ÖZET

Yüksek Lisans Tezi

GALERKİN TABANLI AYRIK ELEMANLAR METODU PARÇACIKLARININ BİRLEŞİMİNLERİNDE DALGA YAYILIMI

Burak ER

Uludağ Üniversitesi
Fen Bilimleri Ensitüsü
Makine Mühendisliği Anabilimdalı

Danışman: Prof.Dr. Osman KOPMAZ

Dalga yayılımı ve onun kırılma, kopma gibi yan sonuçları titreşimli yüklemeleri haiz makineler ve binalar gibi mühendislik yapılarında önemli bir olgudur. Karmaşık şekilli parçalarda dalga yayılımı sayısal incelemesi için tecrübe edilmiş en iyi yöntemlerden biri Sonlu Elemanlar Metodu'dur(SEM). Ancak SEM, dalga yayılımı sonucu gelişen, kırılma gibi süreksizliklerin meydana geldiği problemlerde uygun bir yöntem olmadığından, bu gibi problemlerde Ayrık Elemanlar Metodu(AEM)' nu kullanmak daha uygundur.

Bu çalışma kapsamında, Stuttgart Üniversitesi, Teknik ve Nümerik Mekanik Enstitüsü bünyesinde geliştirilen ve yeni bir yaklaşım olan Galerkin-tabanlı Ayrık Elemanlar Metodu Parçacıklarının Birleşimleri'nde dalga yayılımı incelenmiştir. Dalga yayılımı simülasyonlarının gerçeklenmesi için, aynı enstitü tarafından geliştirilen AEM programı Pasimodo kullanılmış ve simülasyonların mümkün olabilmesi için programa bir eklenti(plug-in) yazılmıştır. Geliştirilen metot, AEM tabanlı olup, birbirinden bağımsız ve deforme olabilen elemanların yine deforme olabilen ancak kütlesiz sanal bağlar ile birbirine bağlanması ve gerektiğinde bağların kolaylıkla kaldırılmasıyla süreklilikten süreksizliğe geçişi modellemeyi amaçlamaktadır. Bu haliyle kırılma mekaniğinde kullanılan yapışkan bölge elemanları metodunun özel bir hali olarak düşünülebilir.

Çalışma bünyesinde, dalga yayılımı karakteristiklerinin incelemesi ince alüminyum bir çubuk üzerinde hem boyuna hem de enine dalga oluşturacak şekilde bir uçtan impuls uygulanmak suretiyle gerçeklenmiştir. Uygulamayla metotta kullanılan bağlayıcının katılığının oluşan dalga yayılımına ve nümerik dispersiyona etkisi farklı ağ yoğunlukları kullanılarak incelenmiştir. İnceleme sonucu görülmüştür ki bağlayıcı katılığı alüminyum elastisite modülünün 100 katı ve üstü değerlerde alınırsa sonuçlar tüm ağ yoğunlukları için analitik çözüme yaklaşmaktadır.

Anahtar Kelimeler: ayrık elemanlar metodu, ince çubuklar, dalga yayılımı, sonlu elemanlar metodu
2013, viii + 64 sayfa.

ABSTRACT

MSc Thesis

WAVE PROPAGATION IN GALERKİN BASED DEM PARTICLE ASSEMBLIES

Burak ER

Uludağ Üniversity
Graduate School of Natural and Applied Sciences
Department of Mechanical Engineering

Supervisor: Prof.Dr. Osman KOPMAZ

Wave propagation and its side effects such as; fracture and failure; are important phenomena in vibrationally excited engineering constructions like machinery and buildings. Finite Element Method(FEM) is the well-known method for the numerical investigation of the wave propagation on complicated shaped structures. Nevertheless, FEM is not a suitable method for the analysis of discontinuities as a result of processes like fracture. Therefore, in this type of problems, using the Discrete Element Method(DEM) is more advantageous.

In this study, wave propagation in thin rods using the method of Galerkin-based DEM Particle Assemblies that is developed within the Institüt für Technische und Numerische Mechanik, Universität Stuttgart is investigated. To this end, the DEM simulation platform Pasimodo(Particle Simulation and MOlecular Dynamics In an Object Oriented fashion) which is developed again within the same institute is used and in order to make the simulations possible a plug-in is written. The developed method is based on the Discrete Element Method(DEM) and aims to model transition from continuity to discontinuity using independent deformable elements that are bonded together with the fictious bonding elements which are massless and deformable and easily deleted in a fracture condition. Therefore, this method can be considered as specific type of the cohesive zone elements that are used in fracture mechanics.

In the scope of this work, investigation of the wave propagation characteristics is done using a thin aluminium rod and using an impulse on the one end with exciting both longitudinal and flexural propagation. With the application, both wave propagation characteristics and the numerical dispersion are studied with the changing bonding element stiffness and the mesh density. From the results, it is seen that the results are nearly same with the analytical solution above the bond stiffness that is equal to and above 100 times the aluminium's Youngs modulus.

Key words: discrete element method, thin rods, wave propagation, finite element method
2013, viii + 64 pages.

ÖNSÖZ

Bu çalışma, Stuttgart Üniversitesi Teknik ve Hesaplamalı Mekanik Enstitüsü bünyesinde yürütülen ve Alman Araştırma Kuruluşu tarafından desteklenen SPP 1486 "Partikel im Kontakt" projesi kapsamında, "Dynamische Simulation hochelastischer, hochfeiner, nichtkonvexer und polydrischer disperser Festoffe unter Einbeziehung von Adhasion un Teichendruck" başlıklı alt proje bünyesinde gerçekleştirilmiştir.

Çalışma, bir çok aşamasında ücretsiz ve tamamen açık kaynak kodlu uygulamalar kullanılarak gerçekleştirilmiştir. Parçacık similasyonları için Pasimodo, kodlama ve derleme için Eclipse, Gedit ve GCC, çizim işlemleri için Blender ve Inkscape, yazım işleri için ise Kile programları kullanılmıştır. Tüm uygulamalar debian işletim sistemi altında çalıştırılmıştır.

Projenin gerçekleşmesinde emeği geçen bir çok kimseye teşekkür etmeyi bir borç bilirim; Proje kapsamında bana çalışma yapma fırsatı sağlayan, Stuttgart Üniversitesi Teknik ve Hesaplamalı Mekanik Enstitüsü Direktörü, Prof. Dr.-Ing. Prof. E.h. Peter Eberhard' a ve enstitütede bulunduğum sürece danışmanlığımı yapan Dipl.-Ing. Sven Stühler'e, YÖK yurt dışı yüksek lisans araştırma bursu kapsamında böyle bir projeye katılmamı sağlayan ve çalışma süresince benden her türlü desteğini esirgemeyen çok değerli hocam ve danışmanım Prof. Dr. Osman Kopmaz' a teşekkürlerimi arz ederim.

Ve, her zaman yanımda olan aileme sonsuz teşekkürlerimi sunarım.

Burak ER
Haziran 2013

İÇİNDEKİLER

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
AEM	Ayrık Elemanlar Metodu
FFT	Fast Fourier Transformation
SEM	Sonlu Elemanlar Metodu
XML	X Markup Language

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

1. GİRİŞ

Günümüzde hesaplama teknolojisinin gelişimiyle beraber, çok parçacıklı sistemler süreksiz ortamlar için sayısal hesaplama yöntemlerinden biri olan Ayrık Elemanlar Metodu(AEM) (Cundall ve Strack 1979) son yıllarda oldukça önem kazanmıştır. Geçmişte hesaplanması hayal dahi edilemeyen binlerce parçacık içeren problemler, AEM metodu ve yüksek hesaplama kabiliyetiyle birlikte günümüz imkanları dahilinde çözülebilir hale gelmiştir.

AEM'nin teorik kökleri Newton'un çalışmalarına dayanmaktadır. İlk olarak granüler küresel şekilli parçacıkların dinamiği problemleri için Cundall tarafından (Cundall ve Strack 1979) geliştirilmiş bir metot olup (Cundall ve Strack 1979), daha sonra Mustoe (Mustoe 1992) tarafından esnek ve keyfi geometriye sahip parçacıklar için genelleştirilmiş ve son çalışmalar ile beraber başta Munjiza ve Owen'nın (Munjiza 2004) çalışmaları olmak üzere, süreksiz deformasyon analizi için sonlu elemanlar ile birlikte bütünleşik bir metot olarak geliştirilmektedir.

Ayrık elemanlar metodunun en yalın hali granüler parçacıkların yalnızca kütleye ve sabit bir geometriye sahip ayrık parçacıklar olarak modellenmesiyle oluşturulmuştur. Ancak, parçacıkların bu modeli zaman içinde soyutlanarak ayrık nesneler olarak modellenen dinamik problemlerin hepsini kapsayacak şekilde bir form almıştır. Dolayısıyla AEM güncel haliyle genel soyut bir yaklaşımı ifade etmektedir. Yaklaşımın gelişimi ile birlikte AEM için soyutlama sağlayan hesaplama kodları da geliştirilmiştir (Šmilauer ve ark. 2010).

Mühendislik problemlerinin hemen hemen hepsi sürekli ortam olarak modellenebilecek sistemlerden oluşmaktadır(gerilme problemleri, akış problemleri, ısı problemleri vb.). Ancak, bazı problemlerin ortam süreksizliğinin önemli ölçüde olması sebebiyle sürekli ortam olarak modellenmesi mümkün olmamaktadır. Örneğin; kum zerrelerinin hareketi. Bu hareket yer yer yoğunlaşmış kütlelerin oluşturduğu bir ortamın hareketi olarak tarif edilebilir. Kütleler birbirleri ile etkileşimleri olmaması sebebiyle tamamen ayrıktırlar. Bu tarz süreksiz bir ortamın analitik modellemesi mümkün olmadığından nümerik yöntemler

kullanılmaktadır. Bunlardan en genel yaklaşıma sahip yöntem de Ayrık Elemanlar Metodu'dur.

Sürekli ortam olarak modellenebilecek bir diğer mühendislik problemlerinden biri de bu çalışmanın konusu olan dalga yayılımı problemleridir. Dalga yayılımı ve onun kırılma, kopma gibi yan etkileri titreşimli yükler altıdan çalışan makineler, binalar gibi mühendislik yapılarının analizi açısından büyük önemi haizdir. Dalga yayılımı problemi yan etkileri ile beraber incelendiğinde yalnızca sürekli ortam modeli uygun olmamaktadır. Örneğin; çatlak içeren bir beton kiriş üzerinde ilerleyen bir dalganın çatlağı genişletmesi problemi gibi. Problem dalganın çatlağa erişinceye kadarki kısmı için sürekli bir ortam problemi olarak modellenebilecek iken dalganın çatlağı ayırması ile beraber sürekli ortamın bir bölümü süreksiz hale gelecektir. Bu nedenle, problem ne tamamen sürekli bir ortam ne de tamamen süreksiz bir ortam problemi olarak modellenebilir. Bunun sonucunda iki yöntemin de avantajlarını içeren bir sayısal yöntemin varlığının bu problemin analizini oldukça kolaylaştıracağı açıktır.

Bu çalışmada sürekli ve süreksiz ortamların tarifini aynı metot üzerinde mümkün kılacak, Ayrık Elemanlar Metodu ve Sonlu Elemanlar Metodu'nun birleşimi bir yöntem olan ve SEM'in AEM içerisine gömülmüş hali olarak nitelendirilebilecek Galerkin-tabanlı AEM Parçacıklarının Birleşimi metodunun incelemesi yapılmıştır. Metod, kırılma mekaniği problemlerinde kırılmanın gerçekleşmesi muhtemel bölgelerinde kullanılan yapışkan elemanların(cohesive elements), yalnızca kırılma bölgesinde değil diğer tüm elemanların bağlanmasında kullanılması ile oluşturulmuştur. Böylelikle model üzerinde kırılmaya müsait olan veya olmayan her yer tek bir yaklaşımla modellenebilmektedir.

1.1 Ayrık Elemanlar Metodu Prensipleri

Ayrık elemanlar metodu, genelleme yapılacak olursa, parçacıklar ve onların arasındaki etkileşimlerden ibarettir. Bu sebeple metodun ana bileşeni parçacık formülasyonları ve onların aralarındaki etkileşimlerin formülasyonlarıdır. AEM metodu parçacıkları rijit veya esnek olacak şekilde modellenirken, elemanların arasındaki etkileşimler de parçacıkların formülasyonuna göre şekillendirilir. Rijit ve basit şekilli parçacıklar için

formülasyonlar gayet basit iken esnek ve karmaşık şekilli parçacıklar için daha zor bir hal almaktadır.

1.1.1 Rijit parçacık tabanlı AEM

Rijit parçacık tabanlı AEM, AEM'nin ilk olarak ortaya atıldığı en temel halidir ve yalnızca rijit parçacıkların birleşimi ile tanımlanabilecek süreksiz sistemlerin analizi için kullanılır (Cundall ve Strack 1979).

1.1.1.1 parçacık dinamiği

AEM ile rijit parçacıkların dinamiği 6 serbestlik dereceli rijit bir cismin hareket denklemlerini ifade eden

$$m\dot{\mathbf{v}} = \mathbf{F} \tag{1.1}$$

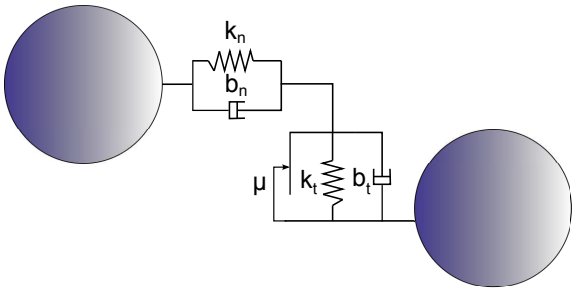
$$\mathbf{I}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) \tag{1.2}$$

Euler-Newton hareket denklemleri ile temsil edilir. Burada $\mathbf{v}$ ve $\boldsymbol{\omega}$ sırasıyla parçacık için dönme ve ötelenme hızları, m ve $\mathbf{I}$ kütle ve kütle atalet momentleri, $\mathbf{F}$ ve $\boldsymbol{\tau}$ dış kuvvet ve torku ifade etmektedir.

Cisimlerin hızlarının sayısal integrasyonu için genellikle parameterizasyon yapılır. Parameterizasyona doğrusal hızlar için gerek olmamakla beraber açısal hızların integrasyonu Euler Parametreleri ile açısal yönelim tarif edilir (Fleissner 2010).

1.1.1.2 parçacık çarpışma etkileşimi

Rijit parçacık tabanlı AEM' de en temel etkileşim küresel şekilli elemanların birbirleri ile çarpışmasıdır. Çarpışma durumunda parçacıklar arasına parçacıkların hem ötelenme hem de dönme hareketlerinin ikisini de kapsayacak şekilde yay ve sönümleyici konarak çarpışma prosesi modellenir. Model Şekil 1.1'de temsil edilmiştir.



Şekil 1.1. Rijit küresel parçacıklar arasında çarpışma etkileşimi

Parçacıkların çarpışma etkileşimleri rijit AEM metodunun sayısal hesaplama bakımından en zorlayıcı kısmıdır. Zaman adımı, integrasyon toleransı gibi değerler etkileşim özelliklerine bire bir bağlıdır. Örneğin, teğetsel etkilerin ihmal edildiği ve yalnızca doğrusal etkilerin göz önüne alındığı bir etkileşim modelinde parçacıkların birbiri üzerine nüfûzu(penetrasyonu)

$$\ddot{\delta} + \overline{M} b_n \dot{\delta} + \overline{M} k_n \delta = 0 \qquad \textbf{(1.3)}$$

denklemiyle yazılabilir. Burada, δ, parçacıkların normal doğrultudaki penetrasyonu ve M indirgenmiş kütle olup $\frac{m_1+m_2}{m_1 m_2}$'ye eşittir. Denklem 1.3 etkileşim yay rijitliğinin seçimi için önemli bir yere sahiptir. Etkileşimdeki yayların rijitliği, parçacıkların birbiri üzerine nüfûzları, parçacık yarıçaplarına kıyasla çok küçük miktarlarda olacak şekilde seçilirken, simülasyonlar için en büyük zaman adımı da böylece belirlenmiş olur.

1.1.2 Genelleştirilmiş parçacık tabanlı AEM

Genelleştirilmiş parçacık tabanlı AEM'de parçacıklar tek bir formül ile rijit veya esnek parçacıklar olacak şekilde genelleştirilmiştir. Bu yöntem Mustoe (Mustoe 1992) tarafından geliştirilmiştir.

1.1.2.1 parçacık dinamiği

Genelleştirilmiş AEM parçacıkları dinamiği ağırlıklı artıklar yöntemi ile formülize edilmektedir. Lineer elastodinamik model için ağırlıklı artıklar yöntemi

$$\int_V \boldsymbol{W}_m(\boldsymbol{x}) \cdot (\rho \ddot{\boldsymbol{u}} - \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} - \boldsymbol{F}) \, \mathrm{d}V = 0 \quad m = 1, 2, ..., n \qquad \textbf{(1.4)}$$

olarak yazılmaktadır. Burada, $\mathbf{W}_m$, konuma bağlı ağırlıklı artıklar fonksiyonları, $\boldsymbol{u}$, yer değiştirmeler, $\boldsymbol{\sigma}$, simetrik gerilme tensörü ve $\boldsymbol{F}$, cisim kuvvetidir. Yer değiştirmeler, bir

fonksiyon ailesi kullanılarak yaklaşık olarak hesaplanmaktadır ve yaklaşım için

$$\boldsymbol{u} = \sum_{l=1}^{n} \boldsymbol{\phi}_l(x,y,z) q_l(t) \tag{1.5}$$

ifadesi kullanılmaktadır. Burada $\boldsymbol{\phi}_l$ n adet taban fonksiyonları ve $q_l(t)$ de zamana bağlı katsayı fonksiyonlarıdır.

Yer değiştirmeler için kullanılan taban fonksiyonlarının farklı formları için rijit, sonlu eleman, dirac yaklaşımı gibi farklı parçacık dinamik formülasyonları elde edilir. Rijit olan parçacıklar, taban fonksiyonlarının koordinatlara göre değişmeyip sabit olduğu formu ile elde edilirken, sonlu elemanlar formülasyonu ağırlık fonksiyonları ile taban fonksiyonların aynı seçildiği halde elde edilir (Mustoe 1992).

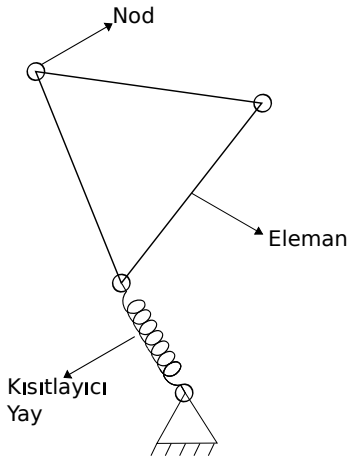
1.1.2.2 parçacık çarpışma etkileşimleri

Genelleştirilmiş AEM parçacıklarının çarpışma etkileşimleri için tek bir yöntem yoktur ve eleman formülasyonu ve şekline göre şekillendirilir. Örneğin; rijit küre şekilli parçacık formülasyonlu AEM için Bölüm 1.1.1.2'de tanımlanan model kullanılırken, sonlu elemanlar formülasyonlu parçacıklar için Lagrange çarpanları metodu, penaltı metodu gibi metotlar kullanılmaktadır (Wriggers 2006).

1.1.3 Galerkin-tabanlı AEM Parçacıklarının Birleşimleri Yöntemi

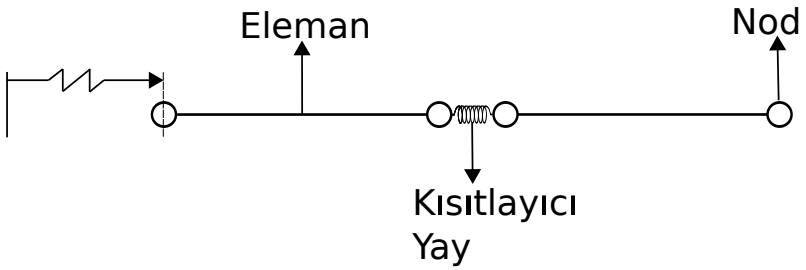
Genelleştirilmiş parçacık tabanlı AEM metodu, her ne kadar genel bir parçacık formülasyonu kullansa da, parçacıkların birbirinden tamamen bağımsız olması sürekli ortamların simülasyonu için gerekli olan bütünlüğü sağlamaz. Ancak bütünlük, parçacıkların kendi içerisinde sürekli ortam olarak modellenmesi ile sağlanabilmektedir. Bu yaklaşımın sonlu elemanlar yönteminden bir farkı yoktur.

Galerkin-tabanlı AEM Parçacıklarının Birleşimi yöntemi ise farklı bir yaklaşım olup, genelleştirilmiş AEM parçacıklarının kütlesiz deforme olabilen elemanlar ile birleştirilmesi ile elde edilen bir yöntemdir. Bu birleştirici elemanlara "bağlayıcı" denmektedir. Birleştirici elemanlar, sonlu elemanlar metodunda bağların muhafazası için kullanılan penaltı yayları(Şekil 1.2) gibi tasavvur edilebilecek elemanlar olup, sağladıkları bağ, komşu parçacıkların çakışık yüzeylerinin beraber hareket etmesi gerekliliğidir.

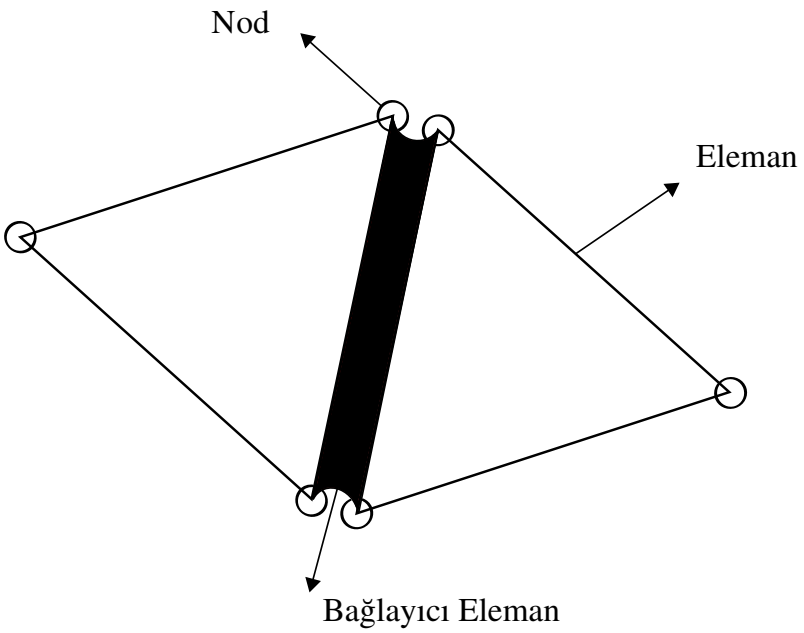


Şekil 1.2. Kısıtlayıcı yay gösterimi

Parçacıkların yüzey çakışıklılığı tek boyutlu bir yaklaşımda tek bir yay ile sağlanırken (Şekil 1.3), daha fazla boyutlu bir yaklaşımda yüzey boyunca tanımlanmış elastik bağlayıcı eleman ile sağlanmaktadır(Şekil 1.4).



Şekil 1.3. Tek boyutlu yaklaşımda Galerkin-tabanlı AEM Metodu Parçacıkları Birleşimleri



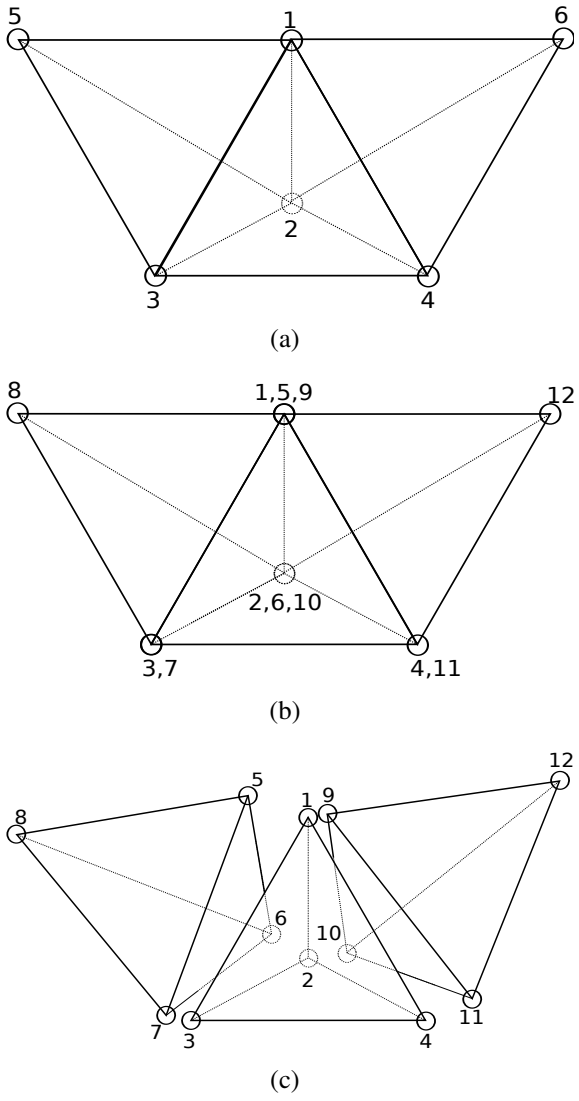
Şekil 1.4. İki ve daha fazla boyutlu yaklaşımda Galerkin-tabanlı AEM Metodu Parçacıkları Birleşimleri

1.1.3.1 Birleşimlerin Oluşturulması

Galerkin-tabanlı AEM parçacıkları birleşimlerinin oluşturulması için sonlu elemanlar metodu ile aynı olan ve farklılığın gerçekleştiği iki ana işlem bölümü ile gerçekleştirilir: İlk olarak çözüm domeni sınırları oluşturulur, daha sonra çeşitli eleman tipleri kullanılarak domen üzerinde sonlu eleman ağı oluşturulur. Buraya kadar olan yaklaşım sonlu elemanlar metodu ile aynı olan kısım olup AEM'de bu noktadan sonra elemanlar kendi nodlarına sahip ayrı ayrı elemanlar olarak oluşturulur ve yüzeyleri ortak olan elemanlar bağlayıcı eleman ile bağlanır (Şekil 1.5). Bağlayıcı elemanın ilk haldeki hacmi sıfırdır, yani, parçacıklar arasında herhangi bir kopukluk yoktur.

1.1.3.2 Avantajları

Yöntemin sürekli ortamdan süreksizliğe geçiş içeren dinamik problemlerin uygulaması için SEM'e oranla birçok avantajları bulunmaktadır. Bu uygulamalardan en önemlisi SEM kullanıldığında benzetim sırasında sürekli yeni ağ oluşturmak gereken dinamik kırılma, kopma gibi uygulamalardır. AEM kullanıldığında kırılma yalnızca aradaki bağın kaldırılması ile gerçekleştirilir.



Şekil 1.5. Sonlu elemanlar ağı (a), Galerkin-tabanlı AEM Metodu Parçacıkları gösterimi (b) ve bu metot ile simülasyon sırasında örnek bir durum (c)

AEM'nin diğer bir avantajı herhangi bir kütle matrisi içermemesidir. Sistem denklemleri baştan sona bağsız halde oluşturulur. Bunun sonucunda nodal ivmeler hesaplama süresininin kısalmasını sağlayan, sistem tabanlı matris operasyonlarının yerine eleman bazlı matris operasyonları kullanılarak bulunur. Çalışmada nodların bağımsız yoğunlaşmış kütle(lumped mass) olarak modellenmesi ile eleman bazında da matris operasyonlarına ihtiyaç kalmamıştır.

2. KURAMSAL TEMELLER

Bu bölümde, Galerkin-tabanlı AEM Parçacıklarının Birleşimlerinde dalga yayılımı problemi için gerekli olan dalga yayılımı teorisi hakkında temel bilgiler sunulmuştur.

2.1 Katılarda İlerleyen Dalgalar

Bir katıdaki ilerleyen dalganın gösterimi için tek boyutlu bir harmonik dalga göz önüne alalım. Bu harmonik dalganın ifadesi (Hauser 1965)

$$u(x,t) = sin(kx - \omega t) \tag{2.1}$$

olarak yazılabilir. Burada k dalga numarası, ω dalganın frekansı, x uzay koordinatı ve t zamandır. Böyle bir ifadeye sahip olan bir değişkenin ilerleyen bir yapıya sahip olduğunu göstermek için harmonik ifadenin uç kısmını almak yeterlidir. Uç kısım dalganın önünü ifade etmekte olup ifadesi için

$$sin(kx - \omega t) = 0$$

dolayısıyla

$$kx - \omega t = 0 \tag{2.2}$$

bulunur. İfadeden dalga ucunun konumunun zamana göre değiştiği görülmektedir, dolayısıyla, zamana göre değişimden dalga ucunun hızı da bulunabilir. Dalga ucunun hızı

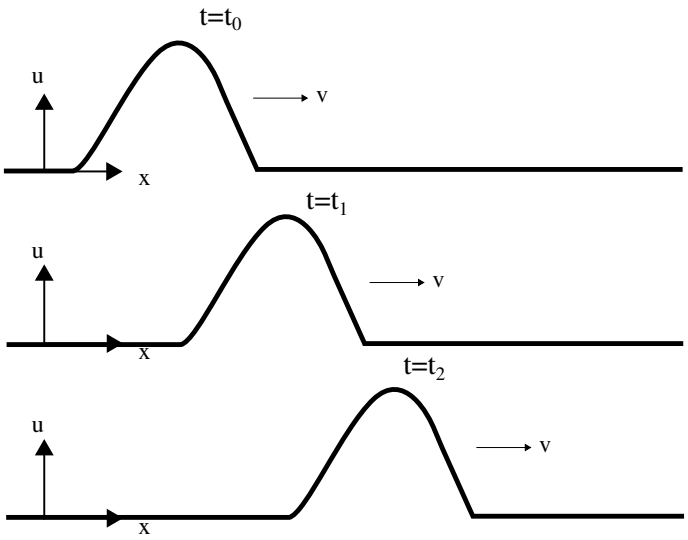
$$kx = \omega t \Rightarrow \frac{x}{t} = \frac{\omega}{k} = v \tag{2.3}$$

eşitliklerinden bulunur.

İlerleyen dalga harmonik dalga gösteriminin yanında herhangi bir fonksiyon için de gösterilebilir. Bunun için fonksiyonun argümanının $kx - \omega t$ cinsinden ifade edilebilmesi yeterlidir. İlerleyen herhangi bir şekle sahip dalga

$$u(x,t) = f(kx - \omega t) \tag{2.4}$$

olarak yazılabilir. İlerleyen dalga gösterimi Şekil 2.1' de verilmiştir.



Şekil 2.1. İlerleyen dalga gösterimi

2.1.1 Taşıyıcı Dalga ve Dalga Grubu

Taşıyıcı dalga veya en basit haliyle dalga, ilerleyen harmonik bir dalgayı ifade eder. İlerleyen harmonik bir dalga değişimini tanımladığı değişkene hükmeden denklemlerin yalnızca tek bir çözümünü gösterir. Münferit, ilerleyen bir dalganın dalga hızı Denklem 2.3' den

$$v = \frac{\omega}{k} \tag{2.5}$$

olarak yazılabilir.

İlerleyen dalgalarda dalga grubu ise, birbirine çok yakın, sabit bir dalga numarası ve frekanstan Δ ve $-\Delta$ kadar sapan dalga numarasına ve frekansa sahip iki dalganın birleşimleridir. Dalga grubunun matematiksel ifadesi (Rose 2004)

$$A(x,t) = 2cos(\Delta kx - \Delta\omega t)cos(kx - \omega t) \tag{2.6}$$

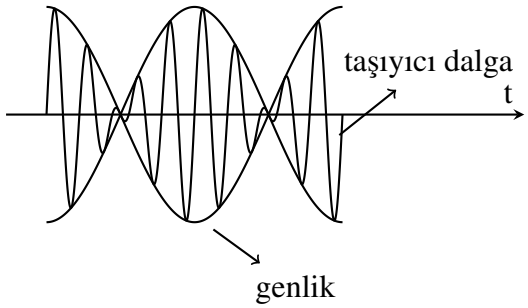
ile yazılır. Bu ifadeden görüldüğü üzere dalga grubunun genliği de ilerleyen bir dalga olmaktadır. Bir dalga grubunun gösterimi Şekil 2.2' de verilmiştir. Farkların sonsuz küçük olması halinde Denklem 2.6 ile verilen ifadeler

$$A(x,t) = 2cos(dk\,x - d\omega\,t)cos(kx - \omega t) \tag{2.7}$$

olarak bulunur. Dalga grubunun genliğinin hızı bulunmak istenirse

$$v_g = \frac{d\omega}{dk} \tag{2.8}$$

ifadesi elde edilir. Dalga grubu hızı bir dalganın enerjisinin akış hızını belirtir (Drozdz 2008) ve bu, katılarda gözlenen deplasman değişkeni için kinetik enerjidir.



Şekil 2.2. Dalga grubu gösterimi

2.1.2 Dispersiyon

Katı ortamda önemli dalga yayılımı davranışlarından biri de dispersiyondur. Dispersiyon, farklı frekanslardaki dalgaların farklı hızlar ile ilerlemesi olayıdır ve önemli sonuçlarından biri dalganın ilk halini muhafaza edemeyerek saçılma göstermesidir. Bu nedenle, dispersif bir ortamda herhangi bir şekle sahip dalga zaman içerisinde şeklinin muhafaza edememekte ve saçılmaktadır. Böylece dispersif bir ortamda dalga enerjisi yolu üzerindeki bir noktaya parçalı enerjiler olarak ulaşır.

Bir ortam için dispersiyon veya saçılma karakteristikleri, ortama hükmeden denklemlere ilerleyen dalga çözümü önermenin sonucu dalga numarası ve frekansı birbirine bağlayan karakteristik denklemlerin elde edilmesiyle bulunur. Elde edilen karakteristik denklemler dispersif bir ortam için ω ve k cinsinden nonlineerdir (Rose 2004). Bulunan karakteristik denklemlerden ω ve k arasındaki bağıntıların çizilmesiyle dispersiyon eğrileri bulunmuş olur. Dispersiyon eğrileri malzeme karekterizasyonu için büyük önem arz etmektedir.

Dispersiyon eşitliklerinin nasıl bulunduğuna örnek olması amacıyla, ölçüleri z boyunca d, x ve y boyunca sonsuz olan serbest bir plak göz önüne alınsın. Plak için hareket denklemleri Hemholtz ayrışımı kullanılarak skaler ve vektörel iki potansiyel fonksiyonu cinsinden (Graff 1975)

$$\boldsymbol{u} = \boldsymbol{\nabla}\phi + \boldsymbol{\nabla} \times \boldsymbol{\psi} \tag{2.9}$$

$$\rho \frac{\partial^2 \phi}{\partial^2 t} = c_{11} \nabla^2 \phi \tag{2.10}$$

$$\rho \frac{\partial^2 \boldsymbol{\psi}}{\partial^2 t} = c_{44} \nabla^2 \boldsymbol{\psi} \tag{2.11}$$

olarak yazılır. Burada, $\boldsymbol{u}$, deplasman vektörü, ϕ, skaler potansiyel, $\boldsymbol{\psi}$, vektörel potansiyel, λ ve μ, Lamé sabitleri olmak üzere $c11 = \lambda + 2\mu$ ve $c44 = \mu$ 'dür.

Dispersiyon eğrilerini bulmak için, çözüm olarak $\phi = \Phi e^{i(kx-\omega t)}$ ve $\boldsymbol{\psi} = \boldsymbol{\Psi} e^{i(kx-\omega t)}$ şeklinde ilerleyen dalga formları önermek suretiyle, 2.9 denkleminde serbest sınır şartlarının kullanılması ile beraber

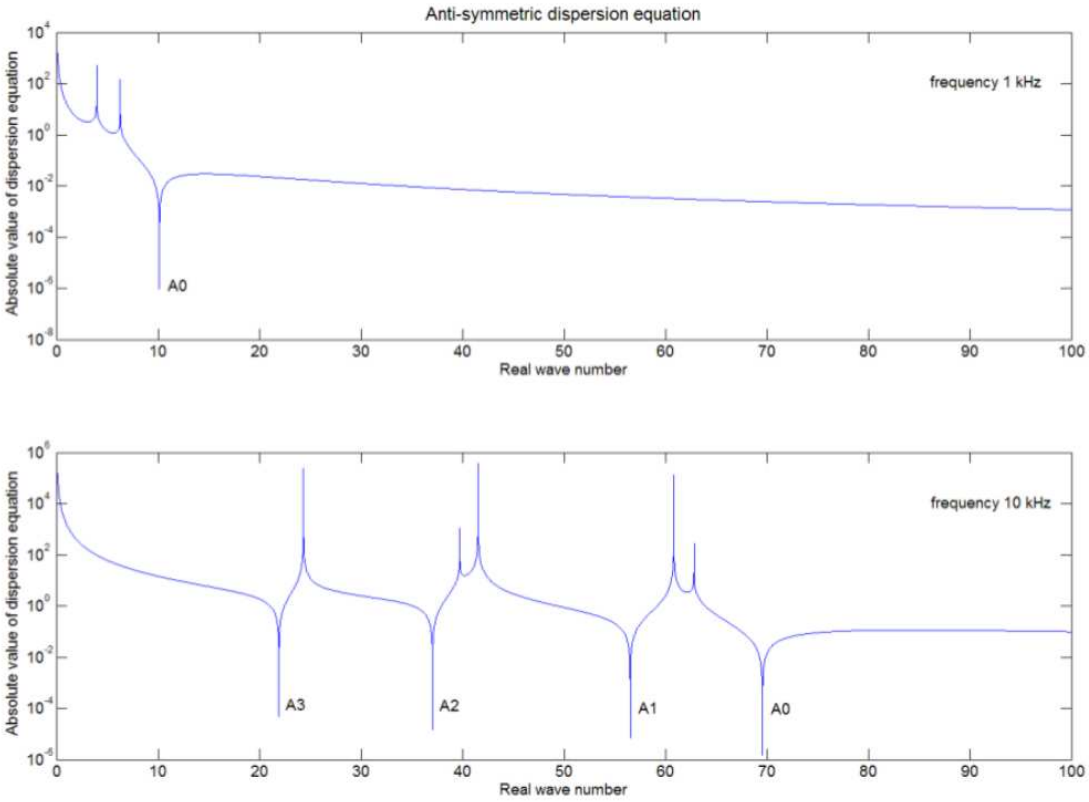
$$\frac{\tan\beta\frac{d}{2}}{\tan\alpha\frac{d}{2}} - \left[\frac{4\alpha\beta k^2}{(k^2-\beta^2)^2}\right]^{\mp 1} = 0 \tag{2.12}$$

serbest plak dispersiyon eşitlikleri bulunur. Burada,

$$\alpha^2 = \frac{\omega^2}{{V_p}^2} - k^2 \tag{2.13}$$

$$\beta^2 = \frac{\omega^2}{{V_s}^2} - k^2 \tag{2.14}$$

Denklemlerde, V_p ve V_s sırasıyla boyuna ve enine dalga hızlarıdır. Denklem 2.12' den üslü ifadenin üssüne göre iki farklı eşitlik bulunmaktadır. Bunlardan pozitif üslü eşitlik simetrik dalga hareketi çözümünü verirken, negatif üslü eşitlik antisimetrik dalga çözümünü vermektedir. Örnek bir plak için antisimetrik eşitliğin grafiği Şekil 2.3' de verilmiştir.



Şekil 2.3. Örnek bir plakta, 1kHz ve 10kHz frekansları için antisimetrik eşitliğin grafiği (Ryden ve ark. 2003)

2.2 Serbest Çubuklarda İlerleyen Dalga Yayılımı

2.3 Boyuna Dalga Yayılımı

Çubuklarda boyuna dalga yayılımı, deplasmanlar ile ilerleyen dalgaların yönünün aynı olduğu dalga yayılımı halinde görülür. En genel halde bu dalga yayılımı hareketi enine atalet etkilerinin ihmal edildiği ve edilmediği hal olarak iki ayrı durumda incelenir. Atalet etkilerinin ihmal edildiği ince çubuk profillerde boyuna dalga hareketi

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{{c_L}^2}\frac{\partial^2 u}{\partial t^2} \tag{2.15}$$

en basit haldeki dalga denklemiyle modellenir (Fung 1965). Burada c_L boyuna dalga yayılımı hızı olup $\sqrt{\frac{E}{\rho}}$, ya eşittir. Denkleme ilerleyen dalga çözümü önerilmesiyle dalga

hızı ve grup hızı

$$c_p = c_L = sabit \tag{2.16}$$

$$c_g = c_L = sabit \tag{2.17}$$

olarak bulunur.

Kalın çubuklarda boyuna dalga yayılımı, atalet etkileri göz önüne alınarak en basit halde Love teorisi ile ifade edilir ve hareket denklemi

$$\frac{\partial^2 u}{\partial x^2} + \frac{(\nu R)^2}{\rho}\frac{\partial^4 u}{\partial x^2 \partial t^2} = \frac{1}{{c_L}^2}\frac{\partial^2 u}{\partial t^2} \tag{2.18}$$

eşitliğiyle yazılır. Burada, R, kesit polar eylemsizlik yarıçapı ve ν, Poisson oranıdır. Yine ilerleyen dalga çözümü önerilmesiyle dalga hızı ve grup hızı artan dalga numarası ile birlikte düşen bir profile sahip olarak bulunur.

2.4 Enine Dalga Yayılımı

Çubuklarda enine dalga yayılımı probleminde dalganın ilerleme yönü ile yerdeğiştirmeler birbirine dik doğrultudadır. Bu dalgalara esneme dalgaları da denmektedir. Enine dalga yayılımının en basit modeli sabit bir A kesitine ve simetri eksenine göre tanımlanmış I alan atalet momentine sahip bir çubuktur. Boyuna dalga yayılımı problemine benzer olarak, enine dalga yayılımı probleminde de yalnızca eğilme enerjisinin gözetildiği Euler-Bernoulli yaklaşımı, eğilmeye ek dönme enerjisinin etkisinin de gözlendiği Rayleigh yaklaşımı ve hem dönme, hem eğilme hem de kayma enerjisinin gözlendiği Timoshenko yaklaşımı gibi farklı yaklaşımlar mevcuttur (Sadd 2009).

Euler-Bernoulli yaklaşımında çubuk model

$$\frac{\partial^4 u}{\partial x^2} + \frac{1}{{c_t}^2}\frac{\partial^2 u}{\partial t^2} = 0 \tag{2.19}$$

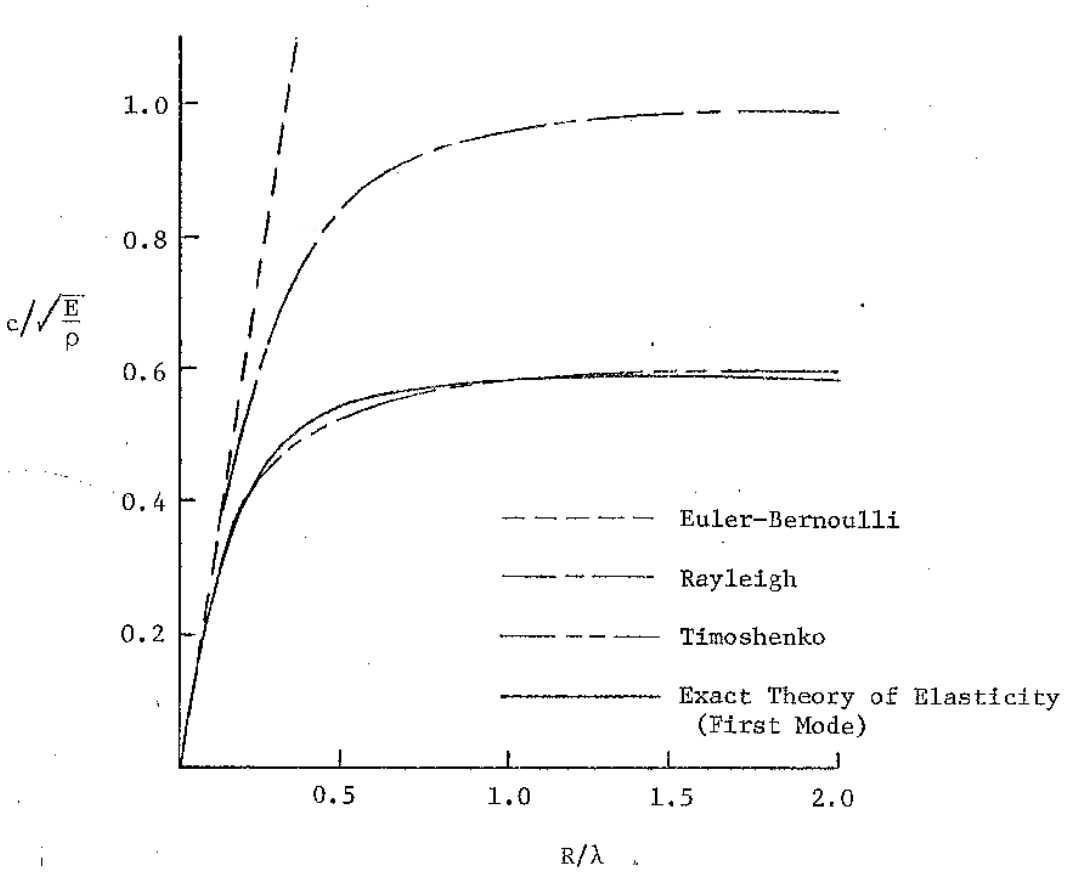
denklemiyle modellenir. Burada, c_t, enine dalga hızı olup, $\sqrt{\frac{EI}{\rho A}}$, ye eşittir. Bu model için faz ve grup hızı eşitlikleri

$$c_p = c_t k \tag{2.20}$$

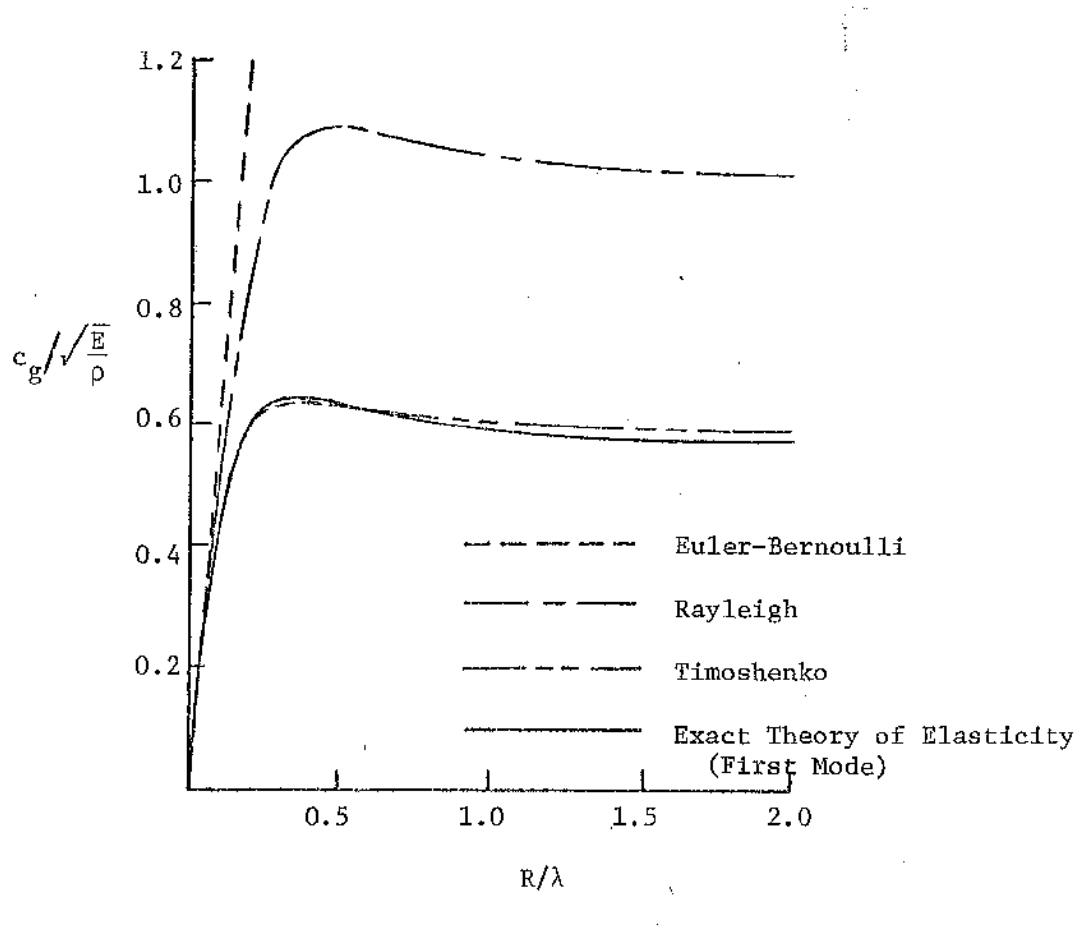
$$c_g = 2c_p \tag{2.21}$$

ifadelerinden bulunur. Görüldüğü üzere enine dalga yayılımı en basit modelde bile dispersif bir karakter mevcuttur.

Euler-Bernoulli yaklaşımı yalnızca düşük frekanslı dalga yayılımı problemleri için elastisite teorisinden bulunan çözüme yakın sonuç vermektedir. Yüksek frekanslı dalgalar için dönme ve kayma enerjileri, eğilme enerjisi ile karşılaştırıldığında önemli ölçülerdedir. Yüksek frekans aralıkları için Euler-Bernoulli yaklaşımı yanlış sonuç verirken, Rayleigh yaklaşımı daha iyi, Timoshenko yaklaşımı ise elastisite teorisi çözümüne en yakın sonucu vermektedir. Farklı yaklaşımlar ile bulunan dalga hızı ve grup hızı boyutsuzlaştırılmış halde Şekiller 2.4 ve 2.5' de verilmiştir.



Şekil 2.4. Enine dalga yayılımı halinde, farklı yaklaşımlar kullanılarak elde edilen taşıyıcı dalga hızının dalga boyuna göre değişimi (Sadd 2009).



Şekil 2.5. Enine dalga yayılımı halinde, farklı yaklaşımlar kullanılarak elde edilen grup hızının dalga boyuna göre değişimi (Sadd 2009).

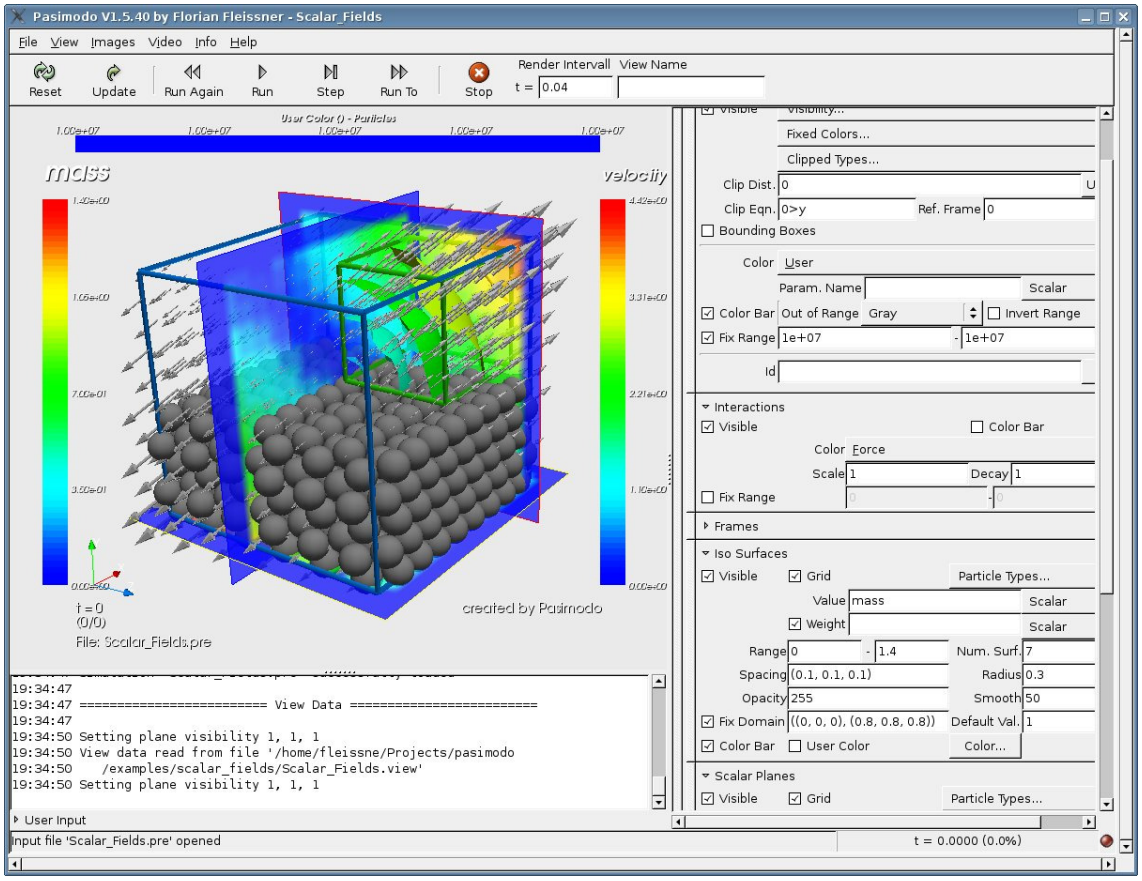
3. MATERYAL VE YÖNTEM

Bu bölümde, ilk olarak Galerkin-tabanlı ayrık elemanlar metodu parçacıklarında dalga yayılımının karakteristiklerinin tayini için kullanılan simülasyon ortamı tanıtılmıştır. Daha sonra bu simülasyon ortamında gerçekleştirilen simülasyonların yöntemleri ve konfigürasyonları hakkında detaylı bilgi verilmiştir.

3.1 Simülasyon Ortamı: Pasimodo

Galerkin-tabanlı Ayrık Elemanlar metodu parçacıklarında dalga yayılımı simülasyonu için Stuttgart Üniversitesi Hesaplamalı Mekanik Enstitüsü tarafından geliştirilen Pasimodo(Particle Simulation and Molecular Dynamics in an Object Oriented Fashion) programı ve onun içerisine gömülen Galerkin-tabanlı ayrık elemanlar birleşimleri kullanılmıştır (Fleissner 2013).

Pasimodo parçacık tabanlı sistemlerin simülasyonunu gerçekleştirmek için geliştirilmiş bir program olup, bünyesinde klasik AEM metodunun yanında aynı zamanda Yumuşatılmış Parçacık Hidrodinamiği(YPH) simülasyonları da yapılabilmektedir. Program son kullanıcı için son derece basit bir yapı sunmakta olup simülasyon girdileri Geliştirilmiş İşaretleme Dili(XML) ile tanımlanmaktadır. Pasimodo programı görsel arayüzü Şekil 3.1'de verilmiştir.



Şekil 3.1. Pasimodo programı arayüzü

Program ile şimdiye kadar AEM ve YPH'ı temel alan birçok uygulama gerçekleştirilmiştir. Bu uygulamalar, yalnızca metotların en yalın haliyle sınırlı olmayıp, farklı yaklaşımlar ile metotlar geliştirilmiş ve ağsız yaklaşımla materyal testi (Gaugele ve ark. 2008), çalkalanan yük taşıyan araç dinamiği uygulaması (Fleissner ve ark. 2009) ve hatta metal kesme simülasyonu (Eberhard ve Gaugele 2013) gibi çok farklı alanlarda uygulamalar gerçekleştirilmiştir. Pasimodo C++ dili ile geliştirilmiş bir program olup Ayrık Elemanlar Metodunu bir arayüz(interface) olarak sunmaktadır. Arayüzü sayesinde farklı yaklaşımlar ile Ayrık Elemanlar Metodu' nu temel alan farklı metotlar türetilmesine olanak sağlamaktadır.

Programın geliştirilebilirlik özelliği ile beraber şu ana kadar program içerisine gömülü olan üç yaklaşım geliştirilmiştir. Bunlar; klasik Ayrık Elemanlar Metodu, Yumuşatılmış Parçacık Hidrodinamiği Metodu ve bu çalışma kapsamında kullanılan Galerkin-tabanlı AEM parçacıkları birleşimi metotlarıdır.

Pasimodo programı tarafından ayrık elemanlar metodu için sunulan arayüz C++ programlama dilinin avantajlarından yararlanılarak çok iyi bir şekilde oluşturulmuştur. C++ dilinin temel bileşenleri olan interface yapıları (Stroustrup 1997) pasimodo DEM arayüzünün temelini oluşturmaktadır (Fleissner 2010).

3.1.1 Bağlayıcı Formülasyonu

Pasimodo içine gömülmüş olan bağlayıcı eleman formülasyonu bağlayıcı elemanın sonlu eleman gibi düşünülmesi ile gerçeklenmiştir. Bağlayıcı eleman yüzey üzerinde sürekli olarak düşünülüp bağladığı yüzeylere, yüzey kuvveti etki edecek şekilde formülize edilmiştir. Yaklaşım sonucu bağlayıcı formülasyonu tetrahedral elemanlar için

$$\boldsymbol{F}_1 = \frac{1}{12} E_b A (2\triangle \boldsymbol{r}_1 + \triangle \boldsymbol{r}_2 + \triangle \boldsymbol{r}_3) \tag{3.1}$$

$$\boldsymbol{F}_2 = \frac{1}{12} E_b A (\triangle \boldsymbol{r}_1 + 2\triangle \boldsymbol{r}_2 + \triangle \boldsymbol{r}_3) \tag{3.2}$$

$$\boldsymbol{F}_3 = \frac{1}{12} E_b A (\triangle \boldsymbol{r}_1 + \triangle \boldsymbol{r}_2 + 2\triangle \boldsymbol{r}_3) \tag{3.3}$$

olarak gömülmüştür. Burada, F_i, i. nod çiftine uygulanan bağlayıcı kuvveti, $\triangle r_i$, tetrahedral elemanların i. bağlı nod çiftinin ayrılma mesafesi, E_b, bağlayıcı katılığı ve A, elemanların bağlanan yüzeylerinin yüzey alanıdır.

3.2 Pasimodo İçin Sınır Şartı Plugini

Çalışma bünyesindeki simülasyonların gerçeklenmesi doğrultusunda Pasimodo programı içerisinde sınır şartlarını uygulamak için bir araç gerekli olduğundan, sınır şartı eklentisi program için yazılmıştır. Eklenti, C++ programlama dili ile yazılmış olup, beraberinde, Pasimodo içerisinde kuvvet sınır şartları uygulaması mümkün olmuştur.

Pasimodo' ya bir plugin yazılması için iki temel arayüz(interface) sunulmuştur. Bu arayüzler;

1. PS_Particle
2. PS_Interaction

arayüzleridir. Bu arayüzler örneğin, benzer bir AEM programı YADE için de aynı şekilde şekilde, *Body* ve *IGeomFunctor* arayüzleri ile sağlanmıştır (Šmilauer ve ark. 2010).

DEM programlarındaki arayüz yapılarından parçacık(body) temsil eden yapılar, parçacık olarak tanımlanabilecek birbirinden tamamen ayrı ve dinamik değişime sahip yapıların temsili için kullanılır. Bu yapılardan türeyen alt yapılar örneğin, 3 ötelenme serbestliğine sahip parçacık veya komple elastik bir cisim de olabilir. Diğer arayüz olan etkileşimleri temsil eden yapılar ise, parçacıkların geometrik olarak birbiri ile etkileşime girdikleri durumda etkileşimin gereği olan işleri yapan yapılardır. Buna örnek ise, iki küresel parçacığın çarpışması halinde aralarındaki fiziksel etkileşimin gerçekleşmesini sağlayan bir etkileşim arayüzüdür.

3.2.1 Sınır Şartı Plugini Yapısı

Çalışma kapsamında sınır şartları olarak yalnızca kuvvet girdisi gerektiğinden sınır şartı olarak kuvvet girdisini mümkün kılacak bir plug-in geliştirilmiştir. Plug-in, *PS_Tetrahedron* parçacıkları ile etkileşime girebilecek sanal *PS_BoundaryShape* parçacıkları, bu etkileşimde ilgili parçacığa etkileşim sonucu sınır şartı olan *PS_TetrahedronBoundary* ekleyecek olan *PS_TetrahedronBoundaryShapeInteraction* etkileşim yapısı ile sağlanmıştır. Bu yapılardan PS_BoundaryShape sınır şartının geometrisi için bir arayüz sağlarken, PS_TetrahedronBoundary diğer tüm sınır şartları tipleri için bir arayüz sağlamaktadır. Bu sınır şartlarından yalnızca yüzey kuvveti uygulayacak olan Traction_Force_Boundary_Force yapısı geliştirilmiştir. Geliştirilen yapıların Pasimodo girdi dosyasındaki tarifleri ve program arayüzü içerisindeki görüntüleri, Kod Parçası 3.1 ve Şekil 3.2'de verilmiştir. Sınır şartı kodları EK2'de verilmiştir.

Kod Parçası 3.1. Sınır şartının Pasimodo girdi dosyasındaki tarifi

```
<Boundry_Shape center="(0,0,505)" radius="20"
XwidthYheightZdepth="(20,20,10)" >
<Nested name="boundry_">
<!--Pressure Boundry-->
<Traction_Force_Boundry force="(0,0,#ForceZ)"/>
</Nested>
</Boundry_Shape>
```

Şekil 3.2. Geliştirilen sınır şartının Pasimodo içerisindeki görüntüsü

3.3 Dalga Yayılımı Simülasyonu

Galerkin-tabanlı AEM parçacıkları birleşimlerinin dinamik karakteristikleri, suni bir dalga yayılımı problemi oluşturularak incelenmiştir. Dalga yayılımının göreceli olarak basit olduğu ince, silindirik profilli sonlu boyutlara haiz bir çubuk model seçilmiş ve yalnızca bir boyutlu dalga yayılımı oluşacak şekilde bir ucundan boyuna(basınç) ve enine yüzey gerilmesi uygulanmak suretiyle tahrik edilmiştir. Tahrik amacıyla uygulanan gerilme çarpışma tabanlı bir problem üzerinden alınmıştır (Hu ve ark. 2003). Çubuğun

boyu 1m ve çapı 20mm olarak alınmış olup fiziksel parametreleri alüminyum ile aynı olacak şekilde elastisite modülü 78 GPa, yoğunluğu 2980 kg/m^3 alınmıştır. Pasimodo ile oluşturulan simülasyon modelleri Şekil 3.3'de verilmiştir.

Şekil 3.3. Dalga yayılımı simülasyonları için oluşturulmuş modeller

Metodun sonuçlarını karşılaştırmak amacıyla aynı zamanda problemin Abaqus/Explicit® programı aracılığıyla sonlu elemanlar metodu(SEM) simülasyonları da yapılmıştır. Gerçekleştirilen simülasyonlarda hem AEM hem de SEM için 4 nodlu lineer tetrahedral eleman (Hughes 2000) kullanılmıştır. Elemanın Abaqus içerisindeki karşılığı C3D4'tür.

Numerik yöntemler ile dinamik analizlerde örnekleme teoremine göre uzay adımı ve zaman adımı, analizde ortaya çıkabilecek mümkün olan tüm frekansları gözeterek seçilmelidir (Press ve ark. 2007). Adım uzunlukları sayısal yöntemlerden sonlu farklar için analitik olarak hesaplanabilirken, SEM gibi metotlar için tecrübe edilmiş tavsiye edilen amprik ifadelerden bulunmaktadır. Galerkin-tabanlı AEM Parçacıkları Birleşimleri'nin SEM ile benzerliği sebebiyle gerçekleştirilen simülasyonlar için, SEM dinamik analizlerinde tavsiye edilen eleman boyu ve zaman adımı aynı şekilde

kullanılmıştır. Tavsiye edilen adım uzunlukları

$$l_e = \frac{c_p}{20 f_{max}} \tag{3.4}$$

$$\Delta t = \frac{1}{20 f_{max}} \tag{3.5}$$

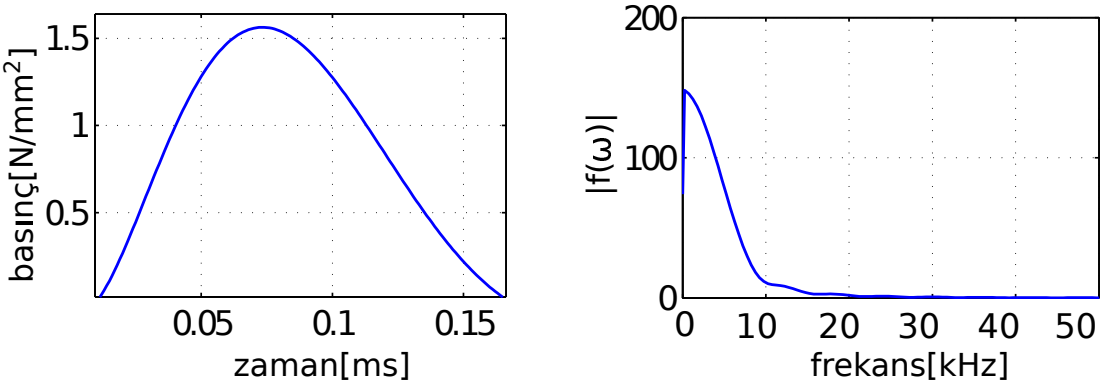
eşitlikleri ile hesaplanmıştır (F. ve ark. 1999). Burada, f_{max}, problemdeki maksimum frekans bileşeni, l_e, maksimum eleman boyu, Δt, zaman adımı ve c_p, ilerleyen bir dalganın faz hızıdır. Faz hızı, ince çubuk için boyuna dalga yayılımı probleminde sabit, enine dalga yayılımı probleminde Eşitlikler 2.20 ve 2.21 ile verildiği üzere değişken değere sahiptir.

3.3.1 Zaman Domeninde İnceleme

Galerkin tabanlı Ayrık Elemanlar Metodu Parçacıkları Birleşimlerinin üzerindeki dalga yayılımı probleminin zaman domeninde incelemesi, Şekil 3.3 ile verilen simülasyon konfigurasyonu ile zamandaki profili ve frekans bileşenleri Şekil 3.4'da verilen tahrik profili kullanılarak yapılmıştır. Tahrik gerilmesi boyuna dalga yayılımı problemi için basınç olarak, enine dalga yayılımı problemi için yüzey kayma gerilmesi olarak uygulanmıştır.

Galerkin-tabanlı AEM Parçacıkları Birleşimleri metodunun en önemli bileşeni bağlayıcı eleman olduğundan, bağlayıcı elemanın çözüm üzerine etkisi SEM metoduna tek katkısı olan katılığının farklı değerleri ile incelenmiştir. Katılık iç materyal katılığının, 1, 10, 100 ve 1000 katı değerlerde alınarak Şekil 3.3 ile verilen simülasyon gerçekleştirilmiştir.

Bağ katılığının etkisine ek olarak ağ yoğunluğunun etkisi de sonuç üzerinde gözlemlenmiştir. Bunun için Çizelge 3.1'de verilen ağ yoğunluklarındaki düşük bağ katılıklarının sebep olduğu davranış gözlemlenmiştir.



Şekil 3.4. Zaman domeninde inceleme için uygulanan gerilme profili ve frekans bileşenleri(dc bileşen kaldırılmıştır).

Zaman domeninde inceleme için seçilen tahrik gerilmesinin içerdiği maksimum frekans 20 kHz'dir(Şekil 3.4). Bu frekansın similasyonu için gerekli olan minimum eleman boyu ve zaman adımı 17.5 mm ve 10^{-6}s olarak hesaplanmıştır.

Çizelge 3.1. Zaman domeninde inceleme için kullanılan ağ yoğunlukları

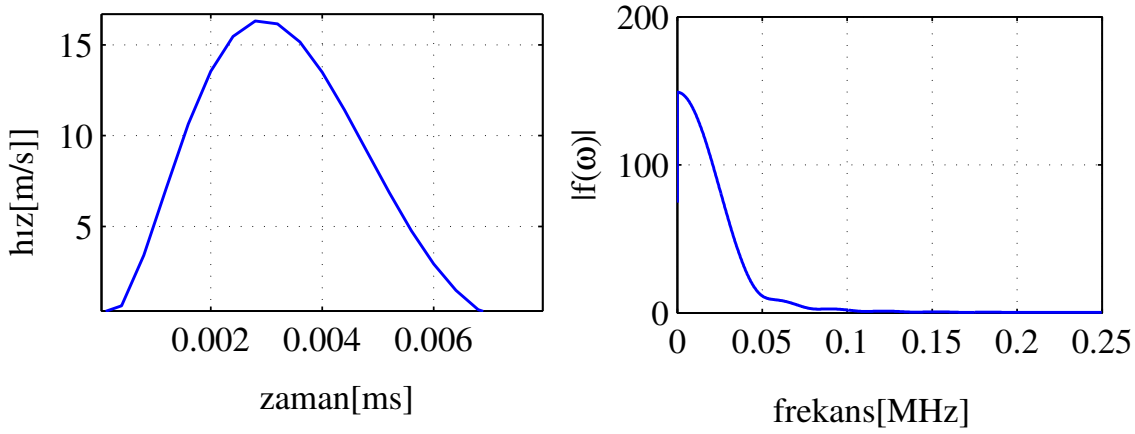
Ağ Yoğunluğu	Eleman Sayisi	Çubuk boyunca maksimum örnekleme adımı
Az yoğun	1931	15.6 mm
Yoğun	4626	7.8 mm
Çok yoğun	12227	3.9 mm

3.3.2 Frekans Domeninde İnceleme

Galerkin-tabanlı AEM Parçacıklarının Birleşimleri yöntemi nümerik bir metot olduğundan yöntemin nümerik dispersiyon özelliklerinin bilinmesi de önem arz etmektedir. Yöntemin frekans domeninde incelenmesi, dolayısıyla, bağ katılığının nümerik bir dispersiyona sebep olup olmadığı ve sonuçlar için frekasa bağlılık olup olmadığı gözlenmelidir.

Frekans domeninde inceleme yalnızca boyuna dalga yayılımı problemi için yapılmış olup, zaman domeni incelemesi için kullanılan gerilme sabit impuls uygulamak suretiyle zamanda 10 kat ölçeklendirilerek yüksek frekanslı tahrik ile gerçekleştirilmiştir(Şekil 3.5). Uygulanan gerilme ile Çizelge 3.1'deki en yoğun ağ için tavsiye edilen maksimum frekans

değerlerinin üstüne çıkılmıştır. Böylelikle metodun SEM için tavsiye edilen maksimum frekans değerlerine karşı hassasiyeti gözlemlenmiştir.

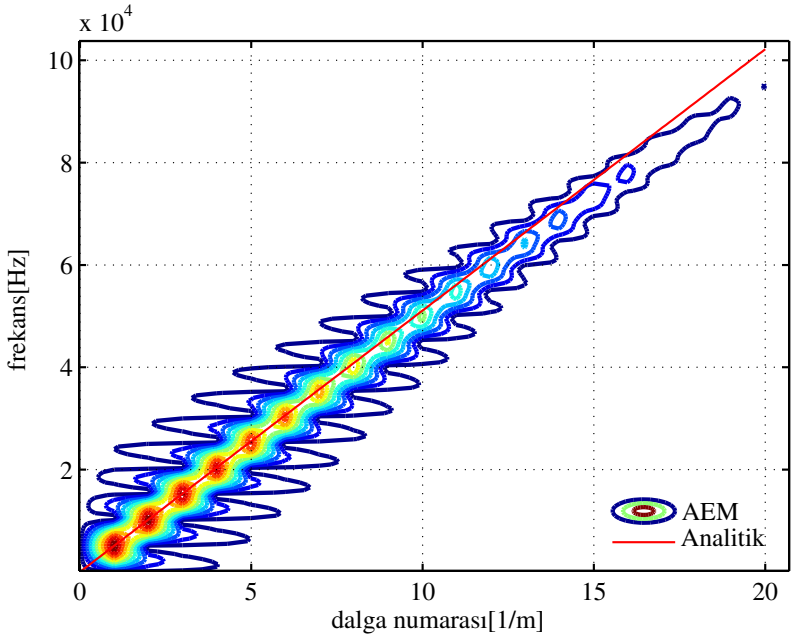


Şekil 3.5. Frekans domeninde inceleme için uygulanan yüksek frekanslı gerilme ve onun frekans bileşenleri(dc bileşen kaldırılmıştır).

İnceleme, çubuk üzerindeki noktaların hızlarının zamana ve konuma göre değişimi kullanılarak 2 boyutlu FFT metodu ile gerçekleştirilmiş olup dönüşümde konuma çubuk üzerinden alınan noktaların aynı doğru üzerinde olması gözetilmemiştir. Çubuk üzerindeki örnekleme adımı sonlu elemanlar diskretizasyonun doğası gereği sabit olmayıp değişmekte olup, interpolasyon kullanılarak tüm hızlar sabit örnekleme adımı ile yeniden örneklenmiştir (Press ve ark. 2007).

Sonuçların frekans çözünürlüğünün arttırılması ve karşı uçtan yansıyan dalgaların veriden çıkartılması amacıyla simülasyon sonuçları 2ms için alınmış ve bir dalga geçiş süresi olan 0.2ms'den sonraki veri sıfırlanmıştır. Ayrıca, frekansa katkısı olmaması sebebi ve daha rahat inceleme yapılması amacıyla rijit cisim hareketi de sonuçlardan çıkarılmıştır (de Silva 2010).

FFT metodu ile model üzerinden alınan verilerin frekans domenine dönüştürülmesi, frekans-dalga numarası domeninde genlik yüzeyleri ve onların konturlarının bulunması ile gerçekleştirilmiştir. Elde edilen konturların lokal olarak maksimum değerini aldığı ve modelin doğal frekanslarına denk gelen noktalar bulunmuş ve bu noktalar her aralıkta lineer olarak birleştirilerek frekans-dalga numarası grafikleri elde edilmiştir(Şekil 3.6).



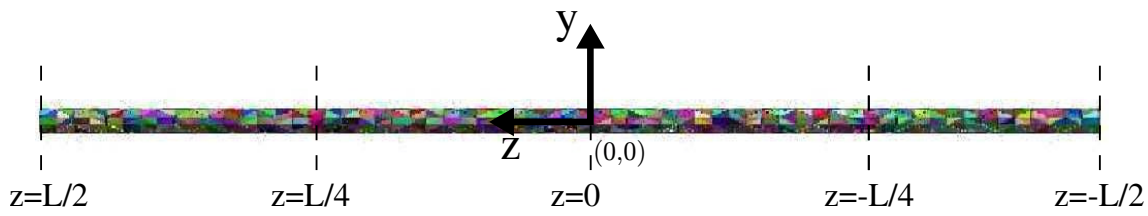
Şekil 3.6. Galerkin-tabanlı AEM ile boyuna dalga yayılımı probleminde elde edilmiş örnek bir frekans-dalga numarası konturu

4. BULGULAR

Galerkin tabanlı AEM Parçacıklarının Birleşimlerinde dalga yayılımı problemi için gerçekleştirilen simülasyonların sonuçları bu bölümde verilmiştir. Öncelikle, zaman domenindeki sonuçlar ve boyuna dalga yayılımı problemi için analitik çözüm verilmiştir. Analitik çözüm ile sonuçlar karşılaştırılmıştır. Zamana bağlı sonuçlardan sonra frekans domenindeki sonuçlar verilmiştir.

4.1 Zamana Bağlı Sonuçlar

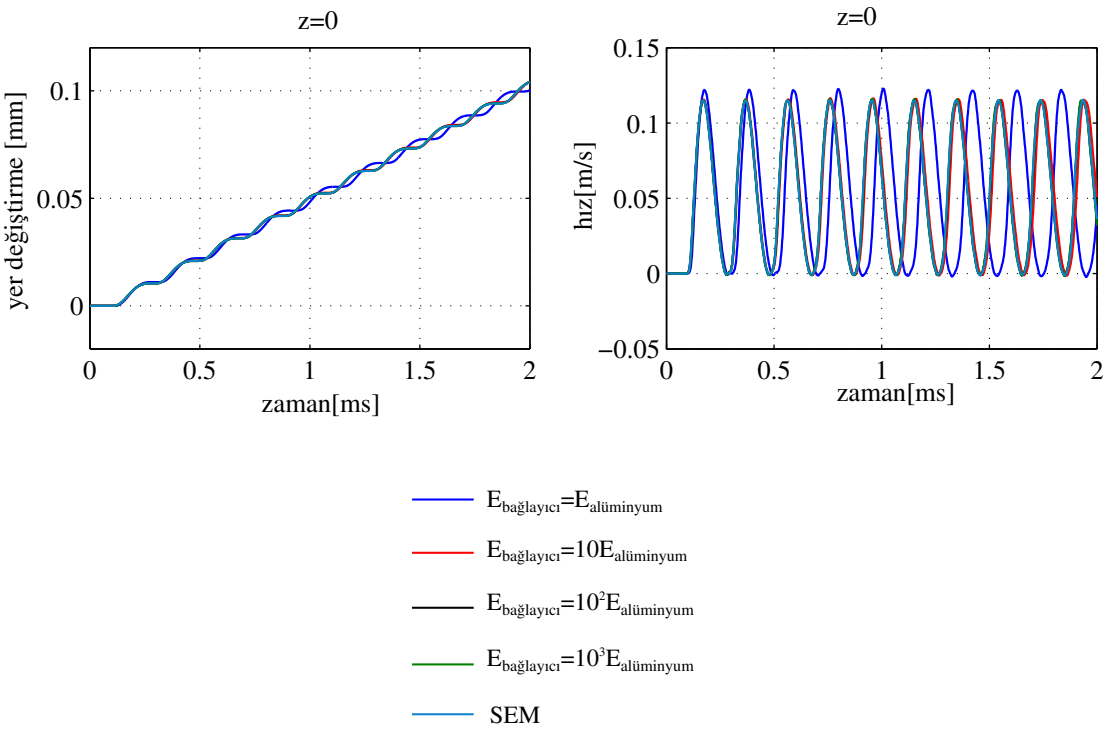
Sonuçlar, farklı bağlayıcı katılıkları ve az yoğun ağ kullanılarak çubuk üzerindeki Şekil 4.1'de gösterilen kesitler için bulunmuştur ve hem boyuna hem de enine dalga yayılımı simülasyonları için verilmiştir.



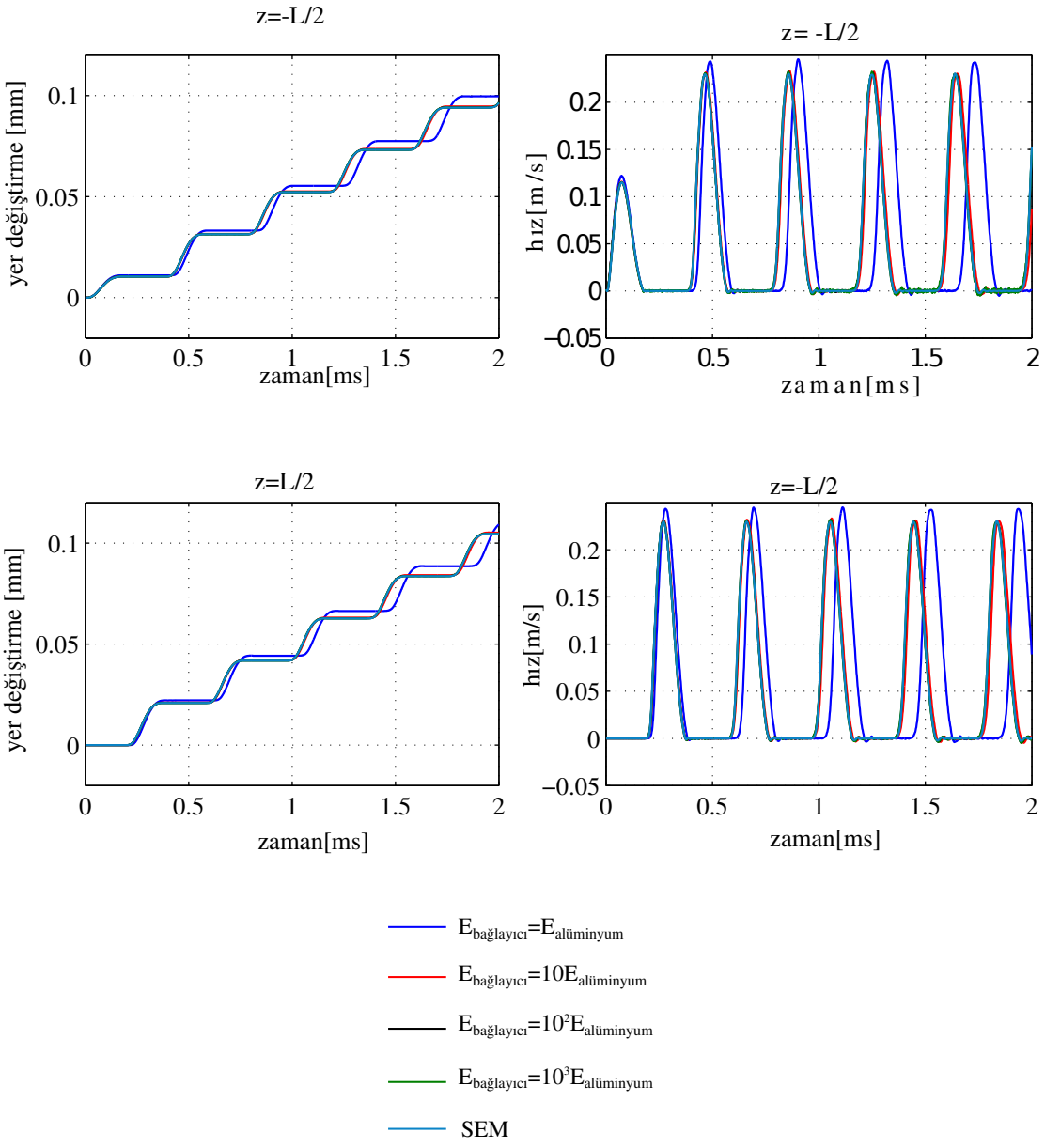
Şekil 4.1. Çubuk üzerinde sonuçların alındığı kesitler

4.1.1 Boyuna Dalga Yayılımı

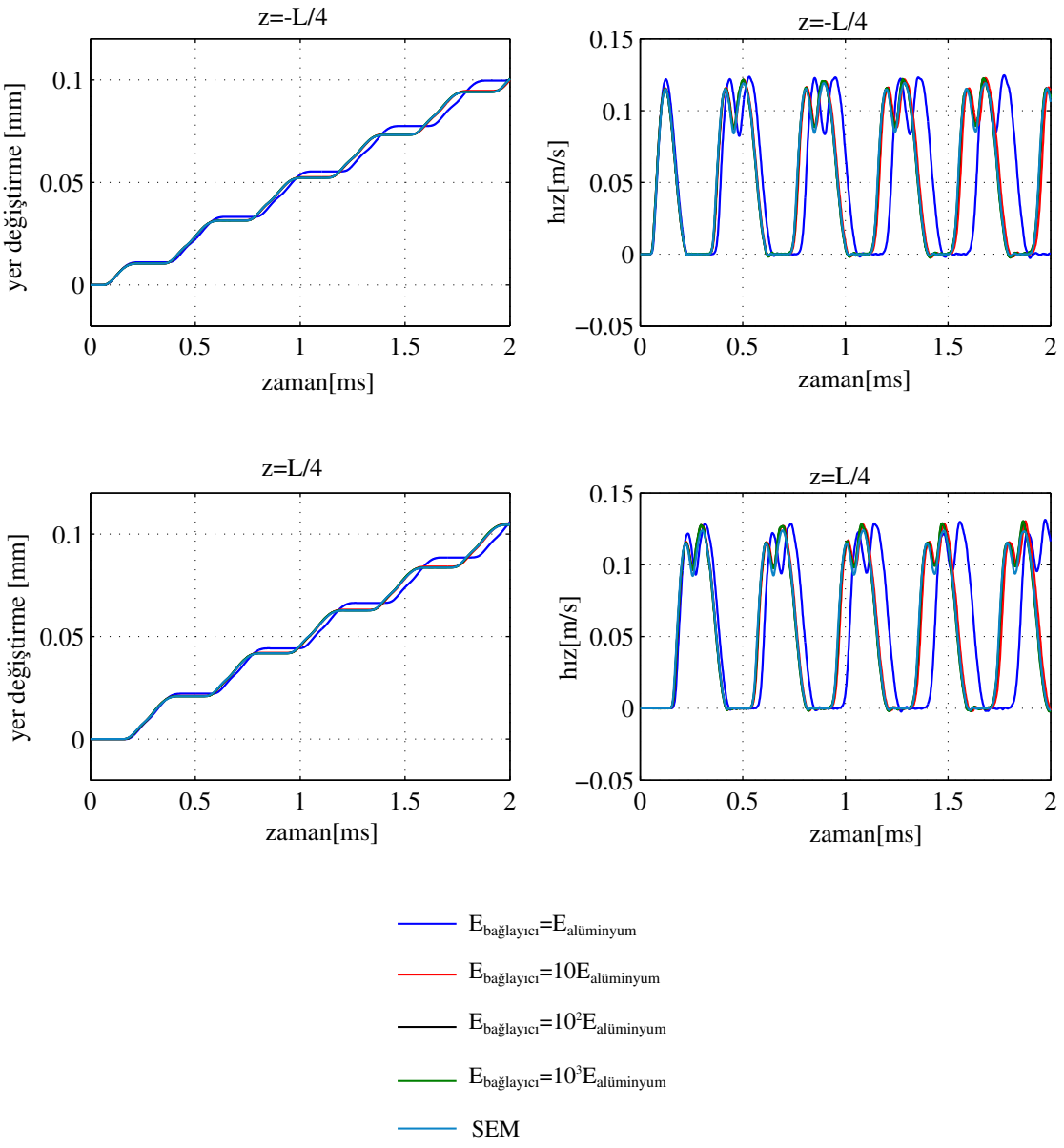
Boyuna dalga yayılımı için kesitlerin zamana bağlı hız ve deplasman grafikleri Şekil 4.2, Şekil 4.3 ve Şekil 4.4' te verilmiştir.



Şekil 4.2. Çubuğun orta kesitinde farklı bağlayıcı katılığı için deplasman ve hız



Şekil 4.3. Çubuğun uç yüzeylerinde farklı bağlayıcı katılığı için deplasman ve hız



Şekil 4.4. Çubuğun uçları ile orta kesitin arasında tam ortada yer alan kesitler için deplasman ve hız

4.1.1.1 Analitik çözüm

Basitliği açısından analitik çözüm yalnızca boyuna dalga yayılımı problemi için bulunmuştur. Şekil 3.3' de verilen simülasyon kurgusunun deplasman tahrikli sonsuz çubuk, yani, yansımasız dalga yayılımı için analitik çözümü daha önceden verilmiştir (Carrier ve Pearson 1988). Problemde deplasman tahriki yerine, gerilme

tahriği uygulanması ile problem

$$\begin{array}{ll} u_{tt} = c^2 u_{xx} & 0 < x < \infty \quad t > 0 \\ u(x,0) = 0 & 0 < x < \infty \\ u_t(x,0) = 0 & 0 < x < \infty \\ u_x(0,t) = -\frac{\sigma(t)}{E} & t > 0 \\ |u(\infty,t)| = 0 & t > 0 \end{array} \tag{4.1}$$

ifadeleri ile yazılabilir. Burada, σ, çubuğun uç noktasından uygulanan gerilmedir ve bası halinde negatif, çeki halinde pozitif işaretli alınmıştır. Problemin çözümü EK1' de verilmiş olup çözümden yer değiştirmeler ve hızlar

$$u(x,t) = \frac{c}{E}\mu(t - \frac{x}{c})Heaviside(t - \frac{x}{c}) \tag{4.2a}$$

$$\dot{u}(x,t) = \frac{c}{E}\sigma(t - \frac{x}{c})Heaviside(t - \frac{x}{c}) \tag{4.2b}$$

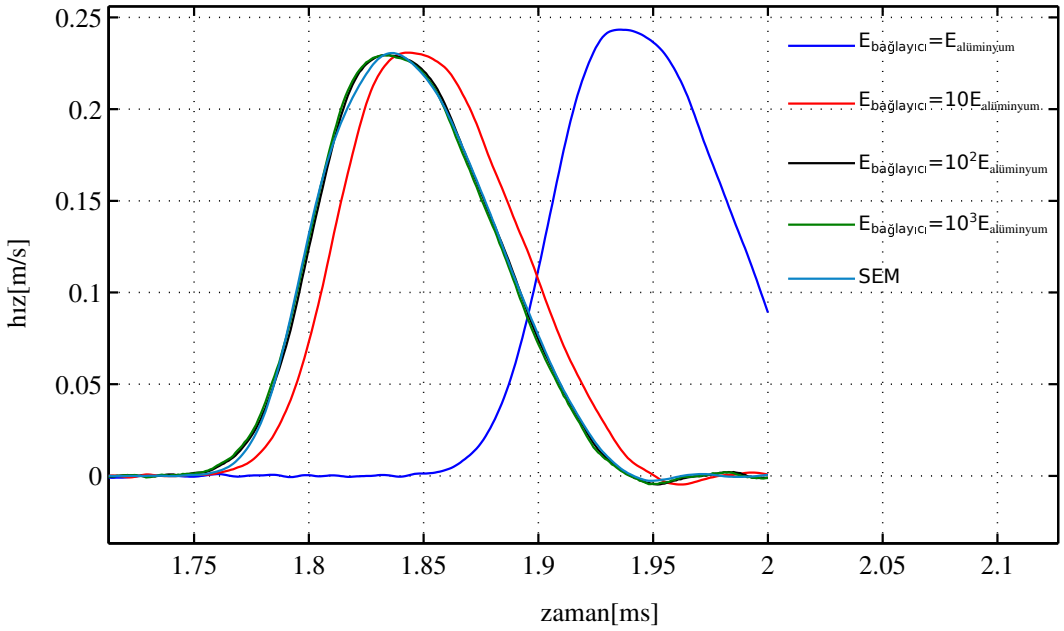
olarak bulunur. Burada,

$$\mu(t) = \int \sigma(t)\, dt$$

bulunur. Görüldüğü üzere problemde yer değiştirmeler ve hızlar genlikleri $\frac{c}{E}$' ye lineer bağlı ilerleyen dalgalar şeklindedir.

4.1.1.2 Bağ katılığı ve ağ yoğunluğunun sonuç üzerine etkisi

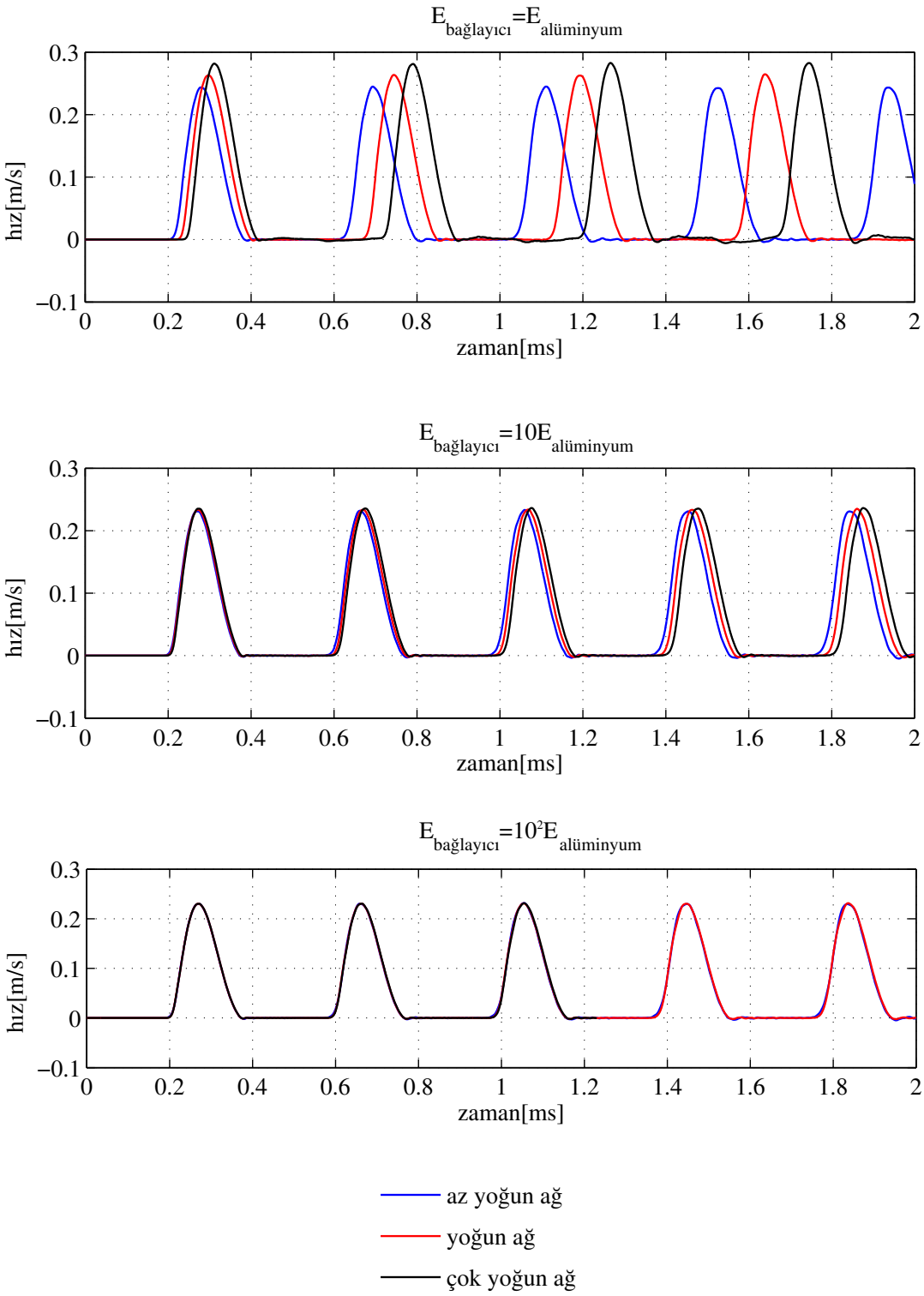
Farklı bağ katılıkları ve az yoğun ağ kullanılarak çubuk üzerindeki hızlar simülasyon süresi bitimine yakın Şekil 4.5' de verilmiştir.



Şekil 4.5. Farklı bağ katılıkları için z=L/2' de simülasyon sonuna doğru sonuçlar(az yoğun ağ)

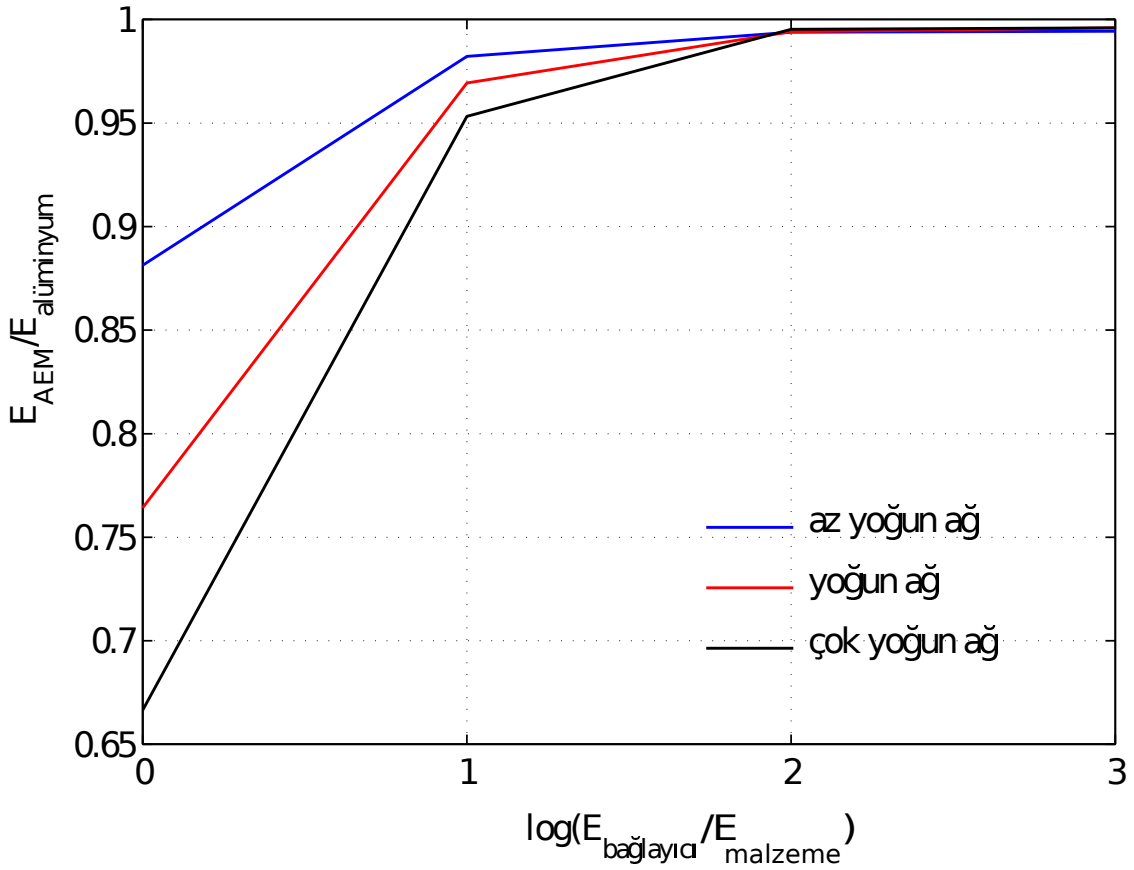
Sonuçlardan görüldüğü üzere, sabit bir ağ yoğunluğu için bağ katılığının değişimi ile iki farklı değişim gözlenmektedir. Bunlar, dalga hızı ve dalga genliğidir. Azalan bağlayıcı katılığı ile genlik artmakta ve dalga hızı düşmektedir. Bu davranış, bağ katılığının alüminyum katılığının 100 katından az değerleri için geçerlidir. Katılığın, alüminyum katılığının 100 katına eşit ve büyük değerleri için sonuç üzerinde herhangi bir etkisi gözlemlenmemektedir.

Farklı ağ yoğunlukları kullanılarak yapılan simülasyonların sonuçları her bir bağ katılığı için Şekil 4.6' da verilmiştir. Farklı bağ katılıklarında ağ yoğunluğunun değişmesi ile beraber sonuçlar değişmektedir. Sonuçlar, alüminyum katılığının 100 katından düşük bağlayıcı katılıklarında, artan ağ yoğunluğunun oluşan hatayı arttırması şeklinde gerçekleşmektedir. Hata, bağ katılığının düşmesi ile artmaktadır.



Şekil 4.6. Çeşitli bağ katılıklarında Çizelge 3.1' de verilen ağ yoğunlukları kullanılarak z=L/2'deki hızlar

Denklemler 4.2a ve 4.2b ile verildiği üzere, problemin çözümünde ilerleyen dalganın genliği $\frac{c}{E}$' ye, dolayısıyla, $\frac{1}{\sqrt{E\rho}}$'ya lineer bağlıdır. İkinci ifade kullanılarak sonuçlar üzerinde farklı bağ katılıkları ve farklı ağ yoğunluklarında, çubuk için, yeni ve tek bir elastisite modülü tanımlamak mümkün olmaktadır. Her bir simülasyonda bulunan, çubuk için yeni elastisite modülü Şekil 4.7' de verilmiştir. Sonuçlardan görüldüğü üzere, simülasyonlarda alüminyum elastisite modülünün 100 katından az bağ katılıkları kullanılması durumunda Galerkin-tabanlı AEM Parçacıklarının Birleşimleri' nin elastisite modülü, alüminyum elastisite modülünden az olmaktadır.



Şekil 4.7. Çubuk için z=L/2'deki sonuçlar kullanılarak farklı bağ katılıkları ve ağ yoğunlukları için tanımlanan çubuğun yeni elastisite modülünün, alüminyum elastisite modülüne oranı

4.1.2 Enine Dalga Yayılımı

Enine dalga yayılımı yer değiştirme ve hız sonuçları, yalnızca bağlayıcı materyal katılığının alüminyum elastisite modülünün 100 katı kadar alındığı ve az yoğun ağ için Şekil 4.8 ve Şekil 4.9' da verilmiştir. FEM ve DEM sonuçlardaki farklılık, eleman formülasyonundan kaynaklanmayıp, Abaqus ve Pasimodo' da kullanılan farklı zaman integrasyonu metotlarından kaynaklanmaktadır. Bu farklılığa rağmen sonuçlar birbirine çok yakındır.

4.2 Frekans Domeninde Sonuçlar

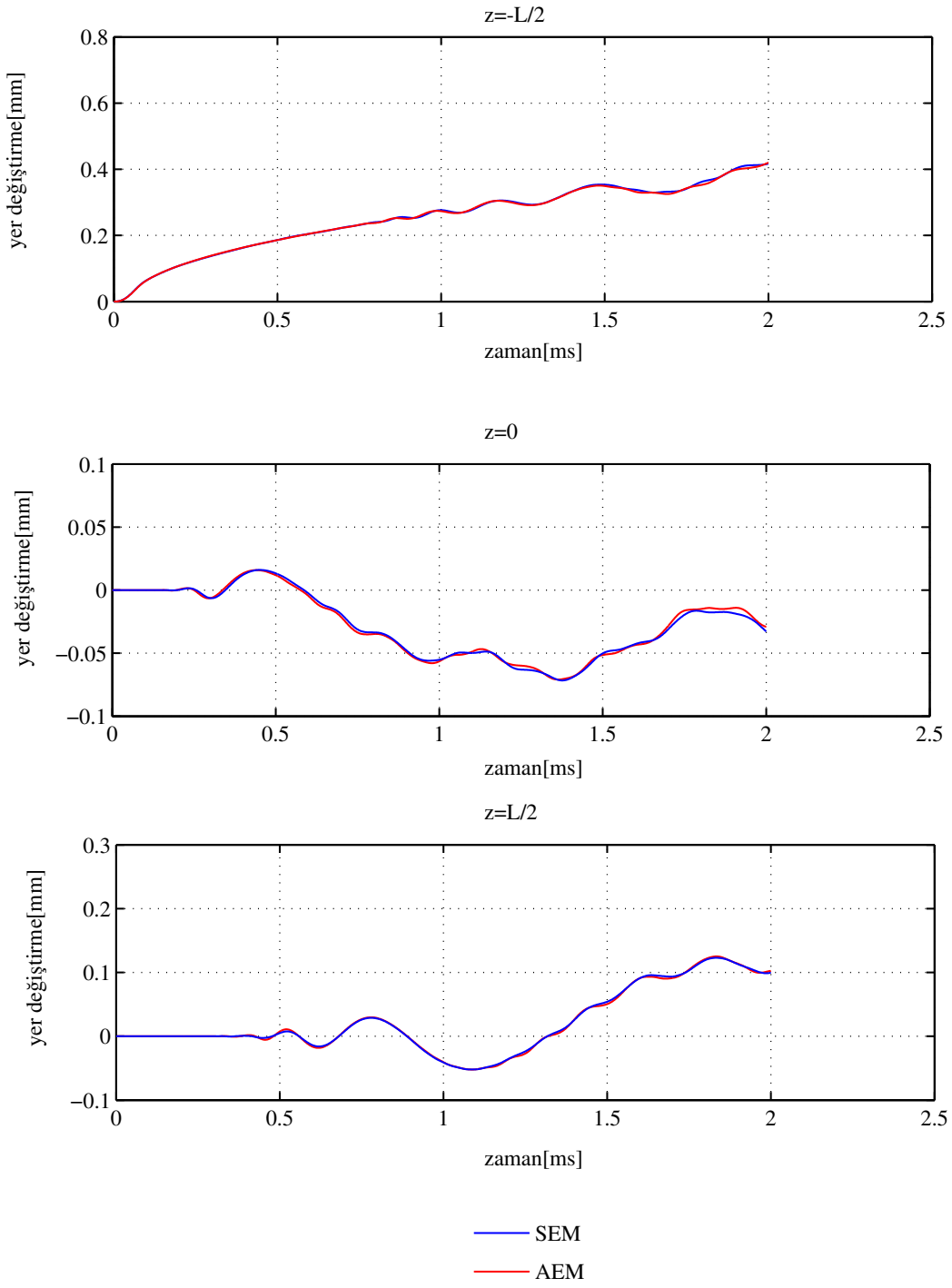
Çubuk üzerindeki noktaların hızlarının frekans-dalga numarası domenindeki gösterimi, yalnızca çok yoğun ağ için farklı bağ katılıkları kullanılarak Şekil 4.10' da verilmiştir. Verilen şekilde işaretlenmiş noktalar çubuğun titreşim modlarının yerleri, kesikli çizgiler ise, ilk iki modun birleşimi ile çizilmiş çizgilerdir.

Simülasyon için tavsiye edilen maximum frekans 66 kHz civarıdır. Bunun üzerindeki frekanslarda eleman sayısının azlığından dolayı sonuçlarda nümerik dispersiyon etkisi gözlemlenmektedir. Verilen ilk iki modun birleşimleri grafiklerinden görüleceği üzere, bağ katılığının düşmesi ile dispersiyon daha düşük frekansta başlamaktadır. Sonuç malzemenin daha düşük bir elastisite modülüne sahipmiş gibi davranmasından ileri gelmektedir.

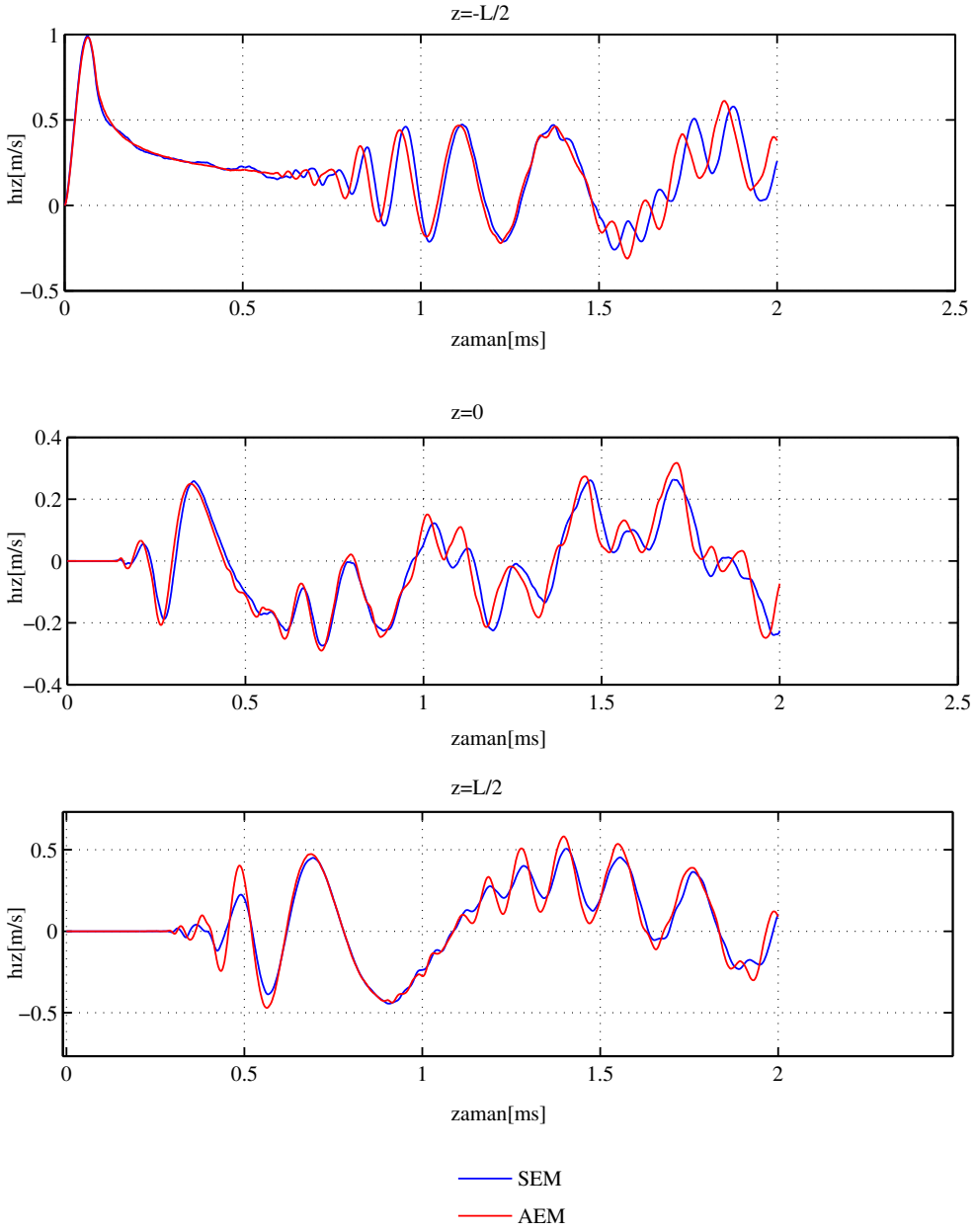
Sonuçlardan görüldüğü üzere, farklı bağ katılıkları kullanıldığında frekans domeninde yalnızca modların frekanslarının değiştiği ancak dalga numaralarının değişmediği görülmektedir. Bir çubuk için dalga numarası(mod şekilleri) geometrik değişkenlere bağlıdır. Bu sebeple Galerkin-tabanlı AEM Parçacıkları Birleşimleri metodunda bağlayıcı eleman katılığı, Sonlu Elemanlar Metodu' ndan geometrik olarak farklılığa sebep olmayıp, yalnızca malzeme farklılığına sebep olmaktadır denebilir.

Frekans domeni sonuçlarından görüldüğü üzere, bağlayıcı katılığının alüminyum katılığının 100 katından düşük değerleri için çubuk daha yumuşak malzemeye sahip gibi davranmaktadır. Bağlayıcı katılığı $10^2 E_{aluminyum}$ alındığı takdirde, sonuçlar SEM

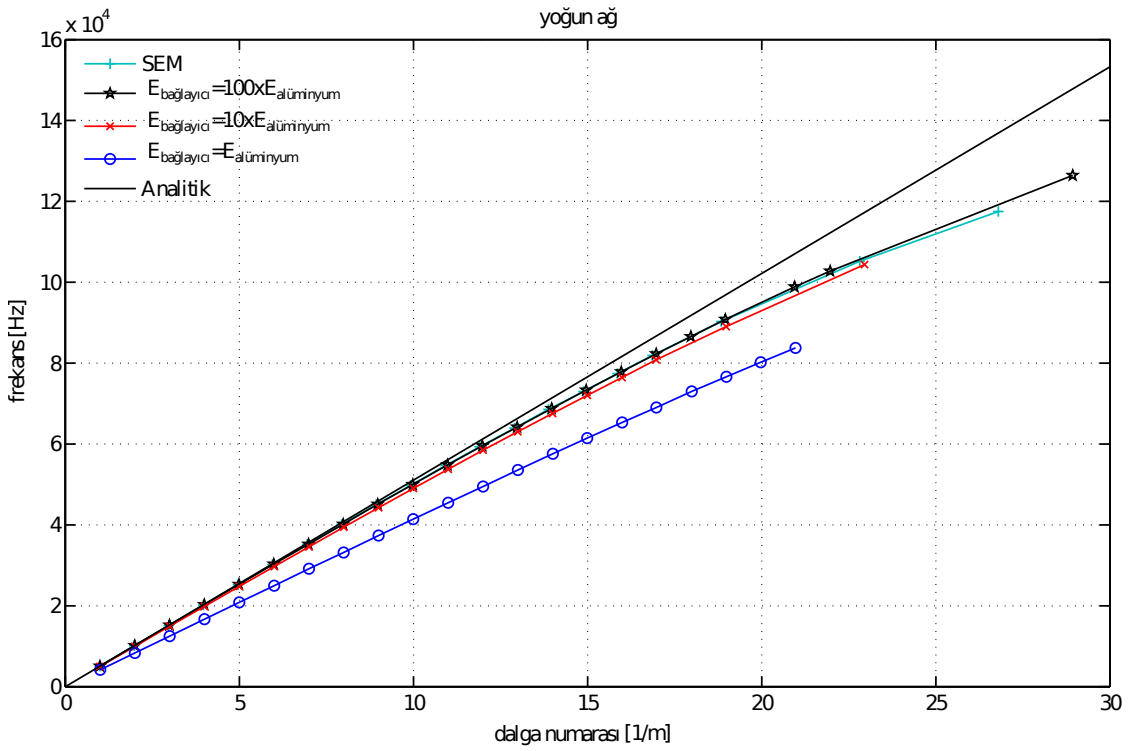
ile bulunan sonuçlarla neredeyse aynı olmaktadır. Farklılık, SEM için 11247, AEM için 12227 olan eleman sayılarının farklılığından ileri gelmektedir.



Şekil 4.8. Enine dalga yayılımı probleminde z=0, z=L/2 ve z=-L/2 için yer değiştirmeler



Şekil 4.9. Enine dalga yayılımı probleminde z=0, z=L/2 ve z=-L/2 için hızlar



Şekil 4.10. Yoğun ağ kullanılarak farklı bağ katılıkları için dalga numarası-frekans grafiği

5. TARTIŞMA VE SONUÇ

Bu çalışma kapsamında, sonlu elemanlara benzer ancak temel olarak Ayrık Elemanlar Metodu tabanlı bir yaklaşım olan, geometrik olarak bağımsız Galerkin-tabanlı tetrahedral sonlu elemanların birleşimlerinde(Galerkin-tabanlı AEM Parçacıkları Birleşimleri) dalga yayılımı karakteristikleri incelenmiştir. Bu amaçla daha önceden incelenmiş olan bir çarpma problemi tabanlı simülasyon kurgusu kullanılmıştır. Simülasyon kurgusu ile hem boyuna hem de enine dalga yayılımı problemleri zaman domeninde incelenmiştir. Boyuna dalga yayılımı problemi aynı zamanda frekans domeninde de incelenmiştir.

Kullanılan metot ile beraber sonlu elemanlar metoduna ek olan elemanlar arası bağlayıcı katılığı çalışma kapsamında odaklanılan ana noktadır. Bağlayıcı katılığının ve ağ yoğunluğunun etkilerinin incelenmesi amacıyla farklı bağ katılıkları ve farklı ağ yoğunlukları kullanılarak dalga yayılımı simülasyonları gerçekleştirilmiştir.

Sonuçlardan görülmüştür ki bağ katılığı ve ağ yoğunluğu dalga yayılımı probleminin sonuçlarına belli bir katılık değerinin altında oldukça fazla etki etmektedir. Bu katılık değeri iç malzeme olarak seçilen alüminyumun katılığının 100 katından az değerlerdir. Bu katılığın üstündeki değerler için dalga yayılımı sonuçları ağ yoğunluğu ne olursa olsun sonlu elemanlar metodu ile elde edilen sonuçlar ile aynı olmaktadır.

Bağlayıcı katılının $100xE_{alüminyum}$ değerinin altında alındığı halde model tekrar tek bir model olarak davranmaktadır ancak hem boyuna hem de enine dalga yayılımı probleminde çözüm dalga hızı düşmekte ve dalga genliği artmaktadır. Bu davranış genel bir malzeme yumuşaması davranışı olarak yorumlanmıştır. Böylelikle düşük bağ katılığı değerleri için yalnızca bir tek malzeme katılığı tanımlanması yeterli olmaktadır. Tanımlanabilecek malzeme katılığı, azalan bağ katılığı ile beraber yaklaşık üstel bir düşüş göstermektedir. Bunun dışında herhangi farklı bir davranış gözlemlenmemiştir.

Farklı ağ yoğunluklarının sonuç üzerindeki etkisi ise artan ağ yoğunluğunun bağ katılığının alüminyum iç katılığının 100 katından az değerlerindeki yumuşama etkisini arttırıcı yönde gözlemlenmiştir. Bağ katılığının 100 kat ve üstü değerleri için ağ yoğunluğu sonuçlara etki etmemektedir.

Zaman domeninde incelemelerin yanında, frekans domeni incelemeleri ile beraber modelin yumuşama davranışı farklı bağ katılıkları için sabit dalga numaraları ve değişen dalga frekansları ile yeniden bulunmuştur. Ayrıca, uygun ağ yoğunluğu seçildiği takdirde, metodun frekansa bağımlılığı olmadığı gözlemlenmiştir.

Çalışma kapsamında lineer elastik bağlayıcı modeli kullanılmıştır. Bağlayıcı modeli kırılma mekaniği uygulamaları için modifiye edilerek, nonlineer ve plastik davranışa sahip olarak da modellenebilir.

KAYNAKLAR

Carrier, G. ve Pearson, C. 1988. Partial differential equations: theory and technique, Academic Press. ISBN 9780121604516, url:http://books.google.de/books?id=eiXvAAAAMAAJ.

Cundall, P.A. ve Strack, O.D.L. 1979. A discrete numerical model for granular assemblies, *Géotechnique*, ***29***(1), 47–65.

de Silva, C. 2010. Vibration: Fundamentals and Practice, Second Edition, Taylor & Francis. ISBN 9780849318085, url:http://books.google.com.tr/books?id=FTw9Ln4RX2oC.

Drozdz, M.B. 2008. Efficient finite element modelling of ultrasound waves in elastic media. *Doktora Tezi*, Imperial College of Science Technology and Medicine, London.

Eberhard, P. ve Gaugele, T. 2013. Simulation of cutting processes using mesh-free Lagrangian particle methods, *Computational Mechanics*, ***51***, 261–278. ISSN 0178-7675. doi:10.1007/s00466-012-0720-z, url:http://dx.doi.org/10.1007/s00466-012-0720-z.

F., M., L.J., J. ve J., Q. 1999. Modeling elastic wave propagation in waveguides with the finite element method, *NDT and E International*, ***32***(4), 225–234. doi:doi:10.1016/S0963-8695(98)00045-0, url:http://www.ingentaconnect.com/content/els/09638695/1999/00000032/00000004/art00045.

Fleissner, F. 2010. Parallel Object Oriented Simulation With Lagrangian Particle Methods, cilt 16, Shaker Verlag, Aachen.

Fleissner, F. 2013. Pasimodo, www.itm.uni-stuttgart.de/research/pasimodo/pasimodo.php.

Fleissner, F., **D'Alessandro**, **Schiehlen, W. ve Eberhard, P. 2009**. Sloshing cargo in silo vehicles, *Journal of Mechanical Science and Technology*, ***23***(4), 968–973.

Fung, Y. 1965. Foundations of solid mechanics, Prentice-Hall international series in dynamics, Prentice-Hall, url:http://books.google.de/books?id=P9AQAQAAIAAJ.

Gaugele, T., **Fleissner, F. ve Eberhard, P. 2008**. Simulation of material tests using meshfree Lagrangian particle methods, *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics*, ***222***(4), 327–338. doi:10.1243/14644193JMBD167.

Graff, K. 1975. Wave Motion in Elastic Solids, Dover Books on Engineering Series, Dover Publ. ISBN 9780486667454, url:http://books.google.de/books?id=5cZFRwLuhdQC.

Hauser, W. 1965. Introduction to the principles of mechanics, Addison-Wesley series in physics, Addison-Wesley Pub. Co., url:http://books.google.com.tr/books?id=d_VQAAAAMAAJ.

Hu, B., **Schiehlen, W. ve Eberhard, P. 2003**. Comparison of Analytical and Experimental Results for Longitudinal Impacts on Elastic Rods, *Journal of Vibration and Control*, ***9***(1-2), 157–174. doi:10.1177/107754603030745, url:http://jvc.sagepub.com/content/9/1-2/157.abstract.

Hughes, T.J. 2000. The Finite Element Method, Dover.

Munjiza, A. 2004. The Combined Finite-Discrete Element Method, John Wiley & Sons. ISBN 9780470841990, url:http://books.google.de/books?id=YVBR_h9WuMYC.

Mustoe, G.G.W. 1992. A Generalized Formulation of the Discrete Element Method, *Engineering Computations*, ***9***(2), 181–190. doi:10.1108/eb023857.

Press, W.H., **A.Teukolsky, S.**, **Vetterling, W.T. ve Flannery, B.P. 2007**. Numerical Recipies, Cambridge University Press.

Rose, J. 2004. Ultrasonic Waves in Solid Media, Cambridge University Press. ISBN 9780521548892, url:http://books.google.de/books?id=DEtHDJJ-RS4C.

Ryden, N., **Park, C.**, **Ulriksen, P. ve Miller, R.D. 2003**. Lamb Wave Analysis for Non-Destructive Testing of Concrete Plate Structures, *Symposium on the Application of Geophysics to Engineering and Environmental Problems*, ***16***(1), 782–793. doi:10.4133/1.2923224, url:`http://link.aip.org/link/?SAG/16/782/1`.

Sadd, M.H. 2009. Wave Motion and Vibration in Continuous Media, Mechanical Engineering & Applied Mechanics Department University of Rhode Island.

Stroustrup, B. 1997. The C++ programming language, Professionelle Programmierung, Addison-Wesley. ISBN 9780201889543, url:`http://books.google.de/books?id=LqRVAAAAMAAJ`.

Wriggers, P.H. 2006. Computational Contact Mechanics, Springer.

Šmilauer, V., **Catalano, E.**, **Chareyre, B.**, **Dorofeenko, S.**, **Duriez, J.**, **Gladky, A.**, **Kozicki, J.**, **Modenese, C.**, **Scholtès, L.**, **Sibille, L.**, **Stránský, J. ve Thoeni, K. 2010**. Yade Documentation.

EKLER

EK1 Sonsuz İnce Çubukta Boyuna Dalga Yayılımı için Analitik Çözüm
EK2 Pasimodo için Geliştirilen Sınır Şartları Plugini Kodu
EK3 Pasimodo XML Modeli

EK 1 Sonsuz İnce Çubukta Boyuna Dalga Yayılımı için Analitik Çözüm

Bir ucundan basınç uygulamak suretiyle tahrik edilen sonsuz bir çubuğun boyuna dalga yayılımı problemi

$$\begin{array}{ll} u_{tt} = c^2 u_{xx} & 0 < x < \infty \quad t > 0 \\ u(x,0) = 0 & 0 < x < \infty \\ u_t(x,0) = 0 & 0 < x < \infty \\ u_x(0,t) = -\frac{\sigma(t)}{E} & t > 0 \\ |u(\infty,t)| = 0 & t > 0 \end{array} \tag{A.1}$$

şeklinde ifade edilir. Burada $\sigma(t)$ uygulanan basınç profilidir. Problem, zamanda Laplace dönüşümü ile adi diferansiyel denkleme dönüştürülerek

$$\begin{array}{ll} c^2 U_{xx} - s^2 U = 0 & 0 < x < \infty \quad t > 0 \\ U_x(0,s) = -\hat{\sigma}(s) & \\ |U(\infty,s)| = 0 & \end{array} \tag{A.2}$$

bulunur. Bu problem için s domeninde yer değiştirmeler için genel çözüm, sonsuzdaki sınır şartı kullanılarak

$$U(x,s) = C_1 e^{-\frac{s}{c}x} \tag{A.3}$$

eşitliği olarak bulunur. Basınç sınır şartı uygulanmak koşuluyla

$$U(x,s) = \frac{c}{E}\frac{\hat{\sigma}(s)}{s} e^{-\frac{s}{c}x} \tag{A.4}$$

bulunur. $\mathscr{L}[f(t-a)Heaviside(t-a)] = \hat{f}(s)e^{-as}$ eşitliği kullanılarak yer değiştirmeler

$$u(x,t) = \frac{c}{E}\mu(t-\frac{x}{c})Heaviside(t-\frac{x}{c}) \tag{A.5}$$

eşitliğini sağlamak zorundadır. Burada

$$\mu(t) = \int \sigma(t)\, dt$$

ve hızlar da

$$\dot{u}(x,t) = \frac{c}{E}\sigma(t-\frac{x}{c})Heaviside(t-\frac{x}{c}) \tag{A.6}$$

olarak bulunur.

EK 2 Pasimodo için Geliştirilen Sınır Şartları Plugini Kodu

Kod Parçası 1. PS_TetrahedronBoundry.hpp

```
#ifndef PS_TETRAHEDRON_BOUNDRY_HPP
#define PS_TETRAHEDRON_BOUNDRY_HPP



#include "PS_Vec.hpp"
#include "PS_SimulatorComponentInterface.hpp"
#include "PS_Tetrahedron_.hpp"
#include "PS_Ptr.hpp"

PS_SIM_COMP(class, PS_TetrahedronBoundry,
         "Tetrahedron_Boundry", PS_SimulatorComponent_)

public:
  friend class PS_TetrahedronBoundryShapeInteraction;
      /*This function returns loads for each
      node of the tetrahedron*/
      virtual void apply(PS_Tetrahedron_& tet){}
      virtual PS_TetrahedronBoundry* createACopy(void)
      {return new PS_TetrahedronBoundry(*this); }
      virtual bool Init(Eigen::Vector4i& enclosedNodes)
      {return true;}
protected:

};
#endif
```

Kod Parçası 2. PS_TetrahedronBoundry.cpp

```
#define PS_ETI_COMPONENT PS_TetrahedronBoundry
#include "PS_TetrahedronBoundry.hpp"
#include "PS_SimulatorComponentInterface_Impl.hpp"

PS_TetrahedronBoundry::PS_TetrahedronBoundry(void)
{

}
```

Kod Parçası 3. PS_TetrahedronForceBoundry.hpp

```
#ifndef PS_TETRAHEDRON_FORCE_BOUNDRY_HPP
#define PS_TETRAHEDRON_FORCE_BOUNDRY_HPP

#include "PS_SimulatorComponentInterface.hpp"
#include "PS_TetrahedronBoundry.hpp"
```

```
#include "PS_Vec.hpp"


PS_SIM_COMP(class, PS_TetrahedronForceBoundry,
         "Tetrahedron_Force_Boundry",
         PS_TetrahedronBoundry)

public:

protected:

};
#endif
```

Kod Parçası 4. PS_TetrahedronForceBoundry.cpp

```
#define PS_ETI_COMPONENT PS_TetrahedronForceBoundry
#include "PS_TetrahedronForceBoundry.hpp"
#include "PS_SimulatorComponentInterface_Impl.hpp"

PS_TetrahedronForceBoundry::
      PS_TetrahedronForceBoundry(void)
{

}
```

Kod Parçası 5. PS_TractionForce.hpp

```
#ifndef PS_TRACTION_FORCE_HPP
#define PS_TRACTION_FORCE_HPP

#include "PS_TetrahedronBoundry.hpp"

PS_SIM_COMP(class, PS_TractionForce,
         "Traction_Force_Boundry", PS_TetrahedronBoundry)

public:
      virtual void apply(PS_Tetrahedron_&);
      virtual PS_TetrahedronBoundry* createACopy(void);
      PS_Vec getForce(void)
      {
            return *ptrToFrc;
      }
      virtual bool Init(Eigen::Vector4i& enclosedNodes);
      void CalculateArea(PS_Tetrahedron_&);
protected:
      PS_ATTRIBUTE(PS_Vec, force_, "force",
      "Force to apply tetrahedron's surface");
```

```
        /*Please use pointers to members that are given
        in xml file.
        This is because this class will be copied in
        createACopy class.
        Members wont be avaliable for copies.*/
        PS_Vec* ptrToFrc;
        /*Variables*/
        Eigen::Vector4i tetrahedron_nodes_index;
        /*Nodes that are used to construct
        tetrahedron surfaces*/
        PS_Scalar area[4];//Area of tetrahedron's faces
        bool is_calculated;

};
#endif
```

Kod Parçası 6. PS_TractionForce.cpp

```
#define PS_ETI_COMPONENT PS_TractionForce
#include "PS_TractionForce.hpp"
#include "PS_SimulatorComponentInterface_Impl.hpp"
#include <iostream>
#include <fstream>
using namespace std;
using namespace Eigen;
#define _node(i) tet.getNodeGlobal(i)
ofstream myfile("fout.txt");

PS_TractionForce::PS_TractionForce()
{
        tetrahedron_nodes_index=Eigen::Vector4i::Zero();
}

/*
        NOT TESTED

*/

PS_TetrahedronBoundry* PS_TractionForce::createACopy(void)
{

        ptrToFrc=&force_;
        //Return pointer to newly created instance.
        return new PS_TractionForce(*this);
}

/*
```

```
      TESTED : OK

*/
void PS_TractionForce::apply(PS_Tetrahedron_& tet)
{

      PS_Vec f[4]; //Force to apply to tetrahedron's nodes
      PS_Vec _force=getForce();
      /*Indexing variables to decide which faces to
      apply traction force*/
      Vector4i _identity,_indexed,_indexes=
                           Vector4i::Zero();
      _identity<<1,1,1,1;
      _indexed<<0,1,2,3;

      int j=0;
      //Arrange indexes
      for(int i=0;i<4;i++)
      {
        if(tetrahedron_nodes_index(i)!=0){
        _indexes(j)= _indexed(i);j++;
        }else{
       _indexes(3)= _indexed(i);
        }
      }

      //CalculateArea(tet);
      if(tetrahedron_nodes_index == _identity){
      /*This boundry includes all the nodes
      of tetrahedron.Therefore,
      this tractionforce must be applied to
      all avaliable faces of the tetrahedron.*/
      area[0]=0.5*(_node(0)-_node(1)).
               cross(_node(0)-_node(2)).norm();
      area[1]=0.5*(_node(0)-_node(1)).
               cross(_node(0)-_node(3)).norm();
      area[2]=0.5*(_node(0)-_node(2)).
               cross(_node(0)-_node(3)).norm();
      area[3]=0.5*(_node(1)-_node(2)).
               cross(_node(1)-_node(3)).norm();
      //Distribute surface force equally to effected nodes
      f[0]=(((PS_Scalar)1)/((PS_Scalar)3))*
        (force_*area[0]+force_*area[1]+force_*area[2]);
      f[1]=(((PS_Scalar)1)/((PS_Scalar)3))*
        (force_*area[0]+force_*area[1]+force_*area[3]);
      f[2]=(((PS_Scalar)1)/((PS_Scalar)3))*
```

```
        (force_*area[0]+force_*area[2]+force_*area[3]);
      f[3]=(((PS_Scalar)1)/((PS_Scalar)3))*
        (force_*area[1]+force_*area[2]+force_*area[3]);
      }
      else{
      /*To only one face that is formed by
      tetrahedron_nodes_index*/

      area[0]=0.5*(_node(_indexes(0))-_node(_indexes(1))).
       cross(_node(_indexes(0))-_node(_indexes(2))).
         norm();
      //Distribute surface force equally to effected nodes
      f[_indexes(0)]=(((PS_Scalar)1)/
         ((PS_Scalar)3))*(_force*area[0]);
      f[_indexes(1)]=(((PS_Scalar)1)/
         ((PS_Scalar)3))*(_force*area[0]);
      f[_indexes(2)]=(((PS_Scalar)1)/
         ((PS_Scalar)3))*(_force*area[0]);
      //Uneffected node
      f[_indexes(3)]=Eigen::Matrix<PS_Scalar,3,1>::Zero();
      }
      for(int i=0;i<4;i++){
      /*Add forces to conservative forces of the
      tetrahedron*/
      tet.addNodalConservativeForce(i,f[i]);
      }
      /*if(area[0]>35.3554){


      plog << "Area is equal to "<<area[0] << pendl;
      plog << "Force[0] is equal to "<<f[0] << pendl;
      plog << "Force[1] is equal to "<<f[1] << pendl;
      plog << "Force[2] is equal to "<<f[2] << pendl;
      plog << "Force[3] is equal to "<<f[3] << pendl;

      #include <unistd.h>
      sleep(2);
}*/



/*
      f[_indexes(0)]=force_;
      f[_indexes(1)]=force_;
      f[_indexes(2)]=force_;
      //Uneffected node
```

```
      f[_indexes(3)]=Eigen::Matrix<PS_Scalar,3,1>::Zero();
      for(int i=0;i<4;i++){
      /*Add forces to conservative forces
        of the tetrahedron
      */
      tet.addNodalConservativeForce(i,f[i]);
      }


/*    plog << "Area is equal to "<<area[0] << pendl;
      f[0]+=getForce();
      f[1]+=getForce();
      f[2]+=getForce();
      f[3]=PS_Vec::Zero();
for(int i=0;i<4;i++){
      tet.addNodalConservativeForce(i,f[i]);
      }

//    myfile<<setprecision(14)<<getForce()<<"\n";
*/

}

/*
      This function is used to check if there is
      meaningfull geometric description of boundry.
      NOT TESTED

*/

void PS_TractionForce::CalculateArea(PS_Tetrahedron_& tet)
{
      if(is_calculated==true){
      return;
      }else{
      /*Indexing variables to decide which faces
      to apply traction force*/
      Vector4i _identity,_indexed,_indexes=
         Vector4i::Zero();
      _identity<<1,1,1,1;
      _indexed<<0,1,2,3;

      int j=0;
      //Arrange indexes
      for(int i=0;i<4;i++)
      {
```

```
        if(tetrahedron_nodes_index(i)!=0){
        _indexes(j)= _indexed(i);j++;
        }else{
       _indexes(3)= _indexed(i);
        }
      }
      if(tetrahedron_nodes_index == _identity){
      /*This boundry includes all the nodes
      of tetrahedron.Therefore,
      this tractionforce must be applied
      to all avaliable faces of the tetrahedron.*/
      area[0]=0.5*(_node(0)-_node(1)).
        cross(_node(0)-_node(2)).norm();
      area[1]=0.5*(_node(0)-_node(1)).
        cross(_node(0)-_node(3)).norm();
      area[2]=0.5*(_node(0)-_node(2)).
        cross(_node(0)-_node(3)).norm();
      area[3]=0.5*(_node(1)-_node(2)).
        cross(_node(1)-_node(3)).norm();
      }
      else{
      /*To only one face that is
      formed by tetrahedron_nodes_index*/

      area[0]=0.5*(_node(_indexes(0))-_node(_indexes(1))).
        cross(_node(_indexes(0))-_node(_indexes(2)))
          .norm();
      area[1]=0;
      area[2]=0;
      area[3]=0;

      }
            is_calculated=true;
}
}


bool PS_TractionForce::Init(
  Eigen::Vector4i& nodes_incontact)
{
      tetrahedron_nodes_index = nodes_incontact;
      plog << pdebug << "Function is Called"<< pendl;
      int total=tetrahedron_nodes_index(0)+
        tetrahedron_nodes_index(1)+
        tetrahedron_nodes_index(2)+
        tetrahedron_nodes_index(3);
```

```
        is_calculated=false;
        if(total>=3)
        {
         /*We need at least three nodes for
         the traction force*/
          return true;
        }
        else{
        return false;
        }
}
```

Kod Parçası 7. PS_BoundryShape.hpp

```
#ifndef PS_BOUNDRY_SHAPE_HPP
#define PS_BOUNDRY_SHAPE_HPP


#include "PS_3DOFSphere.hpp"
#include "PS_TetrahedronBoundry.hpp"
class PS_Tetrahedron_;


PS_PARTICLE(class, PS_BoundryShape,
          "Boundry_Shape", PS_3DOFSphere)

public:
      friend class PS_TetrahedronBoundryShapeInteraction;
      /*This function returns enclosed nodes
      of a tetrahedron within imposer outter radius*/
      virtual Eigen::Vector4i checkEnclosedNodes(
        PS_Tetrahedron_& t);
      bool check4Deletion(void);
      PS_TetrahedronBoundry* getNewBoundryInstance(void);
protected:
      PS_NESTED_COMPONENT(PS_Ptr<PS_TetrahedronBoundry>,
      _boundry,"boundry_",
      "This nested member holds pointer to reference"
      "boundry instance.");
      PS_ATTRIBUTE(PS_Vec,XwidthYheightZdepth,
      "XwidthYheightZdepth",
      "This nested member holds width,"
      "height and depth of cube whose center"
      "coincides with the base class"
      "tetrahedronboundry.");
};
#endif
```

Kod Parçası 8. PS_BoundryShape.cpp

```
#define PS_ETI_COMPONENT PS_BoundryShape
#include "PS_Tetrahedron_.hpp"
#include "PS_TetrahedronBoundry.hpp"
#include "PS_BoundryShape.hpp"
#include "PS_Vec.hpp"
#include "PS_InteractionData_.hpp"
#include "PS_ExtendedState_.hpp"
#include "PS_Statistics.hpp"

#include "PS_Particle_Impl.hpp"

PS_BoundryShape::PS_BoundryShape(void)
{

}

bool PS_BoundryShape::check4Deletion(void)
{
	if (pstat.simulationStep_ > 0)
	{
		return true;
	}
	plog << pdebug << "simulationstep is "<<
	pstat.simulationStep_ << pendl;
	return false;
}

PS_TetrahedronBoundry* PS_BoundryShape::
  getNewBoundryInstance(void)
{


	return _boundry->createACopy();
}

#define _node(tetra,i) tetra.getNodeGlobal(i)

/*
Initially behave like a cube
*/

Eigen::Vector4i PS_BoundryShape::
  checkEnclosedNodes(PS_Tetrahedron_& t)
{
	PS_Vec nodes[4];
```

```
        PS_Vec pos;
        PS_Scalar radius;
        Eigen::Vector4i _enclosed_indices;
        t.getNodesGlobal(nodes);

        pos=this->getTranslationalState().getCenter();
        radius=getRadius();
        //Now check the nodes inside this cube

        for(int i=0;i<4;i++)
        {
          PS_Vec diff=(nodes[i]-pos);

          if((fabs(diff(0))<(0.5*XwidthYheightZdepth(0)))
             &(fabs(diff(1))<(0.5*XwidthYheightZdepth(1)))
             &(fabs(diff(2))<(0.5*XwidthYheightZdepth(2)))
            )
           _enclosed_indices(i)=1;
           else
           enclosed_indices(i)=0;
        }
        return _enclosed_indices;
}
```

Kod Parçası 9. PS_CubeBoundryShape.hpp

```
#ifndef PS_CUBE_BOUNDRY_SHAPE_HPP
#define PS_CUBE_BOUNDRY_SHAPE_HPP


#include "PS_BoundryShape.hpp"



PS_PARTICLE(class, PS_CubeBoundryShape,
   "Cube_Boundry_Shape", PS_BoundryShape)

public:
      /*This function returns enclosed nodes of a
      tetrahedron within imposer outter radius and imposer
      internal cube that is defined*/
      virtual Eigen::Vector4i checkEnclosedNodes
        (PS_Tetrahedron_& t);

protected:
      PS_NESTED_COMPONENT(PS_Vec,XwidthYheightZdepth,
      "widthheightdepth",
```

```
      "This nested member holds width, "
      "height and depth of cube whose center coincides "
      "with the base class tetrahedronboundry.");
};
#endif
```

Kod Parçası 10. PS_CubeBoundryShape.cpp

```
#define PS_ETI_COMPONENT PS_CubeBoundryShape
#include "PS_CubeBoundryShape.hpp"

PS_CubeBoundryShape::PS_CubeBoundryShape(void)
{

}


#define _node(tetra,i) tetra.getNodeGlobal(i)

/*
Initially behave like a sphere
*/

Eigen::Vector4i PS_CubeBoundryShape::
  checkEnclosedNodes(PS_Tetrahedron_& t)
{
      PS_Vec nodes[4];
      PS_Vec pos;
      PS_Scalar radius;
      Eigen::Vector4i _enclosed_indices;
      t.getNodesGlobal(nodes);
      pos=this->getTranslationalState().getCenter();
      radius=getRadius();
      //Now check the nodes inside this cube
      for(int i=0;i<4;i++)
      {
            PS_Vec diff=(nodes[i]-pos);
       if((fabs(diff(0))<(0.5*XwidthYheightZdepth(0)))
          &(fabs(diff(1))<(0.5*XwidthYheightZdepth(1)))
          &(fabs(diff(2))<(0.5*XwidthYheightZdepth(2)))
         )
         _enclosed_indices(i)=1;
        else
         _enclosed_indices(i)=0;
      }
      return _enclosed_indices;
}
```

Kod Parçası 11. PS_TetrahedronBoundryShapeInteraction.hpp

```
#ifndef PS_TETRAHEDRON_BOUNDRY_SHAPE_INTERACTION
#define PS_TETRAHEDRON_BOUNDRY_SHAPE_INTERACTION

#include "PS_Interaction_.hpp"
#include "PS_Tetrahedron_.hpp"
#include "PS_BoundryShape.hpp"
#include "PS_FlexibleTetrahedronInteriorForces.hpp"
#include "PS_FlexibleTetrahedron.hpp"


PS_INTERACTION(
          class,
          PS_TetrahedronBoundryShapeInteraction,
          "Tetrahedron_Boundry_Shape_Interaction",
          PS_Interaction_)
public:
      typedef PS_Tetrahedron_ ParticleType1;
      typedef PS_BoundryShape ParticleType2;

      virtual bool operator()(
      PS_Particle_ &p1,
      PS_Particle_ &p2);

protected:
};
#endif
```

Kod Parçası 12. PS_TetrahedronBoundryShapeInteraction.cpp

```
#define PS_ETI_COMPONENT \
      PS_TetrahedronBoundryShapeInteraction
#include "PS_TetrahedronBoundryShapeInteraction.hpp"

#include "PS_DirectParticleOperation_.hpp"
#include "PS_ParticleDisposer.hpp"
#include "PS_Particle_.hpp"
#include "PS_ParticleProperties.hpp"
#include "PS_Statistics.hpp"
#include "PS_Interaction_Impl.hpp"

PS_TetrahedronBoundryShapeInteraction
      ::PS_TetrahedronBoundryShapeInteraction()
{

}
```

```
bool
PS_TetrahedronBoundryShapeInteraction
      ::operator()(PS_Particle_ &p1, PS_Particle_ &p2)
{

      PS_FlexibleTetrahedron &tet =
        static_cast<PS_FlexibleTetrahedron&>(p1);
      // This is interesting because we are not guaranteed
      // that we always have FlexibleTetrahedron;
      PS_BoundryShape & shape =
        static_cast<PS_BoundryShape&>(p2);

      PS_Ptr<PS_ParticleProperties> tet_prop_tmp =
         tet.getProperties()->clone();

      PS_Ptr<PS_DirectParticleOperation_> tet_ops =
         tet_prop_tmp->getBeforeIntegrationHook();
      if (!tet_ops)
        PS_Exception<>(context_) <<
           "Particle Operations do not exist";

      PS_Ptr<PS_FlexibleTetrahedronInteriorForces>
      tet_int_forces =
         safeCast<PS_FlexibleTetrahedronInteriorForces>
         (tet_ops);
      if (!tet_int_forces)
        PS_Exception<>(context_) <<
             "interior forces are of wrong type";

      /************ Do not add the same boundry
      twice.**********/
      // Also it must be noted that this version does not
      // support moving boundry.Boundry can only be
      // applied for the first step
      // and it exist through the simulation with its
      // initial form.
      // Therefore it is currently not supported to
      // change boundry behaviour while in simulation.

      if (shape.check4Deletion()){
        //remove particle therefore it does not
        // exist in the following steps.
        this->getParticleDisposer().
        removeParticle(shape.getUid());
        //plog << pdebug << "simulationstep is"
                        "not 0 it is equal to "<<
```

```
                        pstat.simulationStep_ << pendl;
            return true;
        }
        else{
          //Initialize boundry by checking enclosed nodes
          Eigen::Vector4i v=
             shape.checkEnclosedNodes(tet);
          PS_TetrahedronBoundry* bndry_to_add=
             shape.getNewBoundryInstance();

          if(bndry_to_add->Init(v))
          {// Init with enclosed node indices
          plog << pdebug << "This tetrahedron is"
            "suitable to apply boundry" <<v<< pendl;
          //Add boundry to tetrahedron
          tet_int_forces->addBoundry(bndry_to_add);
          tet.setProperties(tet_prop_tmp);
        }



        }


        return true;
}
```

EK3 Pasimodo Girdi Dosyası

Kod Parçası 13. Pasimodo Girdi Dosyası

```
<Particle_Simulation>


      <Plugin name="pplug_tetrahedron_mesh_generation"/>
      <Plugin name="pplug_tetrahedron_assembly"/>
      <Plugin name="pplug_h5part"/>

      <Arguments enable_output="true" post_Interval="1e-6"
      post_Enabled="true" input_file_name="AluminiumRod.pre"
      IsParallel="true" NProcessors="4" sim_start="0"
      sim_end="0.004"/>

      <Data_Output data_dir="post">

<!--
<Post_Processing data_dir="out"/>

<Post_Processing_Output data_dir="out"
interval="'post_Interval'" enabled="'post_Enabled'"/>
<H5Part_Output
       priority           = "-1"
       relative_trigger_time = "false"
       next_t_trigger     = "-inf"
       after_every_timestep = "false"
       periodic           = "true"
       interval           = "'paramOut_Interval'"
       data_dir           = "h5part"
       name               = "paraviewout"
       enabled            = "false"
       mode_of_operation  = "2"
       n_digits           = "5"
       particle_types     = "Flexible_Tetrahedron"
    >
<Nested name = "member_description">
<Flexible_Tetrahedron>
<Particle_Properties>
<Nested name = "before_integration">

<Flexible_Tetrahedron_Interior_Forces>

<Nested name = "reference_coordinate_computation">
<Tangential_Reference_Coordinates/>
</Nested>

<Nested name = "interior_force_computation">
<Linear_Finite_Element_Interior_Forces/>
</Nested>
```

```
</Flexible_Tetrahedron_Interior_Forces>
</Nested>

</Particle_Properties>
</Flexible_Tetrahedron></Nested>
    </H5Part_Output>
-->
<Function_Output functions="ForceZ"
enabled="1" interval="1e-6"/>

<All_Particles_Tabular_Parameter_Output priority="-1"
relative_trigger_time="false" next_t_trigger="-inf"
after_every_timestep="false" periodic="true"
interval="1e-6" data_dir="tabular_all" name=""
enabled="true" mode_of_operation="1" n_digits="4"
output_parameters="center;velocity;"
particle_types="Flexible_Tetrahedron">
<Nested name="sample">
<Flexible_Tetrahedron>
<Particle_Properties>
<Nested name="before_integration">
<Flexible_Tetrahedron_Interior_Forces>
<Nested name="reference_coordinate_computation">
<Tangential_Reference_Coordinates/>
</Nested>
<Nested name="interior_force_computation">
<Linear_Finite_Element_Interior_Forces/>
</Nested>
</Flexible_Tetrahedron_Interior_Forces>
</Nested>
</Particle_Properties>
</Flexible_Tetrahedron>
</Nested>
</All_Particles_Tabular_Parameter_Output>
</Data_Output>

<!-- Definition of parameters
-->

<Constants rho="2.789*(1e-9)" E="72800"
tscaled="10" tend="1.755*(1e-4)*(1/tscaled)"
bond_stiffness="72800" />


<Particles gravity="(0, 0, 0)">

<Functions>
```

```
<Analytic_Function name="ForceX" expression="0"/>
<Analytic_Function name="ForceY" expression="0"/>
<Analytic_Function name="ForceZ"
expression="(tend &gt;= t)*
(((-8.78455683548012e29*((tscaled*t)^7)+
9.06751845980641e26*((tscaled*t)^6)-
4.12927170518450e23*((tscaled*t)^5)+
1.00511352088627e20*((tscaled*t)^4)-
1.27738828371930e16*((tscaled*t)^3)+
688924288883*((tscaled*t)^2)-
4386215*((tscaled*t)^1)+
6.42282344770129)/(3.14*100)))*tscaled "/>

</Functions>
<!-- It's box default -->
<Boundry_Shape center="(0,0,505)"
radius="20" XwidthYheightZdepth="(20,20,10)" >
<Nested name="boundry_">
<!--Pressure Boundry-->
<Traction_Force_Boundry force="(0,0,#ForceZ)"/>

</Nested>

</Boundry_Shape>

<Tetrahedron_Mesh_Generator
attach_interaction_data="false"
geometry_filename="Cylinder.mesh"
geometry_filetype="4" max_volume="8e2"
random_color="true">
<Nested name="sample_tetrahedron">

<Flexible_Tetrahedron>

<Particle_Properties density="'rho'">

<Nested name="before_integration">
<Flexible_Tetrahedron_Interior_Forces>
<Nested name="reference_coordinate_computation">

<Tangential_Reference_Coordinates/>

</Nested>

<Nested name="interior_force_computation">

<Linear_Finite_Element_Interior_Forces
youngs_modulus="'E'" have_damping="false"
```

```
rayleigh_stiffness_damping_ratio="0"
rayleigh_mass_damping_ratio="0"
compute_element_stresses="false"
poisson="0.33"/>
</Nested>


</Flexible_Tetrahedron_Interior_Forces>
</Nested>
</Particle_Properties>
</Flexible_Tetrahedron>
</Nested>
</Tetrahedron_Mesh_Generator>
</Particles>

<Integrators>

<Newmark_Integrator_6DOF fixed_step_size="false"
init_step_size="1E-8"
max_step_size="1E-8" max_step_size_safety="0.1"
rel_tol="1e-11"/>

</Integrators>

<Visualization view_filename="AluminiumRod.view"/>

<Runtime_Parameters num_progress_notifies="5"
t_start="'sim_start'"
t_end="'sim_end'" check_initial_consistency="0"
smp_parallel="'IsParallel'" n_processors="'NProcessors'"/>
<Interactions>

<Tetrahedron_Surface_Bond_Interaction
stiffness="'bond_stiffness'"/>

<Tetrahedron_Boundry_Shape_Interaction/>
</Interactions>




</Particle_Simulation>
```

ÖZGEÇMİŞ

Adı Soyadı	:	Burak ER
Doğum Yeri ve Tarihi	:	Bursa, 1988
Yabancı Dili	:	İngilizce

Eğitim Durumu(Kurum ve Yıl)

Lisans	:	Uludağ Üniversitesi Makine Mühendisliği Bölümü, 2010
Ortaöğretim	:	Bursa Cem Sultan Lisesi(Y.D.A.), 2006
İlköğretim	:	Bursa Süleyman Çelebi İlköğretim Okulu, 2002

Çalıştığı Kurum/Kurumlar ve Yıl	:	Arş. Gör., Bursa Teknik Üniversitesi, 2011-
İletişim(e-posta)	:	burak.er@btu.edu.tr