştirilen benzetim ortamında yazılmış program.

```
using Jusdl
import Jusdl.Plugins.Lyapunov
# Benzetim zaman ayarları
t0,dt,tf = 0, 0.001, 250.

# `ODE Dinamik Sistem` 'in oluşturulması
f(x,a=-8/7,b=-5/7)=b*x+1/2*(a-b)*(abs(x+1)-abs(x-1))
function sf1(dx,x,u,t,α=15.6,β=28,f=f)
  dx[1]=α*(x[2]-x[1]-f(x[1]))
  dx[2]=x[1]-x[2]+x[3]
  dx[3]=-β*x[2]
end
of1(x,u,t)=x
x0,t=rand(3)*1e-3,0.
odeds=ODESystem(nothing,Bus(3),sf1,of1,x0,t)

# `SDE Dinamik Sistem` 'in oluşturulması
function sf2(dx,x,u,t,α=15.6,β=28,f=f)
  dx[1]=α*(x[2]-x[1]-f(x[1]))+u[1](t)
  dx[2]=x[1]-x[2]+x[3]
  dx[3]=-β*x[2]
end
df2(dx,x,u,t,η=0.1)=(dx.=[-η,η,0])
of2(x,u,t)=x
x0,t=rand(3)*1e-3,0.
sdeds=SDESystem(Bus(3),Bus(3),(sf2,df2),of2,x0,t)

# `Statik Sistem` 'in oluşturulması
gain = Gain(Bus(3),gain=[0.01 0 0; 0 0 0; 0 0 0])

# `Kütük I, II, III` 'ün oluşturulması
plg=Lyapunov(ts=dt,m=7,J=11,ni=200)
writer1=Writer(Bus(3),buflen=5000,plugin=nothing)
writer2=Writer(Bus(3),buflen=5000,plugin=nothing)
writer3=Writer(Bus(1),buflen=5000,plugin=plg)

# Bileşenlerin bağlanması
connect(odeds.output,gain.input)
connect(gain.output,sdeds.input)
connect(odeds.output,writer1.input)
connect(odeds.output[1],writer3.input)
connect(sdeds.output,writer2.input)

# Modelin oluşturulması
model=Model(odeds,sdeds,gain,writer1,writer3,writer2)

# Benzetimin yapılması
sim=simulate(model,t0,dt,tf)

# Kütüklerde saklanan verilerin okunması ve çizilmesi.
using Plots
t,x1=read(writer1,flatten=true)
t,x2=read(writer2,flatten=true)
t,x3=read(writer3,flatten=true)
plot(x1[:,1],x1[:,2])
plot(x2[:,1],x2[:,2])
```

3.2 Çok sayida bileşen içeren model benzetimi

Geliştirilen benzetim ortamında çok sayıda bileşenden oluşan büyük ölçekli karmaşık topolojiye sahip sistem ağlarının modelleri kolaylıkla oluşturulabilmekte ve amaca yönelik benzetimleri yapılabilmektedir. Bu duruma örnek olarak birbirine bağlı sürekli zamanlı dinamik sistem düğümlerinden oluşan bir ağ Şekil 10'da verilmiştir.



Şekil 10. Özdeş sistemlerden oluşan dinamik sistem ağı. Şekildeki daireler ağın dinamik sistem düğümlerini, daireleri birbirine bağlayan doğru parçaları ağın düğümleri arasındaki bağlantıları temsil etmektedir.

Figure 10. A network consisting of identical dynamical system nodes . The spheres and the line segments represent the dynamical system nodes and the connections between them, respectively.

Ağın düğümleri,

$$u_i = \sum_{i=1}^{n} \epsilon_{ij} P x_j\,, \quad i = 1, \dots, n \tag{6}$$

olmak üzere

$$\dot{x}_i = f(x_i) + u_i, \quad i = 1, \dots, n \tag{7}$$

ile verilen bir adi türevsel denklem takımı ile evrilmektedir. Burada, $n = 50$ ağdaki düğüm sayısı, $x_i = [x_{i,1},\ x_{i,2},\ x_{i,3}] \in \mathbb{R}^3$ i No.lu düğümün durum vektörü, $f: \mathbb{R}^3 \mapsto \mathbb{R}^3$, $\sigma = 10, \beta = 8/3, \rho = 28$ olmak üzere

$$\begin{aligned} \dot{x}_{i,1} &= \sigma(x_{i,2} - x_{i,1}) \\ \dot{x}_{i,2} &= x_{i,1}(\rho - x_{i,3}) - x_{i,2} \\ \dot{x}_{i,3} &= x_{i,1}x_{i,2} - \beta x_{i,3} \end{aligned} \tag{8}$$

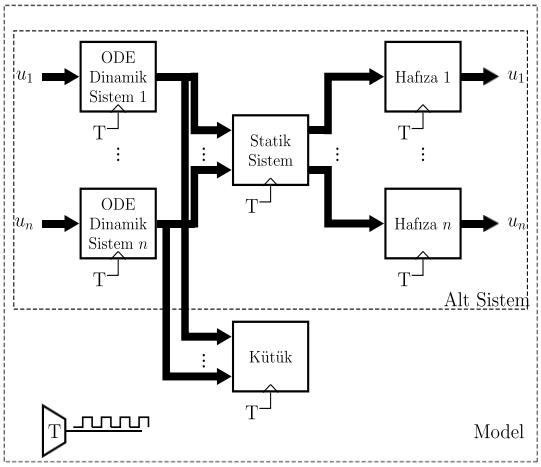
ile verilen bireysel düğüm dinamiğini belirleyen fonksiyon, $P = diag(1,0,0)$ düğümlerin $x_{i,1}$ durum değişkenleri üzerinden birbirine bağlandıklarını belirten matristir. $E = [\epsilon_{ij}] \in \mathbb{R}^{n \times n}$, $\epsilon_{ij} = \epsilon_{ji} > 0$ olmak üzere, ağ topolojisini ve düğümler arasındaki bağ şiddetlerini belirler: i ve j numaralı düğümler arasında bağlantı olması durumunda $\epsilon_{ij} = \epsilon_{ji} = \epsilon_0 > 0$ olurken, aksi durumda $\epsilon_{ij} = \epsilon_{ji} = 0$ olur. Ağ, düğüm çiftlerinin birbirine bağlanma olasılığı p=0.2 olmak üzere Erdos-Renyi topolojisine sahiptir [26].

Bir model bileşeninin başka bileşenlerden oluşan bir alt sistem olması da mümkündür. Benzetimin yapılabilmesi için modelin tüm bileşenlerinin birbirine bağlı olması bir ön gereklilik olduğundan, alt sistemi oluşturan bileşenlerin de birbirine bağlı olması gereklidir. Bileşenlerden bir alt sistem oluşturulurken, istenilen bileşenlerin giriş çıkış bağlantıları alt sistemin giriş çıkış bağlantıları olarak belirlenebilir.

Şekil 10'daki dinamik sistem ağının modeli Şekil 11'de verilmiştir. 'ODE Dinamik Sistem k', $k = 1, \dots, n$, ağın dinamik sistem düğümlerine karşılık gelmektedir. Eşitlik 6'da görüldüğü gibi ağın düğümlerinin durumları ağırlıklandırılarak birbirinin girişlerine geri besleme olarak verilmiştir. Geri besleme sonucu oluşan cebirsel döngülerin kırılması amacıyla 'Hafıza k', $k = 1, \dots, n$, bileşenleri; sistem durumlarının ağırlıklandırılması amacıyla 'Statik Sistem' oluşturulmuştur. Ağ dinamik ve statik sistemlerden oluşan bir alt sistem olarak modellenmiş tüm düğümlerin durumları alt sistemin çıkışları olarak belirlenmiştir.

Betik 2. Şekil 10'de verilen sistemin benzetimi için geliştirilen benzetim ortamında yazılmış program.

```
using Jusdl

# Benzetim zaman ayarları
t0,dt,tf = 0,0.005,100

# Ağın oluşturulması
function constructds(input,output)
  function sf(dx,x,u,t,α=10, β=8/3, ρ=28)
    dx[1]=α*(x[2]-x[1])
    dx[2]=x[1]*(ρ-x[3])-x[2]
    dx[3]=x[1]*x[2]-β*x[3]
  end
  of(x,u,t)=x
  x0,t=rand(3)*1e-3,0.
  return ODESystem(input,output,sf,of,x0,t)
end
n,d=50,3
nodes=[constructds(nothing,Bus(d)) for i=1:n]
ϵ0=20
E=topology(:erdos_renyi,n,0.2,weight=ϵ0)
P=[1 0 0; 0 0 0; 0 0 0]
net=Network(nodes,E,P,outputnodeidx=1:n)

# Kütük oluşturulması
writer=Writer(Bus(length(net.output)))

# Model bileşenlerinin bağlanması
connect(net.output, writer.input)

# Modelin oluşturulması
model=Model(net, writer)

# Benzetimin yapılması
sim=simulate(model,t0,dt,tf)

# Benzetim verisinin okunması ve hata hesaplanması
t,x=read(writer)
err=sqrt.(sum(diff(x[:,1:d:end],dims=2).^2/n,dims=2))

# Benzetim verisinin çizilmesi.
using Plots
plot(t,x[:,1])
plot(x[:,1],x[:,2])
plot(t[1:1500],err[1:1500])
```



Şekil 11. Şekil 10'da verilen sistemin modeli.

Figure 11. The model of the system in Figure 10.

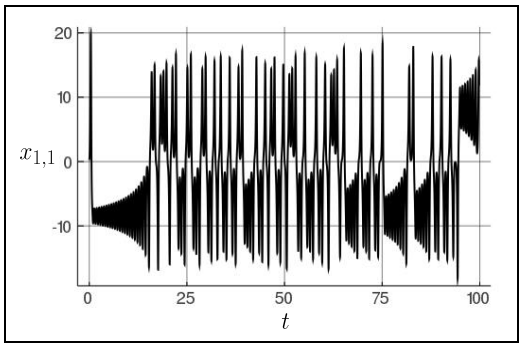
Şekil 10'da verilen ağın benzetimi için geliştirilen benzetim ortamında yazılan program Betik 2'de verilmiştir. Programda belirlenen parametrelere göre ağ ve bir kütük oluşturulmuştur. Ağ oluşturulurken tüm düğümlerin durumları çıkış olarak alınmıştır. Ağın çıkışı kütüğe bağlanmış, model oluşturulmuş ve 0.005 saniye örnekleme periyodu ile 0-100 sn. aralığında benzetimi yapılmıştır. Benzetim sonunda, kütükte saklanan veri okunmuş, ağın birinci düğümü ile diğer düğümleri arasındaki hata karesinin ortalaması (MSE) hesaplanmış ve çizilmiştir. MSE

$$MSE(t) = \frac{1}{n}\sum_{i=2}^{n} \|x_i(t) - x_1(t)\|^2 \qquad (9)$$

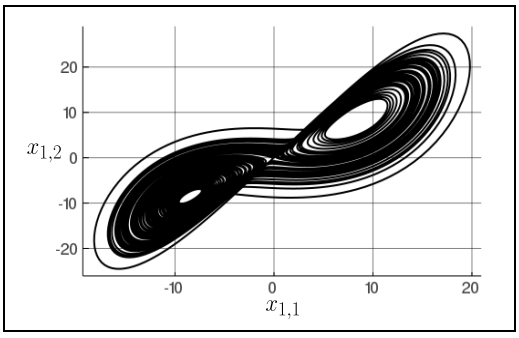
olarak hesaplanmaktadır. Şekil 12c'den MSE'nin zamanla sıfıra doğru gittiği, dolayısıyla ağın tüm düğümlerinin birbiri ile eşzamanlı olduğu gözlenebilmektedir.

Betik 2'de verilen programda görüldüğü gibi, geliştirilen benzetim ortamının tanımlayıcı sistem dilinin Şekil 10'daki gibi çok sayıda dinamik sistem düğümlerinden oluşan karmaşık topolojiye sahip ağların modellerinin hızlıca oluşturulması ve benzetimlerinin yapılması açısından önemlidir. Geliştirilen benzetim ortamında, böyle bir ağ, düğümlerinin tek tek oluşturulması ve düğümlerin birbirlerine teker teker bağlanabilmesiyle oluşturulacağı gibi, ağın düğümlerinin birbirine nasıl ve ne şiddete bağlandığını belirten bir matris ile kolaylıkla ve hızlıca oluşturulabilir.

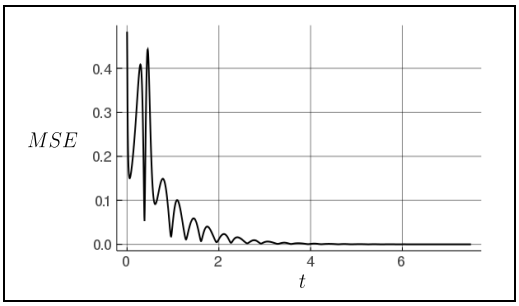
JuSDL'i ve mevcut benzetim ortamlarını hız performansları yönünden kıyaslamak amacıyla Eşitlik (6)-(8) ile tanımlanan ve her bir düğümü diğer tüm düğümlere bağlı olan dinamik sistem ağının benzetimi 0-10 sn. boyunca 0.01 örnekleme zaman aralığı ile yapılmıştır. Benzetimler Intel© Core™ i7-7700HQ CPU@2.80GHz işlemci, 16GB 2400 MHz DDR4 RAM ve Linux işletim sistemine sahip bilgisayar üzerinde JuSDL(v0.1.1), MATLAB™(v9.6)[27] ve OpenModelica™(v1.14.1)[28] ortamlarında yapılmış ve bu ortamlarda sırasıyla 'DP5', 'ode45' ve 'runge_kutta' sayısal diferansiyel denklem çözücü algoritmaları kullanılmıştır. Kullanılan sayısal diferansiyel denklem çözücü algoritmaları hesaplama karmaşası açısından birbirlerine mümkün olduğunca yakın algoritmalar olacak şekilde seçilmiştir. Kıyaslamalarda tutarlı sonuçlar elde etmek amacıyla benzetimlerin koşacağı bilgisayar üzerinde benzetim performanslarına etki edebilecek gürültü kaynakları mümkün olduğunca azaltılmıştır. Tüm benzetimler tek bir işlemci çekirdeği üzerinde herhangi bir paralelleme yapılmadan 25 kez tekrarlanmış ve elde edilen CPU sürelerinin istatistiksel değerleri Tablo 1'de verilmiştir. CPU süresi, işlemcinin bir programın çalışması sırasında gerçekleştirdiği işler için harcadığı sürenin ve program adına sistem çağrıları yapmak için geçirdiği sürenin toplamıdır. Tablo 1'de benzetimlerin derleme (compilation) ve yorumlama(interpretation) süreleri çıkarılmış ve yalnızca koşma süreleri(runtime) verilmiştir. Kıyaslama için ortalama CPU süresi baz alındığında en iyi performans sıralaması JuSDL, OpenModelica™ ve MATLAB™ olarak görülmektedir. JIT(Just-in-Time) derleyicisinin her derleme için farklı şekilde kod üreterek JuSDL'in çalışma performansının standart sapması üzerindeki doğrudan etkisi tablodan gözlemlenebilmektedir.



(a)



(b)



(c)

Şekil 12. Şekil 10'da verilen dinamik sistem ağına ait benzetim sonuçları. (a): Birinci düğümün $x_{1,1}$ durum değişkeninin zaman dalga şekli. (b): Birinci düğümün$x_{1,1} - x_{1,2}$ yörüngesi. (c): Eşzamanlılık performansı.

Figure 12. Simulations results of the dynamical systems network given in Figure 10. (a) Time wavefom of the state variable $x_{1,1}$ of the first node. (b) $x_{1,1} - x_{1,2}$ trajectory of the first node. (c) Synchronization performance.

Tablo 1. JuSDL, MATLAB™ ve OpenModelica™ kıyaslama sonuçları.

Table 1. Benchmark results of JuSDL, MATLAB™ *and* OpenModelica™

Benzetim Ortamı	Ortalama (s)	Standart Sapma (s)	Minimum (s)	Medyan(s)	Maksimum(s)
JuSDL	0.433	0.061	0.160	0.440	0.490
MATLAB™	1.352	0.065	1.100	1.350	1.430
"OpenModelica™	0.636	0.005	0.630	0.640	0.640

4 Sonuçlar

Bu çalışmada JuSDL tanıtılmıştır. JuSDL'de benzetim, bileşenlerin bireysel olarak benzetimin örnekleme aralıklarında kendi matematiksel denklemlerine göre eşzamanlı evrilmesi ile yapılır. Bileşenler statik ya da dinamik olabilir veya sürekli veya ayrık zamanlı zaman değişkenine sahip matematiksel denklemler ile ifade edilebilir. Adi, cebirsel, rastgele adi, rassal ve gecikmeli türevsel denklemler veya ayrık fark denklemleriyle ifade edilen bileşenlerin oluşturduğu modellerin benzetimine eşzamanlı olarak olanak sağlanmaktadır. Model bileşenlerinin tamamı aynı türden matematiksel denklemler ile ifade edilmek zorunda değildir.

Benzetim sırasında bağlantılar üzerinden akan veri doğrudan kayıt edilebilmekte veya görselleştirilebilmektedir. Çevrimdışı analizlere ek olarak kullanıcı tanımlı eklentilerle çevrimiçi veri analizi yapmak da mümkündür.

Model bileşenleri birbirleri ile eşzamanlı ve paralel olarak evrilmektedir. Bu paralel evrilme için Julia programlama dilinin görev(task) birimleri kullanılmıştır. Görevler benzetim sırasında bileşenlerin evrilme aşamaları arasında geçiş yapabilmeyi sağlamaktadır. Julia programlama dilinin dağıtılmış programlama kabiliyeti kullanılarak bu görevlerin birden fazla işlemci üzerine dağıtılması mümkündür.

Binlerce düğümden oluşan çok büyük sistem ağları düşünüldüğünde, sistem modellerinin kolay ve hızlı bir şekilde oluşturulabilmesi önemli bir avantajdır. JuSDL, kolay bir söz dizimi ile Julia'nın dağıtılmış programlama araçlarını kullanarak birden çok mikroişlemci çekirdeğindeki büyük modelleri simüle edebilir.

5 Conclusions

In this paper, we introduced JuSDL. In JuSDL, the simulation is performed by evolving the components according to their mathematical equations between the sampling intervals in parallel and independently. The components can be static or dynamical or represented by mathematical equations having continuous or discrete-time variables. The simulation of the models consisting of components defined by ordinary, random ordinary, stochastic, delay differential, differential-algebraic, or difference equations is possible. It is not an obligation to describe all the model components with the same type of mathematical equation.

The data flowing through the connections can be directly recorded or visualized during the simulation. In addition to offline analyses, it is also possible to carry out online data analysis with user-defined plugins.

The model components evolve in parallel and simultaneously. The tasks of the Julia programming language is used for this parallel evolution. The tasks allow switching between the evolution of the components during the simulation. Using the Julia programming language's distributed computing tools, it is possible to distribute computation workload on multiple processors.

When considering extensive system networks consisting of thousands of nodes, it is an essential advantage that the system models can be created easily and quickly. The JuSDL can simulate large models in multiple microprocessor cores using Julia's distributed programming tools with an easy syntax.

6 Yazar katkı beyanı

Gerçekleştirilen çalışmada, Zekeriya SARI fikrin geliştirilmesi, literatür taraması, kodların oluşturulması, uygulama sonuçlarının elde edilmesi ve değerlendirilmesi başlıklarında; Serkan GÜNEL fikrin oluşturulması, sonuçların değerlendirilmesi, kullanılan bilgi işlem alt yapısının oluşturulması, sonuçların incelenmesi, yazım denetimi ve içerik açısından makalenin kontrol edilmesi, başlıklarında katkı sunmuşlardır.

7 Etik kurul onayı ve çıkar çatışması beyanı

Hazırlanan makalede etik kurul izni alınmasına gerek yoktur.

Hazırlanan makalede herhangi bir kişi/kurum ile çıkar çatışması bulunmamaktadır.

8 Kaynaklar

[1] Elmqvist H. A Structured Model Language for Large Continuous Systems. PhD Thesis, Lund Institute of Technology, Lund, Sweden, 1978.

[2] Nytsch-Geusen C, Ernst T, Nordwig A., Schwarz P, Schneider P, Vetter M, Wittwer C, Holm A, Nouidui T, Leopold J, Schmidt G, Mattes. A. "Advanced Modeling and Simulation Techniques in MOSILAB: A System Development Case Study". *Proceedings of the 5th International Modelica Conference*, Vienna, Austria, 4-5 September 2006.

[3] Zimmer D. "Introducing sol: a general methodology for equation-based modeling of variable-structure systems". *Proceedings of the 6th International Modelica Conference*, Bielefeld, Germany, 3-4 May 2008.

[4] Mosterman PJ. "HYBRSIM-A modeling and simulation environment for hybrid bond graphs". *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* 216(1), 35-46, 2002.

[5] Barton PI. The Modelling and Simulation of Combined Discrete/Continuous Processes. PhD. Thesis, Imperial College of Science, Technology and Medicine, London, United Kingdom, 1992.

[6] Van Beek DA. "Variables and Equations in Hybrid Systems with Structural Changes". *Proc. 13th European Simulation Symposium*, Marseille, France, 18-20 October 2001.

[7] Giorgidze G, Nilsson H. "Higher-Order Non-Causal modelling and simulation of structurally dynamic systems". *Proceedings of the 7th International Modelica Conference*, Como, Italy, 20-22 September 2009.

[8] Pfeiffer, A, Hellerer, M, Hartweg, S, Otter, M, Reiner, M. "PySimulator-A simulation and analysis environment in Python with plugin infrastructure". *9th International Modelica Conference*, Munich, Germany, 3-5 September 2012.
[9] Mathworks. "Simulink-Simulation and Model-Based Design". https://www.mathworks.com/products/simulink.html (18.03.2020).
[10] Coetzee, E. "Dynamical Systems Toolbox". https://www.mathworks.com/matlabcentral/fileexchange/32210-dynamical-systems-toolbox (18.03.2020).
[11] Stewart, H, Breakspear, Michael. *Handbook for the Brain Dynamics Toolbox.* Brisbane, QLD: QIMR Berghofer Medical Research Institute, 2017.
[12] Neirynck N. Advances in Numerical Bifurcation Software: MatCont. PhD Thesis, Ghent University, Ghent, Belgium, 2019.
[13] Datseris G. "DynamicalSystems. jl: A julia software library for chaos and nonlinear dynamics.". *Journal of Open Source Software*, 3(23), 598-602 2018.
[14] Rackauckas C, Nie Q. "Differential equations.jl-a performant and feature-rich ecosystem for solving differential equations in julia". *Journal of Open Research Software*, 2017. https://doi.org/10.5334/jors.15.
[15] Dokuz Eylül Üniversitesi. "Julia Tabanlı Sistem Tanımlama Dili ve Benzetim Ortamı: JuSDL". https://imel.eee.deu.edu.tr/git/JuSDL.jl.git (18.03.2020).
[16] Julia Computing. "The Julia Programming Language". https://julialang.org (18.03.2020).
[17] Bezanson J, Edelman A, Karpinski S, Shah VB. "Julia: a fresh approach to numerical computing". *SIAM Review*, 59, 65-98, 2017.
[18] Bezanson J, Karpinski S, Shah VB, Edelman A. "Julia: A Fast Dynamic Language for Technical Computing". https://arxiv.org/abs/1209.5145 (18.03.2020)
[19] Matei I, Bock C. "Modeling Methodologies and Simulation for Dynamical Systems". National Institute of Standards and Technology, US Department of Commerce, 7875, 2012.
[20] Lamego MM. "Adaptive structures with algebraic loops". *IEEE Transactions on Neural Networks*, 12, 33-42, 2001.
[21] Van Der Schaft AJ, Schumacher JM. *An Introduction to Hybrid Dynamical Systems*. London, United Kingdom, Springer, 2000.
[22] Johnson JB. "Thermal agitation of electricity in conductors". *Physical Review*, 1928. https://doi.org/10.1103/PhysRev.32.97.
[23] Matsumoto T, Chua L, Komuro M. "The Double Scroll". *IEEE Transactions on Circuits and Systems*, 32, 797-818, 1985.
[24] Kloeden PE, Platen E. *Numerical Solution of Stochastic Differential Equations.* Berlin, Germany, Springer, 2013.
[25] Rosenstein MT, Collins JJ, De Luca CJ. "A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets". *Physica D: Nonlinear Phenomena*, 65, 117-134, 1993.
[26] Erdös P, Renyi A. "On random graphs I". *Publicationes Mathematicae*, 6, 290-297, 1959.
[27] Mathworks. "MATLAB". https://www.mathworks.com/products/matlab.html (18.03.202)
[28] Open Source Modelica Consortium. "OpenModelica". https://www.openmodelica.org/ (18.03.2020).