PARALEL ÇÖZÜMLEME DESTEKLİ YÜRÜYÜŞ SİMÜLATÖRÜ

Şahin YILDIRIM, Erdem ARSLAN*

Erciyes Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği Bölümü 38039,KAYSERİ

sahiny@erciyes.edu.tr, erdemarslan@erciyes.edu.tr

Yürüyen robotların hareketi esnasında karşılaşabilecekleri dinamik etkileri göz önünde bulunduran kontrolörlerin tasarımı bacaklı mobil robotlar alanında oldukça önemli bir ihtiyaçtır. Son yıllarda yürüyen mobil robotlar gibi karmaşık dinamik sistemler için analitik olarak modelleme yapılıp ideal kontrolörler tasarlamak yerine, içerisinde sayısal simülasyon algoritmalarının kullanıldığı model esaslı kontrol yapıları tercih edilmeye başlanmıştır. Open Dynamics Engine(ODE) rijit cisimlerin simülasyonu ve kontrolü için kullanılan bir fizik simülatörüdür. Daha önce altı bacaklı bir mobil robot için geliştirdiğimiz ODE tabanlı yürüyüş ve dengeleme algoritması sayesinde kontrol esnasında oluşan dinamik etkilerin göz önünde bulundurulduğu bir kontrolör yapısı elde edilmişti. Bu çalışmada ODE tabanlı yürüyüş ve dengeleme algoritmasına ait on iki adet parametrenin ayarlanması işleminin daha efektif bir şekilde gerçekleştirilebilmesi için paralel çözümleme desteğine sahip bir yürüyüş simülatörü geliştirilmiştir. Geliştirilen bu simülatör sayesinde, normalde test işleminde kullanılan 30 saniyelik simülasyon 480 kat hızlandırılarak 0.0625 saniyede çözülebilir bir hale getirilmiştir. Bu simülatör kullanılarak yapılan parametre ayarlama işlemleri sonucunda daha önce elde edilen %65.54 alan hatası iyileşmesi, %75.85 değerine çıkarılmıştır.

Anahtar Kelimeler: Bacaklı mobil robot, paralel çözümleme, yürüyüş simülatörü, parametre ayarlama

PARALLEL ANALYSIS SUPPORTED WALKING SIMULATOR

ABSTRACT

The design of controllers which considering the dynamic effects that walking robots may encounter during their movement, is an important requirement in the field of legged mobile robots. In recent years, in the control of complex dynamical systems, model based control structures which uses numerical simulation algorithms, have begun to be preferred. Open Dynamics Engine (ODE) is a physics simulator for modelling and controlling of rigid bodies. In the ODE-based stabilization and walking algorithm that earlier developed for a six-legged mobile robot, a model based controller structure was obtained that takes into account the disturbance effects that occur during control. In this study, a gait simulator with parallel analysis support has been developed in order to perform the adjustment of twelve parameters of the ODE-based walking and stability algorithm more effectively. With this developed simulator, the 30-second simulation that is normally used in the test process has been accelerated 480 times, making it solvable in 0.0625 seconds. As a result of the tuning processes performed using this simulator, the previously obtained 65.54% area error improvement was increased to 75.85%.

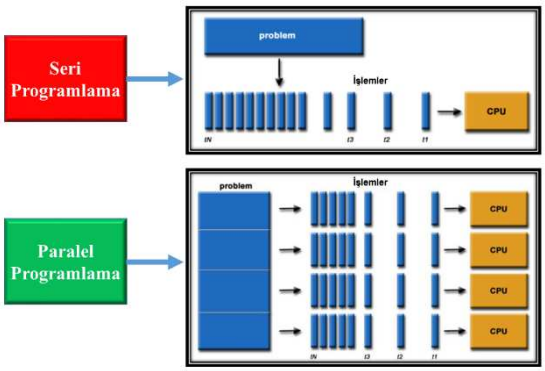
Keywords: Legged mobile robot, parallel computing, walking simulator, parameter tuning

1. GİRİŞ

Paralel çözümleme, aynı anda birçok işlemin eşzamanlı olarak gerçekleştirilebildiği özel bir hesaplama türüdür [1]. Genelde bu hesaplama tekniğinde çözümü oldukça uzun süren problemler eşzamanlı olarak çözülmesi amacı ile küçük parçalara bölünürler. Bu bölünme işlemi sayesinde sistemde kullanılan işlemci sayısına bağlı olmakla birlikte problemin çözüm süresi ciddi manada düşürülür. Geleneksel programlama mimarisine göre kodlanan algoritmalar, altyapısında özel bir mimari kullanılmadığı takdirde seri olarak çalışmak üzere geliştirilmişlerdir [2].

Günümüzde paralel hesaplama teknikleri karmaşık analizlerin kısa bir sürede yapılması adına oldukça büyük bir önem arz etmektedir. Paralel programlama sayesinde bilgisayar üzerinde yapılan hesaplamalar daha hızlı ve verimli bir şekilde gerçekleştirilebilmektedir. En basit tanımı ile paralel programlama belirli bir işlemi gerçekleştirmek için eşzamanlı olarak birden fazla kaynak kullanarak hesaplama süresinin kısaltılması işlemidir [3]. Burada kaynak ifadesi bilgisayar üzerinde hesaplama için kullanılacak işlemci sayısını ve RAM (Random Access Memory) miktarını temsil etmek için kullanılır.

Standart programlama mimarilerine göre kodlanan algoritmalar genelde seri olarak çalışacak şekilde düzenlenmektedir [4]. Bu mimarilerde çözülmesi gereken problemler farklı zamanlarda aynı işlemciye hesaplatılması amacı ile farklı küçük parçalara bölünürler. Standart programlama mimarilerinde CPU üzerinde aynı anda sadece bir işlem yapılabildiğinden, küçük parçalara bölünmüş bu işlemler birbirlerinin ardı sıra çalıştırılmak üzere bekletilirler. Şekil 1'de tek bir CPU üzerinde çalışan seri programlama ve 4 çekirdekli bir CPU üzerindeki çalışan paralel programlama mimarileri gösterilmiştir.



Şekil 1. CPU üzerinde çalışan seri ve paralel programlama mimarileri.

2. PARALEL PROGRAMLAMA

Paralel programlama mimarisinde parçalara bölünmüş işlemler mimari üzerinde bulunan birden çok CPU'ya eşzamanlı olarak gönderilebilmektedir. Bu parçalara bölünmüş işlemlerin eş zamanlı olarak çözülebilmesi, hesaplama için harcanan sürenin kısalacağı anlamına gelmektedir. Her ne kadar paralel programlama mimarisinde çözüm süresi kısaltılıyor olsa da, mimarideki CPU'ların birbirlerinden farklı hızlarda çalışıyor olmalarından dolayı senkronizasyon işleminini doğru yapılmadığı durumlarda büyük bir performans artışı sağlanmayabilir [5]. Bu sebepten dolayı paralel programlama mimarilerinde çalıştırılacak işlemlerin birbirleri ile doğru bir şekilde senkronize edilmesi önemlidir.

Paralel programlama mimarilerinde ilk olarak çözülmesi gereken problemin paralelleştirilmesi işlemi gerçekleştirilir. Problemin paralelleştirilmesinde; paylaşımlı hafıza, paylaşımlı işlemler, dağıtımlı hafıza, veri paralelliği, hibrit, tek program çoklu veri ve çoklu program çoklu veri gibi birçok model kullanılabilmektedir [6]. Problemin durumuna ve çözüm için kullanılacak bilgisayar donanımına göre seçilmesi gereken bu paralelleştirme modelinde çözüm uzayının hangi bölümünün hangi işlemci tarafından ne zaman çözülmesi gerektiği belirlenir. Bu çalışma kapsamında, daha önce geliştirilen ODE tabanlı denge ve yürüyüş kontrolörüne ait

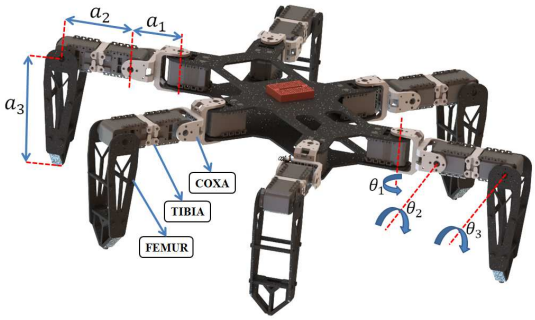
parametrelerin ayarlanması için paralel çözümleme destekli bir yürüyüş simülatörünü geliştirilmiştir. Bu simülatör modeli ve ayarlanması gereken kontrolör parametreleri üçüncü bölümde ayrıntılı olarak açıklanacaktır.

Paylaşımlı işlemler modelinde tek bir işlem farklı CPU'lar üzerinde farklı girdileri işlemek üzere kurgulanabilmektedir. Bu sebepten dolayı, yürüyüş simülatörü paralelleştirilirken paylaşımlı işlemler modeli tercih edilmiştir.

Paylaşımlı işlemler paralelleştirme modelinde temel olarak Posix threadleri ve OpenMP olmak üzere iki farklı yöntem kullanılabilir [7]. Posix threadleri metodu kütüphane tabanlı bir paralelleştirme modeli olduğu için performans konusunda ciddi bir katkı sağlamaz. OpenMP, içerisinde birden çok işlemciye sahip bir PC'nin işlemcilerini paralel olarak programlamak amacı ile geliştirilmiş bir uygulama arayüzüdür. INTEL, AMD ve NVIDIA gibi birçok büyük işlemci üreticisi bu arayüzü CPU'larına donanımsal olarak çalışacak şekilde eklemişlerdir [8]. OpenMP donanımsal olarak işlemciler içerisinde bulunduğu için diğer paralelleştirme arayüzlerine göre daha iyi performans sergilemektedirler [9]. Bu sebepten dolayı bu çalışmada geliştirilen yürüyüş simülatörünün paralelleştirilmesinde OPENMP modeli kullanılmıştır.

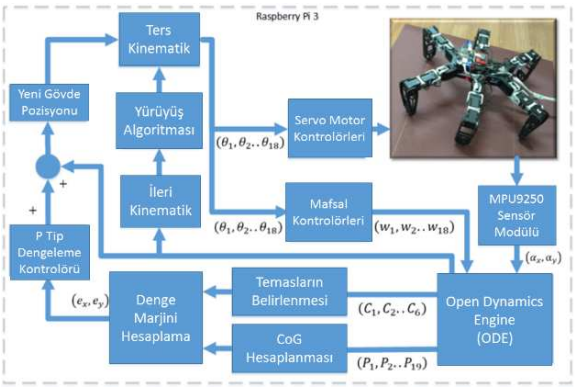
3.ODE TABANLI YÜRÜYÜŞ KONTROL ALGORİTMASI

Bu uygulamalı araştırma çalışmasında yapılan deneylerde altı bacaklı bir mobil robot kullanılmıştır. Deneysel olarak kullanılan robot, her bir bacakta üç serbestlik derecesi olmak üzere toplamda on sekiz adet serbestlik derecesine sahiptir. Servo motorlar, batarya ve elektronik donanım aksamı dâhil robotun toplam ağırlığı 2,4 kg'dır. Deneylerde kullanılan robot Şekil 2'de verilmiştir.



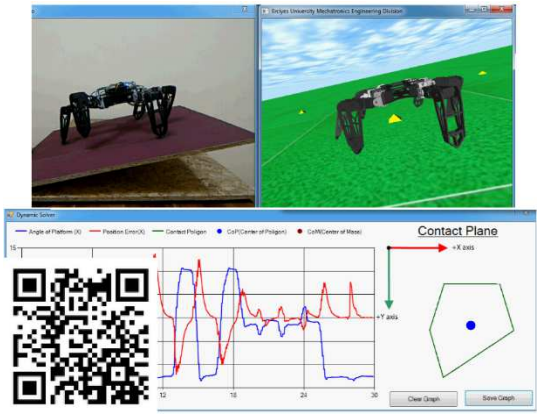
Şekil 2. Deneylerde kullanılan robot.

Yürüyüş kontrolörü içerisinde ODE fizik simülatörünün kullanım amacı hareket esnasında ivmelenmelerden kaynaklı oluşacak atalet kuvvetlerinin göz önünde bulundurulması ihtiyacıdır. Daha önce geliştirilen mafsal ve denge kontrolörleri sayesinde hem bu atalet kuvvetlerinden gelen etkilere hem de robotun dışarıdan gelen bozucu etkilere göre kendisini otomatik olarak dengeye getirebildiği bir kontrol yapısı geliştirilmiştir. Kontrol işlemini açıklar blok diyagram Şekil 3'te verilmiştir.



Şekil 3. Geliştirilmiş ODE tabanlı yürüyüş kontrol algoritmasının akışını gösteren blok diyagramı.

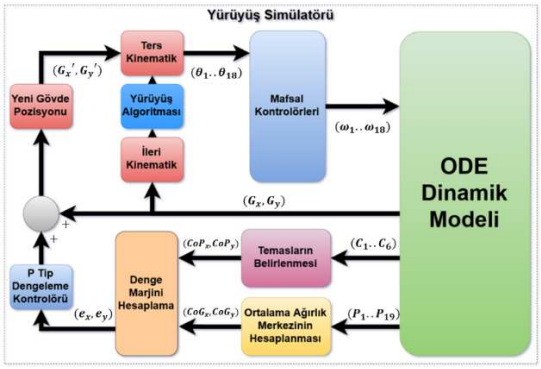
Bu kontrol sisteminin temel amacı robotun ortalama ağırlık merkezini sürekli olarak, robotun yere temas eden ayaklarının oluşturduğu kontak poligonunun geometrik merkezinde tutmaktır. Şekil 4'te Sistemin çalışmasını gösterir arayüz verilmiştir. Sistemin gerçek zamanlı çalışması görseldeki karekod okutularak izlenebilir.



Şekil 4. Sistemin gerçek zamanlı çalışması

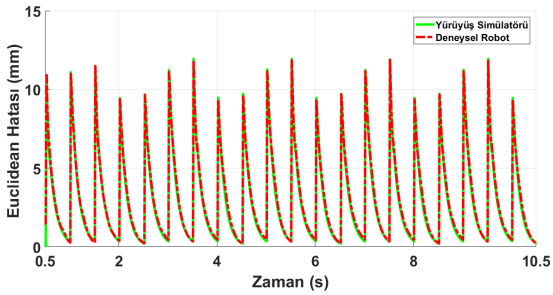
4. PARALEL ÇÖZÜMLEME DESTEKLİ YÜRÜYÜŞ SİMÜLATÖRÜ

Paralel çözümleme destekli yürüyüş simülatörü geliştirilirken, ODE tabanlı denge yürüyüş kontrolörü esas alınmıştır. Simülatör elde edilirken ilk olarak yürüyüş kontrolöründeki deneysel robotun gerçek zamanlı eğim değerini ODE'ye aktarmak için kullanılan MPU-9250 sensör modülü iptal edilmiştir. Simülatörde herhangi bir fiziksel etkileşim incelenmeyeceği için aynı kontrolör üzerindeki servo motor kontrolör blokları da iptal edilmiştir. Daha sonra simülasyon işlemini hızlandırmak adına, ODE platformundaki görselleştirme için kullanılan OpenGL bloğu ve gerçek sistemle eşzamanlı çalışmayı sağlayan gecikme işlemleri, kontrol algoritmasından kaldırılmıştır. Bu blokların çıkarılması sonrası geriye kalan kontrolör yapısı yürüyüş simülatörü olarak adlandırılabilir. Şekil 5'te yürüyüş simülatörüne ait blok diyagram verilmiştir.



Şekil 5. Yürüyüş Simülatörüne ait Blok diyagramı

Yukarıda blok diyagramı verilen yürüyüş simülatörü daha önce Bölüm 3'te geliştirilen yürüyüş kontrolünden elde edilmiştir. Blok diyagramların çıkarılması ile elde edilen yürüyüş simülatörünün doğru çalışıp çalışmadığı valide edilmelidir. Bu amaçla simülatöre ve deneysel robota aynı kontrol parametrelerinin girilmesi durumunda ODE tabanlı tripod yürümenin gerçekleştirildiği 5 adımlık bir senaryo için yürüyüş simülatörü ve deneysel robot üzerinden alınan verilerin karşılaştırması yapılmıştır. Şekil 6'da yürüyüş simülatörü ve deneysel robot çıkışında görülen alan hatalarının karşılaştırması verilmiştir.



Şekil 6. Yürüyüş Simülatörü ve Deneysel Robot Çıkışında Görülen Alan Hatalarının Karşılaştırması.

Yukarıda verilen grafikten de görüleceği üzere robota ve yürüyüş simülatörüne aynı kontrol parametreleri girilmesi durumunda elde edilen alan hatalarının zamana bağlı değişimleri birbirlerine ciddi manada benzemektedir. Tablo 1'de simülatöre ve robota aynı kontrol parametreleri girilmesi durumunda elde edilecek alan ve sapma hataları verilmiştir.

Tablo 1. . Yürüyüş Simülatörü ve Deneysel Robot Sonuçlarının Karşılaştırması.

	Alan Hatası (mm.s)	Sapma Hatası (birimsiz)
Deneysel Robot	30,256	17,903
Yürüyüş Simülatörü	28,957	15,850

Tabloda verilen değerlerin birbirlerine oldukça yakın olması geliştirilen simülatörün doğruluğunu göstermektedir. Sonuçlar arasındaki farkın sebebi deneysel robottaki kontrol yapısında simülatöre göre fazladan bulunan MPU-9250 IMU modülüdür. IMU modülü açısal yerdeğiştirmeyi hesaplarken altyapısında bulunan ivmelenme sensörünü kullanmaktadır. Dolayısı ile robotun hareketi

esnasında gövde parçası üzerinde oluşacak atalet kuvvetleri bu ölçümü doğrudan etkilemektedir. Robot üzerinde bu etki çok hızlı bir döngü içerisinde ivedilikle telafi edildiği için denge ve yürüyüş kontrolü hususunda herhangi bir problem oluşturmamaktadır.
Simülatör üzerinde böyle bir etkinin modellenememesinden dolayı sonuçlar arasında ufak bir farklılık görülmektedir. Yine de sonuçların grafiksel olarak ciddi manada birbirine benziyor olması yürüyüş simülatörünün parametre ayarlama işleminde deneysel robotun yerine kullanılabilmesi için yeterli bir nedendir.

Geliştirilen bu yürüyüş simülatörünün deneysel doğrulama işleminden sonra daha hızlı çözüm yapabilmesi adına paralel olarak çalışabilecek bir şekilde yeniden düzenlenmesi gerekmektedir. Yürüyüş simülatöründe temel olarak yapılan işlem, analiz dosyasından okunan 12 parametrenin ODE tabanlı denge ve yürüyüş kontrolörüne uygulanması ve bu kontrol parametrelerinin kullanıldığı bir yürüyüş senaryosu sonucunda elde edilecek sapma ve alan hatalarının çıktı dosyasına yazılması işlemidir. Normalde bu işlemin tek bir CPU üzerinde yapılması durumunda aynı anda sadece bir simülasyon işlemi gerçekleştirilebilmektedir. Bu işlemin paylaşımlı işlemler modeli kullanarak paralelleştirilmesi durumunda ise aynı anda sistemde bulunan işlemci sayısı kadar simülasyon gerçekleştirebilmektedir. Paralelleştirme işleminde paylaşımlı işlemler modelinin seçilmesinin en önemli sebebi farklı cpular üzerinde eş zamanlı olarak aynı işlemin çalıştırılabiliyor olmasıdır. Bu paralelleştirme modelinde ortak bir veri havuzundan alınan veriler farklı CPU'lar üzerinde çalışan aynı program tarafından işlenmekte ve sonuç yine başka bir çıktı havuzuna yazılmaktadır.

Çok çekirdekli CPU'lar üzerinde paralel olarak çalışabilecek programların geliştirilebilmesi için yazılımsal bir arayüze ihtiyaç duyulmaktadır. ODE yürüyüş simülatörü C++ programlama dili kullanılarak kodlanmıştır. C++ programlama dilinde bir programın farklı CPU'lar üzerinde paralel bir şekilde çalıştırılabilmesi için iki farklı arayüz kullanılabilmektedir. [7]. . Bu arayüzler Posix threadleri ve OpenMP'dir.

C++ derleyicisinde Posix threadleri arayüzü kullanıldığında işlemler, aynı anda farklı CPU'lara doğrudan uygulanmak yerine, yazılımsal bir kütüphaneye gönderilmektedir. Bu yazılımsal kütüphanenin görevi, kendisine gelen işlemleri farklı zamanlarda çalıştırılması için işletim sistemine iletmektir. İşletim sistemlerinde temel olarak yapılan şey kendisine gelen binlerce farklı işlemin CPU'lara dengeli bir şekilde dağıtılmasıdır. İşletim sisteminde Posix threadlerinin yanında aynı anda binlerce işlem de dağıtıldığı için yazılımsal bir kütüphane üzerinden paralelleştirme yapılması durumunda paralelleştirmenin sağlayacağı performans artışı işletim sistemine bağlı olacaktır.

OpenMP, çok çekirdekli işlemcilerin C++ dili kullanılarak programlanabilmesi amacı ile geliştirilmiş yazılımsal bir arayüzdür. Posix threadlerinin aksine OpenMP arayüzünde donanımsal seviyede CPU'lara erişilmesi mümkündür. Yani diğer bir ifade ile OpenMP kullanılması durumunda işletim sistemi bypass edilebilmektedir. Geliştirilen paralel programlarda işletim sisteminin bypass edilmesi, paralel programların performanslarının işletim sisteminde çalışan diğer işlemlerden etkilenmemesi anlamına gelmektedir. Bu sebepten dolayı yürüyüş simülatörünün paralelleştirilmesinde OpenMP yazılımsal arayüzü kullanılmıştır. Paralel çözümleme destekli yürüyüş simülatöründe sırası ile aşağıdaki işlemler gerçekleştirilmektedir:

1.Sistemdeki fiziksel olarak bulunan işlemci sayısı belirlenir (n).

2.Analiz dosyasından denenecek toplam parametre kümesi sayısı okunur.

3.Analiz dosyasından n sayısı kadar parametre kümesi alınır.

4.İstenilen sayıda yürüyüş simülatörü oluşturulur.

5.Okunan bu n sayıdaki parametre kümesi, oluşturulan bu n sayıdaki yürüyüş simülatörüne uygulanır.

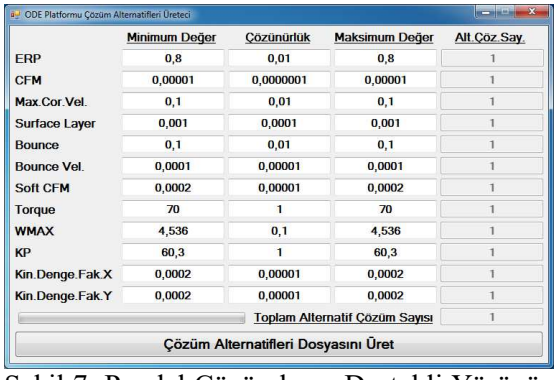
6.n adet yürüyüş simülatörü sistemde bulunan n farklı CPU üzerinde eşzamanlı olarak başlatılır.

7.Paralel olarak yapılan analizler sonucunda elde edilen alan ve sapma hataları, sonuç dosyasına eklenir.

8.Analiz dosyasındaki tüm parametre kümeleri test edildi ise işlem durdurulur. Yoksa 3. nolu adıma gidilir.

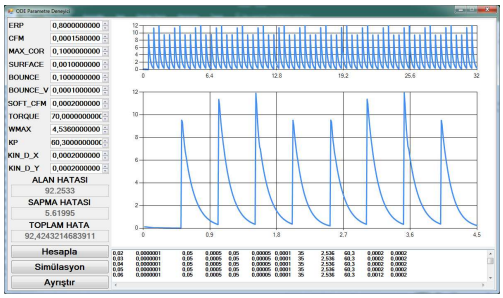
Yukarıda verilen işlemlerin sıralı bir şekilde uygulanması ile analiz dosyası içerisinde bulunan tüm parametre kümeleri sistemde bulunan tüm işlemciler tarafından çözümlenmiş olur. Paralel çözümleme desteği sayesinde sistemde bulunan fiziksel işlemci sayısına bağlı olmak kaydıyla analiz süresi ciddi manada kısaltılmış olur.

Analiz dosyası oluşturulurken, etkisi incelenmek istenen her bir kontrol parametresi için belirli bir aralık ve çözünürlük değeri tanımlanmalıdır. Bu ihtiyacı karşılamak için, çözüm alternatifleri üreteci platformu geliştirilmiştir. Bu programda ilk olarak denenmesi istenen her bir parametre için aralıkların ve çözünürlük değerlerinin tanımlanması gerekmektedir. Daha sonra programda bulunan çözüm alternatifleri dosyasını üret butonuna basılması ile analiz dosyası elde edilmiş olur. Şekil 7'de paralel çözümleme destekli yürüyüş simülatörü için analiz dosyasını üretildiği arayüz verilmiştir.



Şekil 7. Paralel Çözümleme Destekli Yürüyüş Simülatörü İçin Analiz Dosyasını Üretildiği Arayüz.

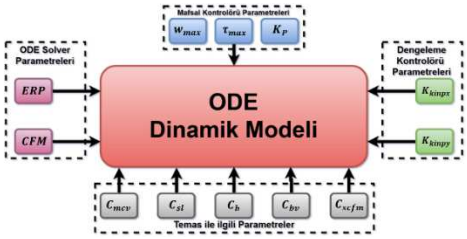
Paralel çözümleme destekli yürüyüş simülatörü sonuçlarının kolay bir şekilde görüntülenebilmesi için ODE parametre deneyici platformu geliştirilmiştir. Bu programda ekran üzerinden girilen kontrol parametreleri hızlı bir şekilde simüle edilmekte ve sonuçlar anlık olarak grafiksel ekran üzerinde görüntülenebilmektedir. Şekil 8'de paralel çözümleme destekli yürüyüş simülatörünün sonuçlarını görüntülemek için kullanılan arayüz verilmiştir.



Şekil 8. Paralel Çözümleme Destekli Yürüyüş Simülatörünün Sonuçlarını Görüntülemek için Kullanılan Arayüz.

5. TUNİNG İŞLEMLERİ

ODE tabanlı denge ve yürüyüş kontrolöründe yürüyüş performansını arttırmak için ayarlanabilecek toplam 12 parametre bulunmaktadır. Bu parametreler sistemde bulunan dört ana işleme etki etmektedir. Bu işlemler sırasıyla; ODE solver, mafsal kontrolörleri, Dengeleme kontrolörleri ve temas algılama işlemleridir. Bu parametreler Şekil 9'da verilen blok diyagram üzerinde gösterilmektedir.



Şekil 9. ODE dinamik modeli üzerinde ayarlanması gereken parametreler.

ODE solver'a ait değiştirilmesi mümkün iki parametre mevcuttur. Bunlar ERP ve CFM parametreleridir. Bu parametrelerin ODE solver üzerinde nasıl bir etki oluşturduğu [10-12] referanslı çalışmalarda detaylı bir şekilde verilmiştir. Ayrıca bu parametrelere ek olarak ODE dinamik modelinde, altı bacaklı mobil robotun ayakları ile yer zemini arasında oluşacak teması etkileyecek beş adet daha parametre bulunmaktadır. Bu parametreler sırasıyla; maksimum temas düzeltme hızı, yüzey katmanı büyüklüğü, temas zıplama faktörü, temas zıplama hızı ve temasın yumuşaklık faktörü parametreleridir. Bu parametrelerin temas oluşması durumunda ne

gibi bir etki oluşturduğunun anlaşılması için ODE'nin kullanıcı manueli incelenebilir [13].
ODE dinamik modelinde mafsal kontrolörlerinin çalışmalarını etkileyecek üç parametre bulunmaktadır. Bu parametreler sırasıyla; maksimum mafsal hızı, maksimum mafsal torku ve mafsal kontrol katsayısı parametreleridir.
ODE tabanlı dengeleme ve yürüyüş kontrolöründe sistemin çalışmasını etkileyecek iki parametre bulunmaktadır. Bu parametreler X ve Y yönündeki dengeleme işlemine doğrudan etki edecek kinematik dengeleme kontrolör katsayılarıdır. Gövde dengelemesinde aynı anda iki yönde de dengeleme gerçekleştirlidiği için iki farklı kontrol katsayısı tanımlanabilmektedir. Fakat robotun başlangıç duruşunun simetrik seçilmesinden dolayı ayarlama işlemi sırasında bu iki parametre için tek bir değer seçilecektir.

Yürüyüş simülatörü üzerinde yapılacak analizlerde, deneysel sistemde doğru bir iyileştirme yapabilmek adına [12] nolu çalışmada uygulanan yürüyüş şartları kullanılmıştır. Bu yürüyüşlerde robotun her bir adımında sadece dx yönünde 3 cm'lik bir ilerleme yapılmış ve bacakların geçici olarak havada tutulduğu yerden yükseklik mesafesi dz 2 cm olarak seçilmiştir. Ayrıca simülatör üzerinde yapılan yürüyüşlerde ayak adımları arası gecikme süresi 500 milisaniye olarak seçilmiştir. Gerçekleştirilen deneylerde, yürüyüşlerin çok zaman alması sebebi ile deneyler robot beş adım attıktan sonra bitirilmiştir. Yürüyüş simülatöründe yapılacak analizler deneylerden farklı olarak, onbeş adım sonunda bitirilmiştir. Yürüyüş simülatöründe yapılacak analizlerde robotun on beş adım yürütülmesinin sebebi, parametrelerin etkilerinin daha uzun bir süre ile incelenmek istenmesidir.

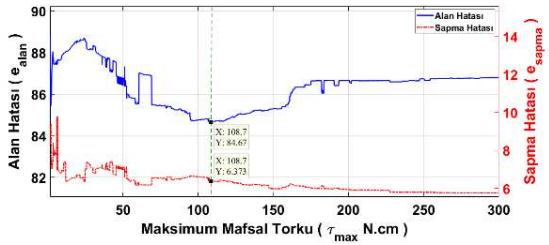
ODE tabanlı dengeleme ve yürüyüş kontrolörüne ait 12 farklı parametrenin seçim işlemlerinde ilk olarak, her bir parametrenin etkisini ayrı olarak incelemek amacı ile 12 farklı analiz dosyası oluşturulmuştur. Bu analiz dosyalarını oluşturmak için şekil 7'de görseli verilen ODE platformu çözüm alternatifleri üreteci programı kullanılmıştır. Oluşturulan her bir analiz dosyasında optimum parametrenin aranacağı uzay için minimum ve maksimum noktalar tanımlanmalıdır. Ayrıca bu programda arama uzayındaki analiz sayısını belirlemek için uygun sayıda bir çözünürlük değerinin girilmesine de ihtiyaç vardır. Daha sonra on iki farklı parametre için oluşturulan bu analiz dosyaları sırasıyla paralel çözümleme destekli yürüyüş simülatörüne uygulanmıştır.

Parametre ayarlama işleminde ODE tabanlı denge ve yürüyüş kontrolörü çıkışında en az alan hatası verecek on iki farklı parametrenin belirlenmesi işlemi gerçekleştirilmektedir. Bu işlem için analizler sonucunda elde edilen içerisinde belirli bir parametre değişimine göre alan ve sapma hatalarının değişimini bulunduran sonuç dosyalarından faydalanılmıştır. Alan ve sapma hataları hakkındaki bilgi için [11] nolu çalışma incelenebilir.
Sonuç dosyaları içerisinde her bir parametre için belirlenen bir çözünürlük ve aralık değerine göre simüle edilmiş onbinlerce yürüyüşe ait alan ve sapma hatası değerleri bulunmaktadır. Bu yürüyüşlerden birçoğunun teknik olarak robota uygulanması imkânsızdır. Sonuç dosyaları içerisinde bulunan binlerce yürüyüşün görsel olarak incelenmesi mümkün olmadığından, uygun olmayan yürüyüşlerin sonuç dosyasından çıkarılmasını sağlayacak bir performans kriterine ihtiyaç duyulmuştur.

Bu amaçla yürüyüş esnasındaki adımların birbirlerine olan benzerliğini temsil etmek üzere geliştirilen sapma hatası değerinden faydalanılmıştır. Bu performans kriterinde sapma hatası için belirli bir eşik değeri tanımlanmakta ve sonuç dosyası içerisindeki teknik olarak uygulanması mümkün olmayan parametre kümeleri bu eşik değerine göre sonuç dosyasından temizlenmektedir. Dolayısı ile 12 farklı parametre için yapılan analizlere ait tüm sonuç dosyaları içerisinden, sapma hatasının eşik değerinden büyük olduğu parametre kümeleri silinmiştir. Bu çalışmadaki parametre belirleme işleminde sapma hataları için belirlenen eşik değeri 10'dur.

Parametre belirleme işleminde ilk olarak mafsal kontrolörüne ait τ_{max} parametresi ile başlanmıştır. Bu parametre için 0.01-300 N.cm aralığında 0.1 çözünürlükte 3000 farklı değer analiz edilmiştir. Elde edilen sonuç Şekil 10'da grafiksel olarak gösterilmektedir.

Bu aralıkta τ_{max} parametresi için alan hatasına göre en uygun değeri 108,7'dir.



Şekil 10. Maksimum mafsal torku parametresine göre alan hatası ve sapma hatasının değişimi.

Bu çalışmada sırası ile 12 parametre denenmiş ve bu deneme işleminin her bir adımında elde edilen parametre değeri seçilen değer olarak kabul edilmiştir. Bildirinin çok uzun olacağı sebebi ile tüm grafikler burada verilememiştir. Fakat sıralı seçim işlemi tarif edildiği şekli ile yürütülmüştür.

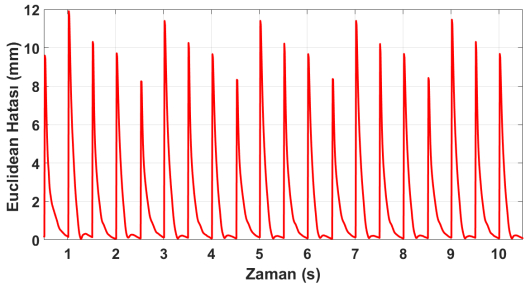
6. SONUÇLAR

Belirlenen bu on iki adet kontrol parametresi daha sonra deneysel robot üzerine uygulanmıştır. Bu parametre ayarlama işlemi sonucunda elde edilen iyileşmenin ortaya koyulması adına bir takım deneyler gerçekleştirilmiştir. Deneylerde robot düz bir zemin üzerinde yürütülmüştür. Bu düz zemin üzerinde sırasıyla ODE tabanlı tripod, ODE tabanlı ripple ve ODE tabanlı wave yürüyüş düzenleri test edilmiştir[11].

Yürüyüş düzenleri için yapılan deneylerde robot toplam beş adım yürütülmüştür. Yürüyüşlerin hızını belirlenemek için kullanılan ayakadımları arası gecikme süresi, bu bölümde elde edilen sonuçların daha önce elde edilen sonuçlar ile karşılaştırılabilmesi için 500 milisaniye olarak alınmıştır.

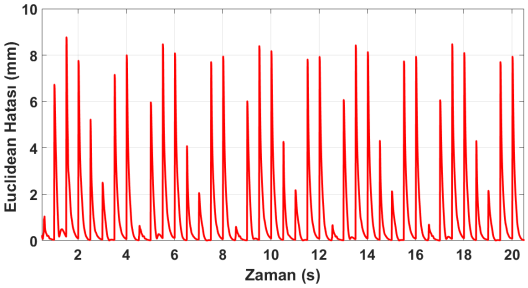
Robotun atacağı her bir yeni adımda robot sadece X ekseni yönünde 3 cm'lik bir ilerlemede bulunacaktır. Yürüyüş düzenlerinde bacakların geçici olarak havada tutulması için gerekli dz mesafesi 2 cm olarak alınmıştır.

Şekil 11'de ODE tabanlı tripod yürüyüş düzenine paralelleştirme sonucunda elde edilen yeni kontrol parametrelerinin uygulanması durumunda elde edilen euclidean hatasının zamana bağlı değişim grafiği verilmektedir.



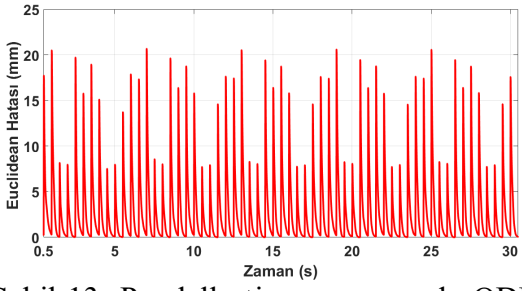
Şekil 11. Paralelleştirme sonucunda ODE tabanlı tripod yürüyüş düzenine ait zamana bağlı hata değişimi (5 adım).

Şekil 12'de ODE tabanlı ripple yürüyüş düzenine paralelleştirme sonucunda elde edilen yeni kontrol parametrelerinin uygulanması durumunda elde edilen euclidean hatasının zamana bağlı değişim grafiği verilmektedir.



Şekil 12. Paralelleştirme sonucunda ODE tabanlı ripple yürüyüş düzenine ait zamana bağlı hata değişimi (5 adım).

Şekil 13'de ODE tabanlı wave yürüyüş düzenine paralelleştirme sonucunda elde edilen yeni kontrol parametrelerinin uygulanması durumunda elde edilen euclidean hatasının zamana bağlı değişim grafiği verilmektedir.



Şekil 13. Paralelleştirme sonucunda ODE tabanlı wave yürüyüş düzenine ait zamana bağlı hata değişimi (5 adım).

Yapılan deneylerin sonucunda parametre ayarlama öncesi ve sonrası durum için elde edilen iyileşme Tablo 2'de verilmiştir.

Tablo 2. . Yürüyüş Simülatörü ve Deneysel Robot Sonuçlarının Karşılaştırması.

	Alan Hatası (mm.s)	**Sapma Hatası**	**Hareket Hızı (mm/s)**	**Parametre Ayarlaması Öncesi Yüzde İyileşme (%)**	**Parametre Ayarlaması Sonrası İyileşme (%)**
Standart Tripod	90,127	20,082	8	66,048	78,99
ODE tabanlı Tripod	30,600	15,976	10		
ODE tabanlı Tripod*	19,102	15,360	10		
Standart Ripple	131,4181	36,291	4,44	74,930	83,84
ODE tabanlı Ripple	32,946	2,533	5		
ODE tabanlı Ripple*	22,085	2,810	5		
Standart Wave	292,446	60,761	3,08	55,645	70,72
ODE tabanlı Wave	129,714	54,704	3,33		
ODE tabanlı Wave*	86,279	55,963	3,33		
Ortalama Yüzde İyileşme				65,541	77,850

7. REFERANSLAR

[1] Quinn, M. J., & Quinn, M. J. (1994). Parallel computing: theory and practice (Vol. 2): McGraw-Hill New York.

[2]. Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., . . . Wawrzynek, J. (2009). A view of the parallel computing landscape. Communications of the ACM, 52(10), 56-67.

[3] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., & White, A. (2003). Sourcebook of parallel computing (Vol. 3003): Morgan Kaufmann Publishers San Francisco.

[4]. Raza, S. B., Meng, A. X., Krall, D. A., Abugharbieh, K. S., & Bettman, R. J. (2003). Design architecture for a parallel and serial programming interface: Google Patents.

[5] Kumar, V., Grama, A., Gupta, A., & Karypis, G. (1994). Introduction to parallel computing: design and analysis of algorithms (Vol. 400): Benjamin/Cummings Redwood City.

[6]. Quinn, M. J., & Quinn, M. J. (1994). Parallel computing: theory and practice (Vol. 2): McGraw-Hill New York.

[7] Sharma, M., & Soni, P. (2014). Comparative study of parallel programming models to compute complex algorithm. International Journal of Computer Applications, 96(19).

[8]. Yang, C.-T., Chang, T.-C., Wang, H.-Y., Chu, W. C., & Chang, C.-H. (2011). Performance comparison with OpenMP parallelization for multi-core systems. Paper presented at the Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on.

[9]. Chandra, R. (2001). Parallel programming in OpenMP: Morgan kaufmann.

[10]. Yıldırım, Ş., Arslan, E.: ODE (Open Dynamics Engine) based stability control algorithm for six legged robot. Measurement, 124, 367-377 (2018).

[11]. Arslan, E., Yıldırım, Ş.: ODE (open dynamics engine) based walking control algorithm for six legged robot. Journal of New Results in Science, 7(2), 35-46 (2018).

[12] YILDIRIM, Ş., Arslan, E.: A Comparison of Six Legged ODE (Open Dynamics Engine) Based Gait Control Algorithm and Standard Walking Gaits. Avrupa Bilim ve Teknoloji Dergisi, 242-255 (2019).

[13]. Smith, R. (2007). Open dynamics engine,Smith R., 2007, http://www.ode.org (accessed on 28 June 2020).