Yapı Mühendisliği için Genişletilebilir Paralel Sonlu Elemanlar Çözümleme Platformu

Proje No: 108M586

Y. Doç .Dr. Özgür Kurç
Doç.Dr. Uğur Polat
Tunç Bahçecioğlu
Utku Albostan
Tolga Kurt
Andaç Lüleç
Semih Özmen

EKİM 2012
ANKARA

Önsöz

Bu projenin en önemli hedeflerinden bir tanesi büyük boyutlu modellerin yapısal mekanik problemlerinde sıkça kullanılan çözümleme yöntemleriyle çözülmesini sağlayan bir sonlu elemanlar yazılım platformunun geliştirilmesidir. Platformun genişletilebilir olması platformun proje sonrasında dahi çeşitli araştırmacılar tarafından geliştirilmesine ve farklı araştırma projelerinde kullanılmasına olanak vermektedir. Bunun yanı sıra, yazılım geliştirilirken paralel hesaplama tekniklerinin kullanılması sayesinde, platform herhangi bir paralel donanımda (paylaşık, dağıtık veya paylaşık/dağıtık karma sistemlerde) kullanılabilmekte, donanımın bellek kapasitesine ve çekirdek/işlemci sayısına göre çözümlenebilecek model boyutu büyümektedir. Ayrıca proje çerçevesinde yeni nesil ekran kartı işlemcilerinde de çalışabilecek çözücüler geliştirilmiş ve özellikle belirtik algoritmalarla zaman alanında doğrusal olmayan dinamik problemlerin çözümünde yüksek başarım elde edilmiştir. Platformun çok çeşitli sonlu elemanlarla zenginleştirilmesi sayesinde pratik yapı mühendisliği uygulamalarında kullanılan yapısal modeller de çözümlenebilmiştir. Proje süresince farklı paralel çözüm yöntemleri geliştirilmiş, sınanmış ve platform en iyi hale getirilmiştir. Son olarak platformun genişletilebilirliğinin sınanması amacıyla ısı transferi elemanları ve ilgili çözümleme algoritmaları platform motoruna dışarıdan eklenerek seri çözümlemelerde başarılı sonuçlar elde edilmiştir.

108M586 nolu TÜBİTAK projesi kapsamında; 1 Haziran 2009 tarihinde başlayan bu çalışma, öngörülen çalışma takvimine uygun olarak 1 Eylül 2012 tarihinde tamamlanmıştır. Çalışma, belirlenen amaç ve kapsama uygun olarak ilerlemiş ve bu süre boyunca TÜBİTAK tarafından desteklenmiştir. Bu çalışma kapsamında iki adet yüksek lisans tezi yürütülerek sonuçlandırılmış, iki adet yüksek lisans tezi ve bir adet doktora tezine de destek sağlanmıştır. Elde edilen sonuçlar ulusal ve uluslararası konferanslarda sunulan bildiriler aracılığı ile yayınlanmış, uluslararası dergilerde yayınlanmak üzere makale taslakları hazırlanmaktadır. Ayrıca proje kapsamında geliştirilen yazılım sayesinde Almanya Kassel Üniversitesi ile FP-7 SERIES projesi kapsamında bir araştırma işbirliği başlatılmıştır. İşbirliği bünyesinde yüksek başarımlı hesaplama teknikleriyle hibrit deney ile ekran kartları, GİB, kullanılarak dağıtık deney simulasyonları üzerinde çalışılmış ve ön deneyler başarıyla gerçekleştirilmiştir.Kasım 2012 tarihinde asıl deneyler gerçekleştirilecektir.

İçindekiler

Şekil Listesi

Tablo Listesi

Özet

Son yıllarda bilgisayar teknolojilerinde meydana gelen ilerlemeler sayesinde paralel bilgisayar sistemleri artık çok daha ucuz ve ulaşılabilir hale gelmiştir. Ancak birçok kurum ve mühendis, ellerinin altındaki paralel bilgisayar sistemlerini, kullandıkları yapısal çözümleme yazılımlarının yetersizliği nedeniyle kullanamamaktadır. Bu projenin de en önemli hedeflerinden biri, yapı mühendisliğinde sıkça kullanılan sonlu elemanlar çözümlemelerini paralelleştirerek hem hali hazırda varolan paralel hesaplama donanımlarının masraf yapılmadan kullanılmasını sağlamak, hem de çözümleme sürelerinde ciddi azalmalar elde etmektir. Bu projede sonlu eleman yönteminde sıkça kullanılan doğrusal statik, doğrusal olmayan statik, zaman alanında tanımlı doğrusal ve doğrusal olmayan dinamik çözümleme yöntemleri üzerinde çalışılmıştır. Paralel çözümleme yöntemleri olarak doğrudan çözücüler incelenmiş ve toptan ve alt-yapı temelli olmak üzere iki tip çözücü üzerinde çalışılarak çözücülerin başarımları sınanmıştır. Zaman alanında tanımlı dinamik çözümleme için örtük ve belirtik integrasyon algoritmalarının paralel uygulamaları geliştirilerek başarımları karşılaştırılmıştır. Belirtik algoritmalar kullanılarak zaman alanında doğrusal olmayan problemlerin de çözümlemesi gerçekleştirilmiştir.

Mekanik ve hesaplama konularıyla ilgili yazılım geliştirmenin en önemli sıkıntılarından bir tanesi, yazılım geliştiren ekibin dışındaki araştırmacıların veya kullanıcıların yazılımı kullanmalarının ve geliştirmelerinin çok zor olması, yazılıma uyum sağlayarak anlamalarının çok mesai gerektiren ve enerji tüketen bir iş olmasıdır. Bu sebepten dolayı da geliştirilen birçok akademik yazılım geniş bir kullanıcı kitlesi bulamamaktadır. Bu projenin diğer bir önemli hedefi de genişletilebilir bir yazılım yapısı oluşturarak, ileride başka araştırmacıların da kolayca kullanabilecekleri bir platform üretmektir. Bu amaçla, platform için nesne yönelimli bir veri yapısı geliştirilmiş ve parça ekleme (plug-in) teknolojisi kullanılarak yazılım platformuna dışarıdan parçalar eklenerek platformun genişletilmesine olanak verilmiştir. Platformun genişletilebilirliği ısı transferi probleminin çözümü için geliştirilen sonlu eleman ve ilgili çözümleme yöntemlerinin platforma eklenmesiyle sınanmıştır.

Son olarak yeni nesil ekran kartlarının sonlu elemanlar yönteminin kullanılabilirliği çeşitli çözümleme algoritmalarının GİB (grafik işlem birimi) uygulamalarıının geliştirilmesiyle sınanmıştır. Her ne kadar seyrek matris çözücülerde istenen başarım yakalanamadıysa, yoğun matris çözücülerinde ve belirtik integrasyon yöntemiyle zaman alanında doğrusal olmayan dinamik çözümlemelerde hız bakımından ciddi kazanımlar elde edilmiştir.

Anahtar Kelimeler: Yapısal çözümleme, paralel hesaplama, sonlu elemanlar, büyük modeller, bölümleme, alt-yapı, bilgisayar kümeleri, tektürel kümeler, paylaşımlı bellek, dağıtık bellek, GİB

Abstract

The parallel computing systems became more affordable and available in consequence of the recent development in computer technology. Many institutions and engineers, however, can not utilize already available parallel computer hardwares due to the insufficiencies of the structural analysis softwares that they were using. Thus, one of the main objectives of this project is presenting a way to utilize the existing parallel computing hardwares without the need of additional cost and creating a considerable reduction in the analysis times by parallelizing the most frequently utilized finite element analysis techniques in structural engineering. In this project, a sigficant effort was spent on the main analysis methods of finite element method such as linear static, non-linear static, linear and non-linear time history analysis. As paralel solution techniques of linear systems of equations, two different solution approach, i.e. globnal and substructure based were implemented and their performances are tested with several structural models. Likewise, for time history analysis of structures, both implicit and explicit time integration techniques were implemented and their parallel efficiency were tested. Parallel non-linear time history analysis algoritms were also implemented utilizing the explicit integration technique.

One of the main problems of developing a computational mechanics software is the difficulty of having the third parties other than the developers to use and further develop such softwares. Because of this reason, most of the academical softwares were being utilized only by a few researchers. Thus, the other important target of this project is to create an expandable software structure so that the framework can easily be utilized and further developed by other researchers. For this reason, an object-oriented data structure was carefully designed for such an analysis software and with the help of the state of the art ‘plug-in’ technolgy, external programs can be easily added to the analysis engine and utilized without any problems. In order to validate the extensibility of the developed analysis framework, finite elements and analysis methods for the heat transfer problems were developed and added to the framework as plug-ins.

As a final step, the use of GP-GPU’s in finite element analysis were examined by developing several analysis methods. Even though fast solution times for direct sparse matrix solvers were not obtained when compared to the performance of multi-core CPUs, significant reduction in solution times for dense matrix operations and explicit time integration methods were obtained.

Keywords: Structural analysis, parallel computing, finite elements, large models, partitioning, substructure, PC clusters, homogenous cluster, shared memory, distributed memory, GPU

1. Bölüm

Genişletilebilir Paralel Sonlu Elemanlar Çözümleme Platformu

Vizyon

Doksanlı yıllardan günümüze bilgisayar teknolojilerinde meydana gelen hızlı ilerlemeler bilgisayarların diğer birçok alanda olduğu gibi inşaat mühendisliği alanında da vazgeçilmez bir araç haline gelmesini sağlamıştır. Günümüzde bilgisayarlar, herhangi bir yapının tasarımının hemen hemen her aşamasında, özellikle de çözümleme (analiz) ve boyutlandırma sırasında yoğun bir biçimde kullanılmaktadır. Gerek bilgisayar donanımlarındaki gerekse yapısal çözümleme yazılımlarındaki ilerlemeler yüzbinlerce sonlu elemana sahip büyük yapısal modellerin kolaylıkla ve hızlıca oluşturulmasına olanak vermekte ve geçmiş yıllarda sadece hayal edilebilecek büyüklüğe sahip bu modeller artık günümüz bilgisayarları tarafından kabul edilebilir bir sürede çözülebilmektedir. Bu durum, mühendislerin gittikçe daha detaylı, büyük ve karmaşık yapısal modeller yaratmasına ve daha detaylı çözümleme yöntemlerine yönelmelerine yol açmaktadır. Diğer taraftan işverenler ise yapı tasarımının geçmişe nazaran çok daha kısa bir zaman içerisinde bitirilmesini istemekte ve mühendislere bu konuda büyük baskılar uygulamaktadır. Ayrıca inşaat mühendisliği şartnameleri de gelişen teknolojilere ve bilgi düzeylerine göre yenilenmekte ve zamanla yapıların daha detaylı çözümlemelerinin yapılmasını şart koşmaktadır. Sonuç olarak, bir yapısal modelin boyutu büyüdükçe daha fazla çözüm alanına ve zamanına ihtiyaç duyduğu ve çözümleme yöntemlerinin detay düzeyi arttıkça daha fazla işlem gerektirdiği düşünülürse, bilgisayar donanım ve yazılım teknolojilerindeki gelişmeler ile kullanıcı istekleri arasındaki yarışın hala devam etmekte olduğu ve daha hızlı hesaplama yöntemlerine hala büyük bir ihtiyaç duyulduğu görülecektir.

Herhangi bir yapının tasarımı yinelgen bir süreç olup, bu süreç içerisinde yapıda birçok değişiklik meydana gelir. Bu değişikliklerin sebebi her zaman mukavemet ve kararlılık gibi yapısal nedenlerden değil, aynı zamanda mimari, ekonomik veya üretim kolaylığı gibi birçok farklı meslek grubunun ihtiyaçları doğrultusunda da olabilir. Yapının taşıyıcı sisteminde meydana gelen en ufak bir değişiklik, yapısal modelin yeniden oluşturulmasına, çözümlenmesine ve yapısal elemanların tekrar tasarlanmasına yol açar. Bu süreç uzun sürer ve yorucudur. Standart bir yapının yapısal tasarım süreci dört farklı adımla özetlenebilir. İlk adım, mimari çizimler kullanılarak yapının matematiksel modelinin kurulmasıdır. Modelleme olarak da adlandırılabilen bu adımdan sonra çözümleme adımı gelir. Çözümleme sonucunda herhangi bir dış etkinin yapı üzerindeki etkileri elde edilir. Çözümleme yöntemi daha detaylı hale geldikçe, gerçek yapı elemanlarını modellemek için kullanılan matematiksel elemanların sayısı arttıkça ve mühendisin doğru bir modelleme yaptığı varsayılırsa elde edilecek sonuçların teorik sonuca yaklaşacağı kabul görmüş bir yaklaşımdır. Üçüncü adımda yapısal elemanlar çözümleme sonuçları kullanılarak boyutlandırılır. Yapı geometrisinde herhangi bir değişiklik olmasa bile, ikinci ve üçüncü adımlar tasarım süreci içerisinde kaliteyi yükseltmek ve maliyeti düşürmek amacıyla birçok defa tekrarlanır. Tasarım son haline geldikten sonra dördüncü adımda yapısal elemanlar detaylandırılır ve paftalar hazırlanır. Eğer yapı

yüksek bina, baraj veya uzun açıklıklı köprü gibi özel yapı statüsündeyse ve ciddi deprem riski içeren bir bölgedeyse, bu durumda yapısal tasarım doğrusal olmayan dinamik çözümleme yöntemleriyle sınanmakta, yapı-zemin etkileşimi, deprem sırasında doğrusal olmayan enerji sönümleme mekanizmaları gibi yapıya özel durumlar anlaşılarak tasarım son haline getirilmektedir.

Hesaplama bakış açısından standart bir yapının tasarım aşaması içerisinde en çok vakit alan kısım çözümlemedir. Özellikle modelde kullanılan matematiksel elemanların sayısı arttıkça çözümleme süresi katlanarak artacaktır. En temel çözümleme yöntemi olan doğrusal statik çözümlemede yapısal modelin davranışını anlatan direngenlik (rijitlik) matrisinin sadece bir defa oluşturulması, ayrıştırılması (factorization) ve her bir yüklemenin (dış etkinin) sonuçlarının bir defa çözülmesi yeterlidir. Ancak doğrusal olmayan veya zaman alanında tanımlı dinamik gibi daha detaylı çözümleme yöntemleri kullanıldığı zaman direngenlik matrisinin birçok kez ayrıştırılması ve/veya sistemin on binler hatta yüz binlerle ifade edilebilecek sayıda çözülmesi gerekmektedir. Doğrusal statik çözümleme süresi dakika mertebesinde olan bir modelin doğrusal olmayan veya zaman alanında tanımlı dinamik çözümlemesi günlerce sürebilir. Dolayısıyla, çözümleme aşamasında kaybedilen süreyi azaltabilecek herhangi bir gelişme, tasarım mühendisine yapı kalitesi ve güvenliği için tasarım süreci içerisinde daha fazla alternatifi inceleme şansı tanıyacak ve böylece yapıların daha kaliteli bir mühendisliğe sahip olmasına yardımcı olacaktır.

Çözümleme adımında harcanan vakti azaltmanın bir yolu paralel hesaplama yöntemlerini sonlu elemanlar çözümlemelerinde kullanmaktır. Günümüzde paralel hesaplama yapabilecek dört çekirdekli işlemciler artık asgari donanım olarak satılmakta ve piyasada on altı çekirdekli işlemciler rahatça bulunabilmektedir. Ayrıca birçok mühendislik bürosunda bilgisayarlar yerel ağla birbirlerine bağlıdır. Dolayısıyla bu tip sistemler için uyarlanmış bir sonlu elemanlar yazılımı, hem büyük yapısal modellerin detaylı çözümlemelerinin çok daha kısa bir sürede sonuçlandırılmasına olanak verip farklı boyutlandırma seçeneklerinin gözden geçirilmesini sağlayacak hem de büroların fazladan para harcamadan hali hazırda sahip oldukları bilgisayar sistemleriyle yüksek başarımlı hesaplama yapmalarına olanak verecektir. Bu projenin de en önemli hedefi, birbirlerine yerel ağla bağlı bilgisayarlardan oluşan veya çok çekirdekli herhangi bir bilgisayar kümesinde etkin çalışabilecek, inşaat mühendisliği modellerinin detaylı çözümlemelerinde kullanılabilecek bir sonlu elemanlar çözümleme platformunun geliştirilmesidir.

Literatür Özeti

İnşaat mühendisliği yapılarının tasarımı sırasında genelde doğrusal statik, özdeğer (modal), doğrusal olmayan statik, doğrusal ve doğrusal olmayan dinamik çözümlemeler kullanılmaktadır. Bütün bu çözümleme yöntemlerinin ortak yanı, her birinin doğrusal denklem sistemlerinin bir defa veya ard arda çözümlerine ihtiyaç duymasıdır. Bu adım genelde çözümlemenin en çok vakit harcanan bölümlerinden bir tanesidir. Günümüzde doğrusal denklem sistemlerinin çözümü için geliştirilmiş birçok paralel çözüm yöntemi bulunmaktadır. Bu yöntemler çözüm yaklaşımlarına göre eleman (Bitzarakis et al.,1997), alt-yapı (substructure) (Farhat et al. 1994, Farhat et al. 2000, Fulton ve Su 1992, Kurc ve Will 2005) ve toptan (global) (Amestoy et al. 2000, Ashcraft et al. 1990, Ashcraft ve Grimes 1999) düzeyinde olmak üzere üç ya da çözüm algoritmalarına göre doğrudan (direct) (Amestoy et al. 2000, Ashcraft et al. 1990, Heath ve Raghavan 1994) ya da yinelgen (iterative) (Farhat et al. 1994, Bitzarakis et al. 1997) olmak üzere iki sınıfta incelenebilir.

Daha önce de belirtildiği gibi inşaat mühendisliğinde, yapısal modellerin çözümlenmesinde en sık kullanılan yöntem, birden fazla yüklemeyi içeren doğrusal çözümlemedir. Yinelgen çözüm yöntemleri bu amaç için uygun değildir çünkü her bir yükleme için ayrı çözüm yapılması gerekmektedir. Ayrıca bir boyutlu elemanlara sahip yapısal modellerin yinelgen paralel çözümü de çoğu kez mümkün

olamamaktadır (Escaig et al. 1994). Yinelgen yöntemler genel olarak doğrusal olmayan çözümlemeler için tercih ediliyor olsa da bu yöntemlerin başarımı kullanılacak olan ön işlem yöntemine (pre-conditioning) bağlıdır (Farhat et al. 1998). Kullanılacak olan ön işlem yönteminin modelin malzeme, geometrik ve yapısal özelliklerine bağlı olarak değiştiği düşünülürse (Bitzarakis et al. 1997), yinelgen yöntemlerin pratik mühendislik uygulamalarında kullanılması çok zordur.

Doğrudan çözüm yöntemlerinde ise iki yöntem ön plana çıkmaktadır, alt-yapı tabanlı ve toptan çözüm yaklaşımları. Seyrek matrisler için geliştirilen toptan çözüm yöntemlerinde aşamalı çoklu-sınır (multi-frontal) çözüm algoritmaları öne çıkmaktadır. Bu algoritmaların dayandığı en önemli nokta, yapının sistem matrisine ait denklemlerin sıralaması ayarlanarak, ayrıştırma hesabının sınır matrisleri olarak adlandırılan (frontal matrices) küçük boyutlu yoğun matris işlemleriyle gerçekleştirilmesidir. Bu sayede hem bellek kullanımı sırasında yaşanan başarım kayıplar azaltımış hem de işlemler sırasında hız için en iyileştirilmiş yoğun matris kütüphanelerinin kullanılması mümkün olmuştur (Dongarra et al. 1998, Amestoy et al. 1996, Amestoy et al. 2001, Schenk ve Gartner 2002). Dongarra et al. (1998) ise bu tip doğrudan seyrek matris çözücülerinin sistem, matris ve alt-matris düzeyi olmak üzere üç seviye parallellik kullandığını savunmaktadır. Sistem düzeyi parallellikte denklemlerin gruplanarak birbirinden olabildiğince bağımsız hale getirilerek her bir grubun aynı anda farklı işlemciler tarafından çözümlenmesi mümkündür. Matris seviyesindeki paralellik ise bu sefer birbirinden ayrılan gruplardaki matrislerin içindeki bağımsız parçaların birleştirme ağaçları sayesinde belirlenerek çözülmesidir. Son olarak çözümün küçük boyutlu yoğun sınır matrisleri üzerinde gerçekleştirilmesi PBLAS veya SCALAPACK gibi en iyileştirilmiş kütüphanelerin kullanılmasına olanak vermesidir. Bu tip seyrek matris çözücüleri dört ana işlem aşamasına sahiptirler: (1) Denklemlerin Sıralanması, (2) Sembolik Ayrıştırma, (3) Sayısal Ayrıştırma, (4) Yük Vektörlerinin Çözümü. Denklemlerin sıralanması aslında paralel çözümün başarımını belirleyen en önemli faktör olup bu aşama hem işlem sayısını ve paralel çözüm sırasını belirleyen birleştirme ağacının (assembly or elimination tree) şeklini hem de çözüm sırasında kullanılacak bellek miktarını belirlemektedir. Guermouche et al. (2003) tarafından gerçekleştirilen bir çalışmada farklı denklem sıralama algoritmalarının çoklu sınır çözüm yönteminin başarımına olan etkileri araştırılmıştır. Çalışmada AMD (Amestoy et al. 1996), AMF (Amestoy et al. 2000), PORD (Schulze 2001), METIS (Karypis ve Kumar 1998) ve SCOTCH (Pellegrini ve Roman 1996) olmak üzere beş farklı sıralama yönteminin oluşturdukları birleştirme ağacının yapısı ve sonucunda elde edilen çözüm süresi ve bellek kullanımı incelenmiştir. Sonuç olarak SCOTCH ve METIS kütüphanelerinde sıralama yöntemlerinden paralel hesaplama için uygun geniş ve dengeli ağaç yapısı elde edilmiştir. Çalışmadan elde edilen bir diğer sonuç ise bu tip çözücülerin bellek kullanımının yüksek olduğu ve paralel başarımlarının birleştirme ağacının yapısına bağlı olduğudur.

Literatür doğrudan seyrek çözücülerin geliştirilmesi üzerine çok sayıda yayın ve çeşitli yazılım kütüphaneleri bulunmaktadır. Paralel seyrek çözücülerde üç farklı yazılım kütüphanesi öne çıkmaktadır, SuperLU Dist (Li 2005), MUMPS (Amestoy et al. 2000) ve Taucs (Irony et al. 2004). SuperLU Dist (Li 2005) birleştirme ağaclarında süper noktalar adı verdiği, işlemlerin birbirinden bağımsızlaştığı kritik noktaları tespiti üzerinden çözüm işlemini gerçekleştirir. Bloklar üzerinden gerçekleştirilen çözümde her bir süper nokta matristeki ayrıştırılan blokları göstermektedir. Ayrıştırılan her bir blok kullanılarak diğer bloklardaki veriler güncellenir. Aşamalı çoklu sınır çözücü yöntemini kullanan (Amestoy et al. 2000) kütüphanesi ise birleştirme ağacının en alt seviyesindeki noktalara ait yoğun sınır matrislerinin Schur bileşenlerinin hesaplanması ve aynı seviyedeki Schur bileşenlerinin birleştirilerek bir üst seviyedeki sınır matrisinin elde edilmesi presibine dayanmaktadır. Çoklu sınır çözücü yöntemini kullanan bir diğer çözüm kütüphanesi, Taucs, MUMPS'dan farklı olarak yoğun sınır matrislerinin Schur bileşenlerini hesaplamak yerine matris kolonlarının ayrıştırılması prensibine kullanmaktadır. Bu yöntemde birleştirme ağacının her bir noktası matrislerde ayrıştırılacak kolon gruplarına denk gelmektedir. SuperLU ve MUMPS kullanılarak gerçekleştirilen karşılaştırmalı çalışmalarda (Berglund ve Leeuw 2006), SuperLU yöntemiyle elde edilen

blok boyutlarının, MUMPS'ın blok boyutlarından küçük olduğu, bu sebepten dolayı yoğun matris işlemlerinde verimliliğin, dolayısıyla çözüm hızının azaldığı raporlandırılmıştır. Diğer taraftan, çözüm sırasında kullanılan işlemci sayısının artması MUMPS'ın bellek kullanımını ciddi oranda arttırdığı ve artan iletişim miktarı yüzünden çözüm verimliliğinin azaltdığı da gözlenmiştir.

Doğrudan (direct) çözüm algoritmaları kullanılarak alt-yapı tabanlı çözümleme yaklaşımında ise (Kurc ve Will 2005) herhangi bir yapısal model ilk olarak alt-yapılara bölünür. Her bilgisayara tek bir alt-yapının veritabanı dağıtıldıktan sonra her bilgisayar aynı anda kendi alt-yapısının direngenlik (stiffness) matrislerini oluşturur, serbestlik derecelerini alt-yapı sınırlarına indirger (condensation) ve sınırlardaki deplasmanlar hesaplandıktan sonra kendi alt-yapılarına ait elemanların eleman kuvvetlerini ve gerilmelerini hesaplar. Bütün bu hesaplamalar sırasında bilgisayarlar arası veri iletişimine gerek duyulmaz. Serbestlik derecelerinin indirgenmesi sırasında doğrudan çözüm algoritmaları kullanıldığı için her çeşit ve şekildeki yapısal modeller çözümlenebilecek ve alt-yapı şekli de herhangi bir soruna yol açmayacaktır (Farhat et al. 2000).

Alt-yapı tabanlı çözümleme yöntemleri, bir yapısal modeli alt-yapılara ayıracak bölümleme algoritmalarına ihtiyaç duyar. Alt-yapı şekillerinin ve boyutlarının, paralel çözümlemenin başarımını önemli ölçüde etkilediği düşünülürse, kullanılacak olan bölümleme algoritması çözüm süresi bakımından son derece önemlidir. Günümüzdeki bölümleme algoritmaları üç ana grupta incelenebilir. İlk grup algoritmalar yapının düğüm noktası koordinatları gibi geometrik bilgileri kullanarak bölümleme yapar. Örneğin Berger ve Bokhari (1987) tarafından geliştirilen RCB yönteminde yapısal model ilk olarak bir doğru ile ortadan ikiye ayrılır. Daha sonra ortaya çıkan iki parça da ortadan ikiye ayrılır. Bu işlem istenen parça sayısı elde edilinceye kadar sürdürülür. İkinci grup algoritmalar ise yapısal modelin elemanları arasındaki bağlantıları kullanır. Farhat (1988) tarafından geliştirilen ve Greedy algoritması olarak adlandırılan yöntemde, bir başlangıç düğüm noktası seçildikten sonra uygun komşu düğüm noktaları eklenerek ilk alt-yapı oluşturulmaya başlanır. Düğüm noktaları ilk alt-yapı için önceden belirlenmiş olan düğüm noktası sayısına ulaşılıncaya kadar alt-yapı eklenmeye devam edilir. İlk alt-yapının sınır düğüm noktalarından birine komşu diğer bir diğer düğüm noktasından başlanarak ikinci alt-yapı oluşturulmaya başlanır. Bu işlem tüm alt-yapılar oluşturuluncaya kadar devam ettirilir.

Son grup algoritmalar ise yapısal modelin grafiksel gösterimini kullanırlar. Grafiksel gösterim noktalardan (vertex) ve doğrulardan (edge) oluşur. Noktalar genel olarak sistemdeki çözüm noktalarını, doğrular ise çözüm noktaları arasındaki ilişkiyi tanımlarlar. Ayrıca noktalara ve doğrulara bir ağırlık değeri tanımlanaraktan noktalar veya doğrular arasında göreceli farklılıklar yaratmak mümkün olabilmektedir. Literatürde yapısal modellerin bölümlenmesi için dört farklı grafiksel gösterim yöntemi kullanılmıştır: Noktasal grafik (nodal graph) (Kurc ve Will 2007, Hsieh et al. 1995), eleman grafiği (dual graph) (Topping ve Ivanyi 2001, Hendrickson ve Kolda 2000), iletişim grafiği (communication graph) (Hendrickson ve Kolda 2000, Hsieh et al. 1995] ve küresel grafik (bubble graph) (Topping ve Ivanyi 2001). Değişik grafiksel gösterim yöntemlerinin avantajları, çözüm süresine ve veri transferi boyutuna olan etkileri çeşitli çalışmalarda irdelenmiştir (Kurc ve Will 2007, Topping ve Ivanyi 2001).

Grafiksel gösterimi kullanarak bölümleme yapmanın bir yolu bölümleme problemini matematiksel olarak tanımlayıp çözmektir. Literatürde bu tip problem NP-Tam (NP-Complete) olarak tanımlanmış olup, problemin ulaşılabilir bir sonucu olduğu ortaya konmuştur (Driessche ve Roose 1995, Garey et al. 1976). Ancak tam çözüme gitmek için sistemi tanımlayan grafiksel gösterim matrisinin ikinci en küçük özdeğerine (eigenvalue) karşılık gelen özvektörün (eigenvector) hesaplanması gerekmektedir. Bu yöntem çok fazla hesaplama gerektiren çözümlerden önce kullanılabilir aksi halde bölümleme için harcanan vakit çözüm süresini geçebilir.

Bir diğer alternatif yöntem ise en iyi bölümlemeyi hedeflemek yerine iyi bir bölümlemeyi çok daha kısa bir sürede bulabilmektir. Bu amaçla geliştirilen çok-aşamalı (multilevel) bölümleme yaklaşımında, grafiksel gösterimin boyutu aşama aşama küçültülür. Olabilecek en küçük boyutlu grafiksel gösterimde bölümleme gerçekleştirilir ve elde edilen her parçanın boyutu, ilk boyutuna ulaşıncaya kadar aşama aşama büyütülür. Bu yaklaşımla çalışan METIS (Karypis ve Kumar 1998) bölümleme kütüphanesi çok hızlı bir şekilde bölümleme gerçekleştirebilen çeşitli bölümleme algoritmalarına sahiptir.

Alt-yapı yaklaşımının en önemli sorunlarından bir tanesi, bir yapısal modelin her bilgisayarda serbestlik derecesi indirgeme süresini eşitleyecek şekilde alt-yapılara bölünmesidir (Hendrickson 2000). Aksi takdirde sistemdeki tüm bilgisayarların çözüme devam edebilmek için en yavaş ya da en yüklü bilgisayarın işini bitirmesini beklemesi gerekecektir (Kurc ve Will 2007). Bu durum alt-yapı tabanlı çözüm yöntemlerinin verimliliğini önemli ölçüde düşürmektedir. Ne yazık ki günümüzde doğrudan çözüm algoritmaları için serbestlik derecelerini indirgeme sürelerini eşitleyebilecek şekilde alt-yapı oluşturabilen bölümleme (partitioning) yaklaşımı henüz geliştirilememiştir (Hendrikson ve Kolda 2000). Yukarıda bahsedilen bölümleme algoritmalarının hepsi bahsedilen amaç için yetersizdir çünkü alt-yapılar oluşturulmadan alt-yapı indirgeme süresinin hesaplanamamasıdır. Bu konuda Fulton ve Su (1992) ve Escaig et al. (1994) paylaşımlı bellekli (shared memory) bilgisayarlar için yöntemler önermiş olsalar da bu yöntemleri dağıtık bellekli (distributed memory) sistemlerde uygulamak mümkün değildir.

Dağıtık bellekli sistemler için Pınar ve Hendrikson (2001) genel bir bölümleme ve bölüştürme algoritması önermiş ve bu algoritmanın alt-yapıların serbestlik derecesi indirgeme sürelerini eşitlemek için de kullanılabilineceğinden bahsetmiştir. Ancak bu algoritmayı farklı uygulamalarda test etmiştir. Yang ve Hsieh (2002) ise benzer bir yaklaşımı alt-yapıların serbestlik derecesi indirgeme sürelerini eşitlemek amacıyla kullanmıştır. Tek bir bilgisayarda çalışan yöntemleri alt-yapıların serbestlik derecesi indirgeme sürelerini birbirine yaklaştırabilmiş olsa da bu eşitleme için harcanan süre bu yöntemi doğrusal çözümleme için kullanmayı olanaksız kılmıştır. Kurç ve Will'in (2007) geliştirdikleri yöntem ise çok kısa sürede bu eşitlemeyi gerçekleştirmiş ve toplam çözüm süresinde önemli bir iyileşme sağlamıştır. Daha sonra Kurç (2010), çoktürel bilgisayar kümeleri için bir veri hazırlama yöntemi önermiş ve çeşitli örnek modellerde indirgeme sürelerinde ciddi azalmalar elde etmiştir.

Alt-yapı tabanlı çözüm yöntemlerinin bir diğer önemli sorunu sınır denklemlerinin etkin bir şekilde oluşturulup çözülmesidir. Düşük sayıda bilgisayar kullanılarak gerçekleştirilen alt-yapı tabanlı çözümlemelerde sınır direngenlik matrisi yoğundur. Bu tip matrislerin doğrudan çözümleri için tek boyutlu ve iki boyutlu çözüm yaklaşımları mevcuttur. Tek boyutlu çözücüler matrisi kolon-kolon veya sıra-sıra ayrıştırırken çift boyutlu çözücüler ayrıştırmaları matrisi daha küçük alt matrislere ya da diğer bir deyişle bloklara ayırarak gerçekleştirirler (Aschcraft et al. 1990, Ashcraft 1993). Schreiber'in seyrek matrisler için dağıtık sistemlerde gerçekleştirdiği karşılaştırmalı çalışmada, bilgisayar sayısı arttıkça tek boyutlu çözücülerden iki boyutlu çözücülere yönelinmesi gerektiği sonucuna varmıştır. Kurç (2006) ise tek boyutlu çözüm yöntemlerinden paralel değişken sıra tabanlı çözücülerin alt-yapı sınır denklemleri çözümünde bilgisayar sayısı arttıkça ciddi başarım düşüklüğüne yol açtığını gözlemlemiştir. Benzer şekilde Hsieh et al. (1995) da paralel kolon tabanlı çözücüyle alt-yapı sınır denklemlerini çözmeyi denemiş ve önemli bir başarım elde edememiştir. Choi et al. (1992) tarafından geliştirilen iki boyutlu çözüm stratejisini kullanan döngüsel-blok (block-cyclic) çözücüsü bugün SCALAPACK kütüphanesinde yoğun matrislerin çözümünde kullanılmaktadır. Çok yönergeli çok verili (MIMD) sistemler için geliştirilmiş olan ScaLAPACK, LAPACK (–ki bu kütüphanede üstün başarımlı BLAS yöntemlerine dayanan yaygın olarak kullanılan bir kütüphanedir-) yöntemlerinin paralel mesajlaşma yöntemleriyle bilgisayar kümelerinde çalıştırılmasını ve bu sayede ölçeklendirilebilirliği arttırmayı amaçlamaktadır (Choi et al. 1994). Ancak çalışmalar göstermiştir ki ScaLAPACK, çoktürel bilgisayar kümelerinde tektürel kümelerde elde ettiği başarımı yakalayamamıştır. Bu nedenle ScaLAPACK'in çoktürel kümelerde kullanımında bilgisayarlar arası

veri dağılımının iyileştirmesi gerekmiştir (Beaumont et al. 2001). Yang et al. (2012) ise aşamalı alt-yapı tabanlı bir çözümleme yöntemi önermiş ve bu sayede yoğun sınır matrisleri çözümünde önemli kazanımlar elde etmiştir.

Bahçecioğlu et al. (2009) tarafından gerçekleştirilen çalışmada doğrudan toptan ve alt-yapı tabanlı çözüm algoritmalarının başarımları doğrusal statik çözümleme yöntemi kullanılarak karşılaştırılmıştır. Toptan seyrek matris çözücü olarak MUMPS, alt-yapı tabanlı çözücü için ise Özmen (2009) tarafından geliştirilen alt-yapı tabanlı çözücü kullanılmıştır. Çalışmada farklı geometrik karmaşıklığa sahip büyük boyutlu yapısal modeller kullanılmış ve hesaplamalar sırasında dört çekirdekli işlemciye sahip sekiz bilgisayardan oluşan bir bilgisayar kümesi kullanılmıştır. Çalışma sonucunda yapısal modelin geometrik özelliklerine göre çözücülerin başarımının değiştiği, simetrik ve düzgün modellerde toptan çözücü sürelerinin alt-yapı tabanlı çözümlere göre büyük bir üstünlük sağladığı görülmüştür. Kullanılan bilgisayar sayısının ikinin katı olmadığı ve model geometrisinin karmaşıklaştığı durumlarda ise alt-yapı tabanlı çözüm yönteminin başarımının artarak toptan çözüm yöntemi hızını yakaladığı görülmüştür.

Literatürdeki paralel dinamik çözümlemelerle ilgili çalışmalar genellikle belirtik algoritmalar kullanılarak gerçekleştirilmiştir (Krysl ve Belytschko 1998). Özellikte Belytschko et al. (2004) tarafından geliştirilen belirtik integrasyon algoritması sayesinde belirli bir sönüme sahip sistemlerin bile sistem matrislerinin oluşturulmadan sadece eleman düzeyinde iç kuvvet hesaplayaraktan çözümlenmesine olanak vermiştir. Bu sayede paralel dinamik çözümlemelerde herhangi bir yapının eşit sayıda elemanlara sahip olacak şekilde bölümlenmesi bilgisayarlar arası eşit iş yükü bölüştürülmesi için yeterli olmuştur. Belirtik algoritmaların bir diğer avantajı da doğrusal olmayan sistemlerin çözümlenmesine göze çarpmaktadır. Denklem sistemlerinin bütün parametrelerin bilindiği zaman adımı için kuruluyor olması, doğrusal olmayan çözümlemede sistemin dengeye getirilmesi için yineleme ihtiyacını ortadan kaldırmaktadır (Belytschko et al. 2004). Diğer taraftan çözümlemenin sayısal kararlılığı için kullanılacak zaman adımının büyüklüğünün sınırlı olması, çözümleme için örtük algoritmalara göre daha fazla zaman adımı kullanılmasını gerektirmektedir. Örtük algoritmalar kullanıldığı zaman ise sistem matrislerinin oluşturulup ayrıştırılması gerekmektedir (Cook et al. 2001). Doğrusal olmayan dinamik çözümlemelerde ise örtük algoritmalar zaman adımı esnekliğinin getirdiği avantajlarını kaybedebilmektedirler. Örtük algoritmada sistem matrisleri bir sonraki zaman adımında oluşturulduğu için doğrusal olmayan problemlerde yapının denge durumunun sağlanması için yinelemelere ihtiyaç duyulmaktadır. Özmen et al. (2010) tarafından gerçekleştirilen bir çalışmada, doğrusal dinamik çözümleme için örtük algoritma kullanılarak toptan ve alt-yapı tabanlı çözücülerinin başarımları kıyaslanmıştır. Çalışmada simetrik ve düzgün bir kare plak modeli farklı sayıda bilgisayarlar kullanılarak çözümlenmiş ve her iki yöntemle de birbirlerine çok yakın sürelerde çözüm elde edilmiştir.

Genel amaçlı grafik işlemci birimlerinin, GA-GİB (General Purpose Graphical Processing Unit, GP-GPU), bilimsel hesaplama amaçlı kullanımı son yıllarda ciddi oranda artmıştır. Grafik işlemci birimlerinin çalışma prensibi merkezi işlemci birimlerinden (Central Processing Unit, CPU) farklıdır. NVidia tarafından geliştirilen GİB'ler ve CUDA yazılım geliştirme ortamı, bir iş parçacığının (İng. *Thread*) ızgara şeklinde dizilmiş, 32'li gruplar halinde ortak bir paylaşımlı belleğe sahip çekirdek öbekleri (İng. W*arp*) tarafından aynı anda çalıştırılması temeline dayanmaktadır (Damien 2012). CUDA çekirdekleri çalışırken veriye birden fazla bellekten erişebilirler. Her çekirdeğin çalıştıracağı iş parçacığı için yazmaçları (*Registers*) yetmediğinde kullanabileceği sadece bu çekirdeğin ulaşabildiği bir yerel bellek mevcuttur. Bu hafızanın yanı sıra her işlemcinin dâhil olduğu çekirdek öbeğindeki diğer çekirdeklerle veri alışverişini ve koordinasyonunu sağlayabilmesi amacıyla kullanılabilen her bir çekirdeğin aynı anda erişebileceği paylaşımlı bir bellek bulunmaktadır. Son olarak, grafik işlem birimindeki bütün çekirdeklerin erişebildiği erişim hızı olarak yavaş olan ancak kapasite olarak en büyük hafızayı sağlayan genel bir aygıt belleği de yer almaktadır (CUDA 2012).

GAGİB'lerin sahip olduğu çok işlemcili-tek işlemli donanım yapısı farklı mühendislik problemlerinde MİB'lere nazaran çok daha etkili olabilmektedir. Örneğin yoğun matrislerle ilgili hesaplamalar GAGİB mimarisi için çok uygundur. Literatürde GAGİB kullanılarak gerçekleştirilen, yoğun matris işlemleri üzerine yapılmış birçok araştırma mevcuttur (Jung 2006, Cao et al. 2009, Tomov et al. 2010, Ltaief et al., Volkov ve Demmel, 2008). Bu alandaki öncü çalışmalardan birisi [6], günümüzde mevcut olan CUDA benzeri genel amaçlı GİB programlama modelleri olmadan gerçekleştirilmiş ve bu çalışmada, araştırmacı GİB programlamanın zorluğuna ve GİB'lerin çift duyarlıklı (*Double Precision*) işlem hassasiyetine sahip olmamasına değinmiştir. Daha sonra, Cao yaptığı çalışmada (2009), yoğun matris çözümünü GİB'den destek alarak hızlandırmaya çalışmıştır. Tekrarlayan düzeltme (*Iterative Refinement*) aracılığıyla, GAGİB'lerin çok daha hızlı olduğu tek duyarlıklı (*Single Precision*) işlemler kullanarak çift duyarlıklı işlem hassasiyetinde sonuçlar elde etmiştir. Ayrıca, MİB ve GİB'e eş zamanlı olarak işlem yaptırarak elde ettiği başarımı daha da arttırmıştır.

Zamanla GİB'ler için genel amaçlı programlama modellerinin ortaya çıkması ve tek duyarlıklı işlem hassasiyetine nazaran çok daha yavaş olsa da çift duyarlıklı işlem hassasiyetinin de desteklenmesi birçok çalışmanın önünü açmıştır. Tomov kapsamlı bir çalışma (2010) yaparak MİB-GİB karma sistemlerden nasıl daha iyi yararlanılabileceğini araştırmıştır. Elde ettiği bulgular ışığında seri işlemciler için geliştirilmiş olan BLAS fonksiyonlarını (LAPACK 2012) MİB-GİB karma sistemler için eniyilemiştir. Sonrasında ise bu kernel fonksiyonlarını kullanarak Cholesky, LU ve QR ayrıştırmalarının da yer aldığı MAGMA kütüphanesini geliştirmiştir (MAGMA 2012). Aynı çalışma grubu tarafından bu kütüphanenin, birden fazla GİB bulunan sistemlerdeki başarımını arttırmak amacıyla yapılan araştırmada (Ltaief et al.) çift katmanlı iç içe bir paralellik önerilmiştir. İlk katman donanımlar arası veri alış-verişini belirlerken, ikinci katman ihtiyaca göre MİB-GİB karma ortamını da kullanabilecek şekilde GİB'ler arasında iş yükü dengesini belirlemektedir. Bu çalışmanın sonuçlarına ileriki bölümlerde değinilecektir. GİB'lerin vektör işlemcilere olan mimari benzerliklerini vurguladıkları çalışmalarında, Volkov ve Demmel'in (2008), geliştirdikleri algoritma, veriyi en hızlı şekilde aktarmaya ve veriye en hızlı şekilde erişmeyi hedeflerken kernel fonksiyon çağırma maliyetini (İng. *Kernel Startup Latency*) de en aza indirmeye çalışmıştır. Ayrıca, çalışmalarında MİB ve GİB'den eşzamanlı olarak hesap yapabilecek şekilde yararlanmışlardır. Son olarak EM Photonics firması LAPACK kütüphanesinden (LAPACK 2012) birçok fonksiyonu içeren yalnız GİB kullanarak çalışan CULA isimli yoğun matris kütüphanesini (CULA 2012) ticari bir ürün olarak geliştirmiştir. Özmen ve Kurç (2012) ise GAGİB'ler için hem alt-yapı tabanlı çözücülerde sınır denklemlerinin çözümü aşamasında, hem de çoklu sınır yönteminde sınır matrislerinin ayrıştırılmasında kullanılabilecek bir indirgeme yöntemi geliştirmişlerdir. Çeşitli matris boyutları için gerçekleştirilen karşılaştırmalı çalışmalar sonucunda, GA-GİB çözümlerinin Intel MKL kullanılarak yapılan hesaplamalara göre iki-sekiz kat arası hızlı olduğu görülmüştür.

Seyrek matrix çözümü için GA-GİB kullanımıyla ilgili çalışmalar genelde yinelgen yöntemlere yoğunlaşmıştır. Bolz et al.'un (2003) iki farklı yinelgen yöntemin GİB uygulamalarını yapıp MİB çözümleriyle karşılaştırdıkları çalışmada, GİB uygulamasında MİB'lere nazaran daha hızlı çözümler elde edilmiştir. Krüger ve Westermann (2003), Bautois et al. (2009) ve Couturier ve Domas (2011) benzer şekilde seyrek matrisler için yinelgen çözüm yöntemlerini GİB'lere uygulamış ve genelde MİB'lerden daha hızlı sonuç elde etmişlerdir. GİB'lerde seyrek matrisler için doğrudan çözüm yöntemi Lucas et al. (2010) tarafından geliştirilmiştir. Aşamalı çoklu sınır yönteminin farklı bir uygulamasının yapıldığı bu çalışmada GİB'ler MİB'lerle beraber kullanılmıştır. Çözüm sırasında elde edilen küçük boyutlu sınır matrislerinin ayrıştırılması için MİB'ler kullanılmış, sınır matrislerinin boyutları büyüdükçe yoğun matris ayrıştırma işlemleri için GİB'ler tercih edilmiştir. Elde edilen çözüm süreleri, sadece MİB kullanılarak gerçekleştirilen çoklu sınır yöntemi süreleriyle karşılaştırıldığında, tek işlemcili çözüm göre 5.91 kat, sekiz işlemcili çözüme göre ise 1.34 kat hızlanma elde edilmiştir. Andaç (2011) ise sadece GİB kullanan iki farklı seyrek

matris çözücü geliştirmiş ve çözücülerin başarımını MUMPS sonuçlarıyla karşılaştırmıştır. Çözücülerden ilki, alt-yapı tabanlı çözümün GİB uygulaması olan çoklu sınır çözücüdür. Diğer çözücü ise aşamalı çoklı sınır çözücünün GİB uygulamasıdır. Simetrik kare plak üzerinde gerçekleştirilen çözümler sonucunda, GİB'lerde aşamalı çoklu sınır çözücülerin, çoklu sınır çözücülerden çok daha iyi başarım gösterdiği gösterilmiştir. Ancak, seyrek matrislerin sadece GİB kullanılarak çözülmesinin başarımı henüz MİB'ler için geliştirilen çözücülerin başarımının çok gerisinde olduğu da Andaç (2011) tarafından gerçekleştirilen çalışmanın ortaya koyduğu önemli bir sonuçtur.

Proje Kapsamı

Proje kapsamında farklı paralel bilgisayar sistemlerinde çalışabilen, gerek mekanik gerekse ısı transferi çözümlemelerinde kullanılabilen bir sonlu elemanlar çözümleme platformu geliştirilmiştir. Platformun özel olarak tasarlanan yapısı sayesinde ana çözümleme motorunu hiç değiştirmeden dışarıdan yeni eleman, malzeme modeli, çözücü ve çözümleme algoritmalarının eklenmesi mümkündür. Platformun geliştirilmesi aşamasında gerçekleştirilen çalışmalar dört ana başlık altında incelenebilir:

1. **Sonlu Eleman ve Malzeme Kütüphanesinin Geliştirilmesi ve Seri Çözümleme Algoritmasları:** Bu aşamada platforma bir, iki ve üç boyutlu çok sayıda sonlu eleman eklenmiştir. Elemanlar genelde literatürde sıkça adı geçen, zamanın testinden geçmiş elemanlardır. Elemanların bir kısmı, doğrusal olmayan geometrik ve malzeme davranışının incelendiği çözümlemelerde kullanılacak haldedir. Ayrıca mekanik elemanların yanı sıra ısı transferi çözümlemelerinde kullanılabilecek elemanlar da kütüphaneye eklenerek, kütüphanedeki eleman çeşitliliği ve genişliği arttırılmıştır. Malzeme modeli olarak zemin ve çelik yapısal elemanların modellenmesinde kullanılan birer doğrusal olmayan model kütüphaneye eklenmiş, bu sayede doğrusal olmayan algoritmaların başarımlarının sınanması mümkün olmuştur. Bunların yanı sıra, tek bilgisayarda çalışmak üzere tasarlanmış doğrusal statik, doğrusal olmayan statik, doğrusal ve doğrusal olmayan dinamik çözümleme algoritmalarının yanı sıra ısı transferi için kararlı hal ve zaman alanında çözümleme algoritmaları da platforma eklenmiştir. Son olarak, kütüphanedeki bütün elemanların, malzeme modellerinin ve seri çözümleme algoritmalarının literatüre sıkça kullanılan ve analitik sonuçları bilinen problemlerle doğrulukları sınanmıştır.
2. **Çözümleme Platformunun Veri Yapısının Geliştirilmesi:** Yeni nesil yazılımların geliştirilmesinde önemli konular arasında, yazılım büyüdükçe farklı araştırmacıların yazılımı geliştirmekte zorlanmaları, yazılımın bir bölümünde yapılan bir değişikliğin yazılımın diğer parçalarını bozabilme riskinin var olması ve paralel hesaplama ihtiyacının getirdiği ek karmaşılıklar olarak sayılabilir. Dolayısıyla, yapı mühendisliği için genişletilebir yeni bir sonlu elemanlar platformunun geliştirilmesindeki en önemli ihtiyaç, platformun veri yapısının anlaşılabilir, kullanılabilir ve modüler bir yapıda olup, yeni parçaların eklenmesine olanak sağlamasıdır. Bu amaçla, proje çerçevesinde Panthalassa isimli bir yazılım geliştirilmiş, yazılımın nesne yönelimli veri yapısı özel olarak tasarlanmış ve genel bir çözümleme motoru geliştirilmiştir. Bu çözümleme motoruna "plug-in" teknolojisi kullanılarak yeni parçaların eklenmesi sağlanmıştır. Günümüzdeki çeşitli ticari yazılımlardan farklı olarak, çözümleme moturuna eleman, malzeme modeli gibi parçaların yanı sıra, çözümleme algoritması, bölümleme algoritması ve hatta farklı veri dosyalarını okuyup platform yapısına dönüştüren çeviriciler gibi farklı parçaların, çözümleme motorunu değiştirmeden, motorun kaynak koduna ihtiyaç duymadan eklenmesi sağlanmıştır.
3. **Paylaşık ve Dağıtık Sistemler için Paralel Çözümleme Algoritmalarının Geliştirilmesi:** Projenin belkemiğini oluşturan bu bölümde, yapı mekaniği çözümlemelerinde sıkça kullanılan algoritmaların paylaşık ve dağıtık sistemler için özelleştirilmesi ve büyük boyutlu modellerin çözümlenebilmesinin sağlanması üzerinde çalışmalar yapılmıştır. İlk olarak büyük doğrusal denklem sistemlerinin birden fazla durum için çözülmesini gerektirecek çözümleme algoritmaları üzerine çalışılmış, doğrusal statik

çözümleme yönteminin, hem paralel toptan, hem de alt-yapı tabanlı çözücülerle uygulamaları yapılmıştır. Her iki paralel çözümleme algoritmasının başarımları en iyi hale getirildikten sonra, farklı hesaplama platformlarında, büyük boyutlu yapısal modellerin çözümlenmesiyle sınanmış ve karşılaştırılmıştır. Benzer şekilde, hem toptan hem de alt-yapı tabanlı çözücü kullanılarak, zaman alanında örtük çözümleme algoritmaları geliştirilmiş ve çeşitli denektaşı problemlerde her iki yöntemin de başarımı sınanmıştır. Doğrusal olmayan çözümleme için toptan çözücü kullanan bir paralel algoritma geliştirilmiş ve algoritmanın başarımı çeşitli denektaşı problemlerle sınanmıştır. Doğrusal ve doğrusal olmayan dinamik çözümlemelerin gerçekleştirilmesi için ise paralel belirtik algoritmaların üzerinde durulmuştur. Belirtik algoritmaların paralel verimliliğinin yüksek olması, sistem matrislerinin oluşturulmasına gerek duyulmadığı için büyük boyutlu modellerin çözümüne olanak vermesi nedeniyle bu algoritmalar üzerinde durulmuş, dağıtık bellekli, paylaşımlı bellekli ve hem dağıtık hem de paylaşımlı bellekli sistemlerde çalışabilen, doğrusal ve doğrusal olmayan problemlerin zaman alanında çözümlenmesine olanak veren bir paralel algoritma geliştirilmiştir. Algoritmanın başarımı çeşitli boyutlardaki yapısal modellerle sınanmış ve milyon mertebesinde üç boyutlu sonlu elemanlara sahip modeller çözümlenebilmiştir.

4. GA-GİB'ler için Çözümleme Algoritmalarının Geliştirilmesi: Genel amaçlı grafik işlem birimleri, GA-GİB, çok sayıda düşük kapasiteli işlemci içermesi sebebiyle son yıllarda bilimsel hesaplama için kullanılabilecek bir seçenek haline gelmiştir. Proje kapsamında ise GA-GİB'ler için çeşitli çözücüler geliştirilmiş ve bu çözücülerin yapısal mekanik problemlerinde kullanılmasıyla ilgili çalışmalar yapılmıştır. Sonlu eleman yöntemi kullanıldığında elde edilen matrislerin seyrek matris olması nedeniyle ilk olarak seyrek matris çözücüler üzerine yoğunlaşılmış ve çoklu-sınır ve aşamalı çoklu sınır yöntemi kullanan iki tip seyrek matris çözücüzü geliştirilmiştir. Bu çözücülerin başarımları farklı GİB'lerde sınanmıştır. Çoklu sınır ve aşamalı çoklu sınır yöntemlerinin çözüm mantığı, seyrek matrislerden küçük boyutlu yoğun matris oluşturmak üzerine kuruludur. Dolayısıyla yoğun matris işlemlerindeki hızlanmalar direk olarak seyrek matris çözümünün başarımını etkilemektedir. Bu sebepten dolayı yüksek başarımla çalışan yoğun matris çözücüsünün geliştirilmesi üzerine de çalışılmıştır. Yapı mekaniği problemlerinde hızlanmaya en ihtiyaç duyulan alanlardan bir tanesi de zaman alanında doğrusal olmayan dinamik çözümlemedir. Proje kapsamında GA-GİB'ler için belirtik algoritma kullanılarak bir doğrusal olmayan dinamik algoritma geliştirilmiş ve algoritmanın başarımı hem farklı GİB'lerde sınanmış hem de elde edilen sonuçlar paralel MİB çözümleriyle karşılaştırılmıştır.

2. Bölüm

Sonlu Eleman ve Malzeme Modeli Kütüphanelerinin Geliştirilmesi, Seri Çözümleme Algoritmaları

Giriş

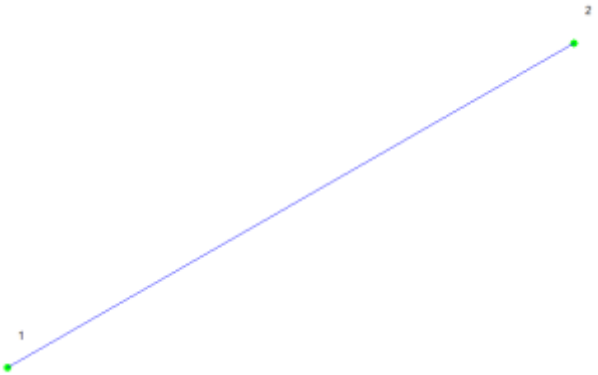
Bu bölümde platforma eklenen bir, iki ve üç boyutlu sonlu elemanlar hakkında detaylı bilgiler verilmiştir. Elemanlar genelde literatürde sıkça adı geçen, zamanın testinden geçmiş elemanlardır. Elemanların bir kısmı, doğrusal olmayan geometrik ve malzeme davranışının incelendiği çözümlemelerde kullanılacak haldedir. Ayrıca mekanik elemanların yanı sıra ısı transferi çözümlemelerinde kullanılabilecek elemanlar da kütüphaneye eklenerek, kütüphanedeki eleman çeşitliliği ve genişliği arttırılmıştır. Malzeme modeli olarak zemin ve çelik yapısal elemanların modellenmesinde kullanılan birer doğrusal olmayan model kütüphaneye eklenmiş, bu sayede doğrusal olmayan algoritmaların başarımlarının sınanması mümkün olmuştur. Bunların yanı sıra, tek bilgisayar da çalışmak üzere tasarlanmış doğrusal statik, doğrusal olmayan statik, doğrusal ve doğrusal olmayan dinamik çözümleme algoritmalarının yanı sıra ısı transferi için kararlı hal ve zaman alanında çözümleme algoritmaları da platforma eklenmiştir. Son olarak, kütüphanedeki bütün elemanların, malzeme modellerinin ve seri çözümleme algoritmalarının literatüre sıkça kullanılan ve analitik sonuçları bilinen problemlerle doğrulukları sınanmıştır.

Sonlu Elemanlar Kütüphanesinin Geliştirilmesi

Sonlu elemanlar kütüphanesi iki ve üç boyutlu elemanlarla zenginleştirilmiştir. Elemanlar mekanik ve ısı elemanları olarak iki ana tipe ayrılmıştır. Mekanik elemanlar katı mekaniği problemleri için geliştirilmiş olup her bir elemanın doğrusal direngenlik ve kütle matrisleri hesabı, çeşitli tip yüklemeler için eşdeğer yük hesabı, eleman iç kuvvetleri ve gerilmeleri hesabı eklenmiştir. İki boyutlu membran ve üç boyutlu tüm elemanlarda malzemedeki doğrusal olmayan davranışın çözümlenmesine olanak veren hesap algoritmaları da bulunmaktadır. Aynı şekilde ısı transferi özelliklerine sahip elemanların her birine doğrusal kondüksiyon, ısı kapasitesi ve çeşitli yükleme tipleri (konveksiyon, yüzey akımı, radyasyon ve ısı üretimi) hesabı eklenmiştir. Bu elemanların doğrusal ve doğrusal olmayan çeşitli problemlerdeki başarımları takip eden bölümlerde detaylıca incelenmiştir. Sonlu elemanlar kütüphanesinde bulunan elemanlarla ilgili genel bilgiler takip eden bölümlerde verilmiştir.

Makas Elemanı

Eleman kütüphanesi bünyesinde tek boyutlu makas elemanı bulundurmaktadır (Şekil 2-1).



Şekil 2-1 Makas Elemanı

Bu elemanın analitik formülasyonunun bulunmasından dolayı bu eleman için sonlu elemanlar yaklaşımına gerek duyulmamıştır. Makas elemanının analitik formülasyonu denklem 2-1'de gösterilmiştir.

$$K' = \frac{EA}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (2\text{-}1)$$

Denklem 2-1'de E elastik modülünü, A kesit alanını, L elemanın boyunu ve K elemanın direngenlik matrisini temsil etmektedir. Tek boyutlu elemanı üç boyutlu sistemde kullanabilmek amacı ile tahvil(transformation) matrisinden yararlanılmıştır (Denklem 2-2).

$$R = \begin{bmatrix} l & m & n & 0 & 0 & 0 \\ 0 & 0 & 0 & l & m & n \end{bmatrix} \quad (2\text{-}2)$$

Denklem D2-2'de gösterilen l, m ve n değerleri denklem 2-3'te gösterilmiştir.

$$l = \frac{x_j - x_i}{L}$$

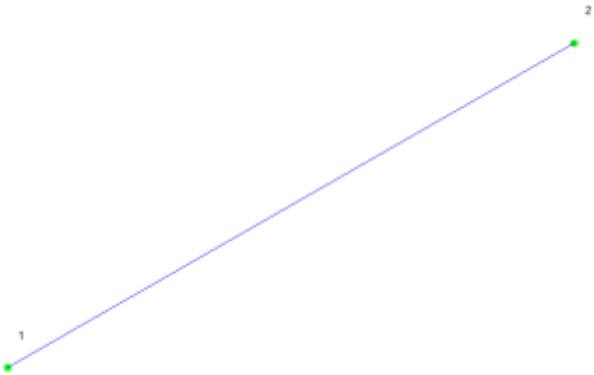
$$m = \frac{y_j - y_i}{L}$$

$$n = \frac{z_j - z_i}{L} \quad (2\text{-}3)$$

Böylelikle makas elemanı üç eksende deformasyon serbestliğine sahip hale getirilmiştir. Bu eleman yalnızca mekanik problemler için geliştirilmiş olup ısı transferi analizi için gerekli formülasyona sahip değildir.

Kiriş Elemanı

3 boyutlu bu eleman sonlu elemanlar yaklaşımı yerine analitik formülasyonu kullanılarak geliştirilmiştir (Şekil 2-2).



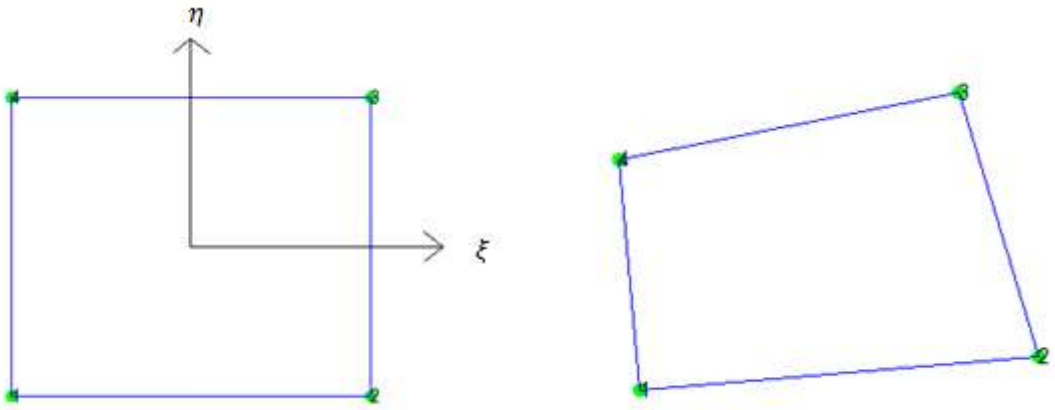
Şekil 2-2 Kiriş Elemanı

Bu elemanın geliştirilmesinde Timoshenko'nun formülasyonu kullanılmıştır. Bu formülasyonun tercih edilme sebebi kesme deformasyonlarını da içermesidir. Ayrıca Euler kiriş elemanı performansına ihtiyaç duyulduğunda elemanın direngenlik matrisi hesabına katılan kesme parametresini sıfıra eşitleyerek kullanmak mümkündür.

Bu eleman iki düğüm noktasına sahip olup her bir düğüm noktasında üç eksende deformasyon ve dönme serbestliğine sahiptir. Bu eleman yalnızca mekanik problemler için geliştirilmiş olup ısı transferi analizi için uygun formülasyona sahip değildir.

Dörtgen Kabuk Elemanı (ShellQ)

Mekanik problemler için geliştirilmiş olan iki boyutlu dörtgen kabuk elemanı plak ve membran elemanlarının birleştirilmesiyle oluşturulmuştur. Eleman iki farklı plak ve üç farklı membran elemanını içermekte olup, çözümleme sırasında kullanıcı istediği plak-membran eleman kombinasyonunu kullanabilmektedir. Plak elemanlarının ince ve kalın olmak üzere iki tipi mevcuttur. İnce kabuk elemanının plak kısmı Batoz ve Tahar'ın (1980) formülasyonları kullanılarak, kalın kabuk elamanının plak kısmı ise Ibrahimbegovic'in (1993) formülasyonu kullanılarak oluşturulmuştur. Membran elemanlar ise doğrusal (Cook 2001), ikinci derece (Cook 2001) ve doğrusal elemana dönme serbestliklerinin (drilling dof) eklenmesiyle oluşturulmuş üç farklı tiptedir. Bu elemanların dönme serbestliğine sahip membran kısmı Ibrahimbegovic et al. (1990) formülasyonları kullanılarak geliştirilmiştir. Dörtgen kabuk elemanının gerçek ve doğal koordinat sistemlerindeki tanımı Şekil 2-3' te gösterilmiştir.

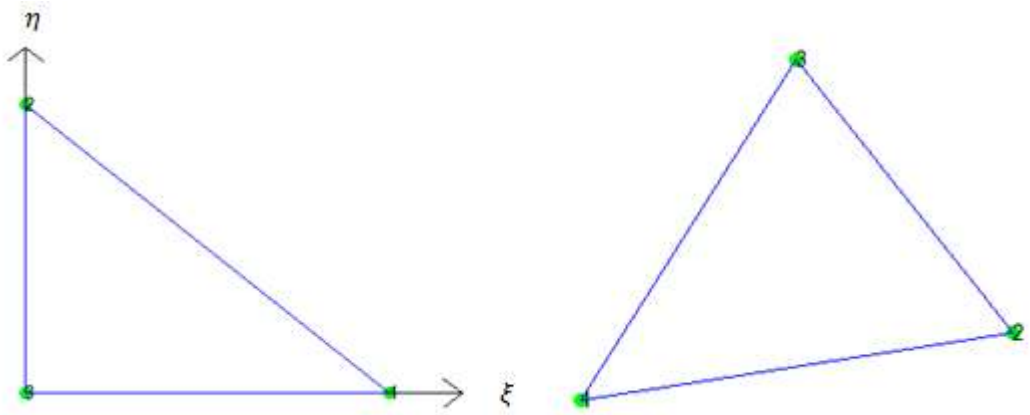


Şekil 2-3 Dörtgen Kabuk Elemanı

Dört düğüm noktasına sahip bu kabuk elemanının doğrusal ve ikinci derece membran özelliğine sahip tiplerinin her bir düğüm noktasında üç eksende deformasyon ve üç eksende dönme serbestliği tanımlanmıştır. Bu tip elemanlar dışarıdan altı serbestlik derecesine sahip görünseler de eleman yüzeyine dik eksendeki dönme rijitliği çözümleme sırasında sayısal kararlılığı bozmaması amacıyla çok küçük bir sayısal değere sahip olacak şekilde tanımlanmaktadır. Gerçek anlamda altı serbestlik derecesine sahip kabuk elemanı doğrusal elemana dönme serbestliklerinin eklenmesiyle oluşturulmuş membran elemana sahip olanıdır. Dönme serbestliğine sahip membran elemanın formülasyonunda düğüm noktalarındaki dönmeler, eleman kenarlarının orta noktalarındaki deplasmanlar kullanılarak hesaplanmakta, daha sonra eleman rijitlik matrisindeki çeşitli sıfır enerjili modlar da düzeltilmektedir. Dörtgen kabuk elemanının kapsadığı iki tip plak ve üç tip membran elemanı tek başlarına da kullanılabilmektedir.

Üçgen Kabuk Elemanı (ShellT):

Mekanik problemler için geliştirilen iki boyutlu üçgen kabuk elemanı dörtgen kabuk elemanının eşleneği olarak geliştirilmiştir. İki farklı plak ve iki farklı membran elemanlarını kullanan bu elemanın plak elemanları ince ve kalın olmak üzere iki tiptir. İnce kabuk elemanının plak kısmı Batoz ve Tahar'ın geliştirdiği DKT isimli formülasyonu kullanılarak (Batoz et al. 1982), kalın kabuk elemanının plak kısmı ise Ibrahimbegovic'in dörtgen elemanlar için geliştirdiği formülasyonun üçgen elemanlara uygulanmasıyla geliştirilmiştir (1993). Elemanın membran kısmında ise doğrusal (Cook 2001) ve Allman (1984) tarafından geliştirilen ve düğüm noktalarında dönme serbestlik derecesine sahip membran formülasyonları kullanılmıştır. Üçgen kabuk elemanının doğal ve gerçek koordinat sistemlerindeki tanımı Şekil 2-4'te gösterilmiştir.

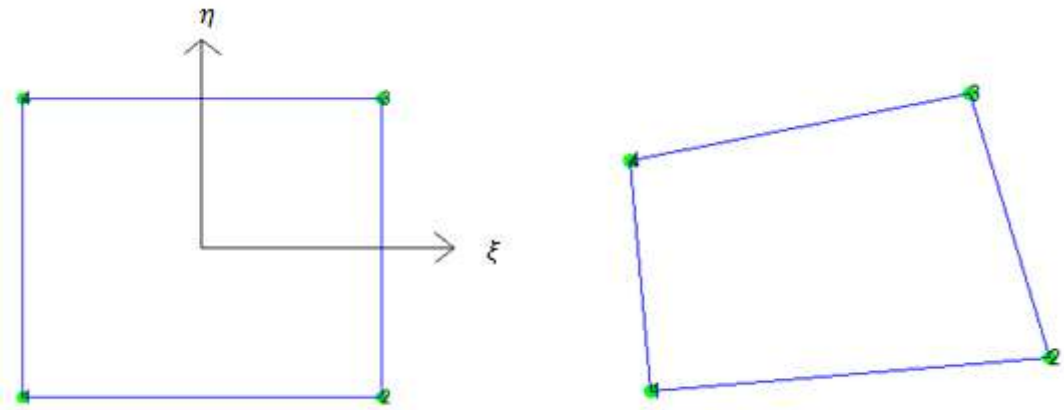


Şekil 2-4 Üçgen Kabuk Elemanı

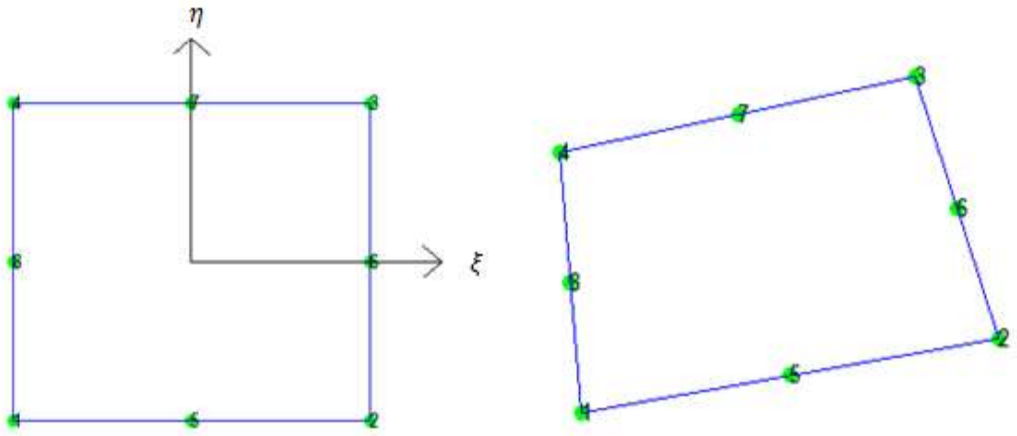
Bu eleman üç düğüm noktasına ve her bir düğüm noktasında üç eksende deformasyon ve dönme serbestliği olmak üzere on sekiz serbestlik derecesine sahiptir. Dörtgen kabuk elemanında olduğu gibi doğrusal membran elemanı kullanıldığında eleman yüzeyine dik yöndeki dönme rijitliği küçük bir değer olarak tanımlanmaktadır. Gerçekçi dönme rijitliğine ihtiyaç duyulduğu çözümlemelerde Allman membran elemanı kullanılmalıdır. Üçgen kabuk elemanının kapsadığı iki tip plak ve iki tip membran elemanı tek başlarına da kullanılabilmektedir.

Dörtgen Isı Elemanları (RecM4 & RecM8)

İki boyutlu bu elemanlar dörtgen kabuk elemanlarıyla beraber veya ayrı kullanılarak ısı iletimi çözümlemelerinde kullanılmak amacıyla geliştirilmiştir. Doğrusal ve ikinci derece şekil fonksiyonları [Cook] ile oluşturulmuş dört ve sekiz düğüm noktasına sahip bu elemanlar sırasıyla Şekil 2-5 ve Şekil 2-6'da gösterilmiştir.



Şekil 2-5 Doğrusal Dörtgen Isı Elemanı



Şekil 2-6 İkinci Derece Dörtgen Isı Elemanı

Bu elemanların her bir düğüm noktasında tek sıcaklık değeri bulunmaktadır. Bu elemanların ısı kondüksiyonu, ısı konveksiyonu, ısı üretimi ve ısı kapasitesi matrislerinin oluşturulmasında kullanılan doğrusal ve ikinci derece şekil fonksiyonları sırasıyla Denklem 2-4 ve Denklem 2-5'te gösterilmiştir.

$$N4 := 0.25 \begin{bmatrix} (1-\xi)\,(1-\eta) \\ (1+\xi)\,(1-\eta) \\ (1+\xi)\,(1+\eta) \\ (1-\xi)\,(1+\eta) \end{bmatrix} \tag{2-4}$$

$$N8 := 0.25 \begin{bmatrix} (1-\xi)(1-\eta) - [(1-\xi)(1-\eta^2) + (1-\xi^2)(1-\eta)] \\ (1+\xi)(1-\eta) - [(1-\xi^2)(1-\eta) + (1+\xi)(1-\eta^2)] \\ (1+\xi)(1+\eta) - [(1+\xi)(1-\eta^2) + (1-\xi^2)(1+\eta)] \\ (1-\xi)(1+\eta) - [(1-\xi^2)(1+\eta) + (1-\xi)(1-\eta^2)] \\ (1-\xi^2)(1-\eta) \\ (1+\xi)(1-\eta^2) \\ (1-\xi^2)(1+\eta) \\ (1-\xi)(1-\eta^2) \end{bmatrix} \tag{2-5}$$

$-1 < \xi < 1$ ve $-1 < \eta < 1$

Bu elemanın yüzey akımı ve ısı konveksiyonun hesaplanırken kullanılan yüzey integralinde doğrusal ve ikinci derece doğrunun şekil fonksiyonları kullanılmıştır ve sırasıyla Denklem 2-6 ve Denklem 2-7'de gösterilmiştir.

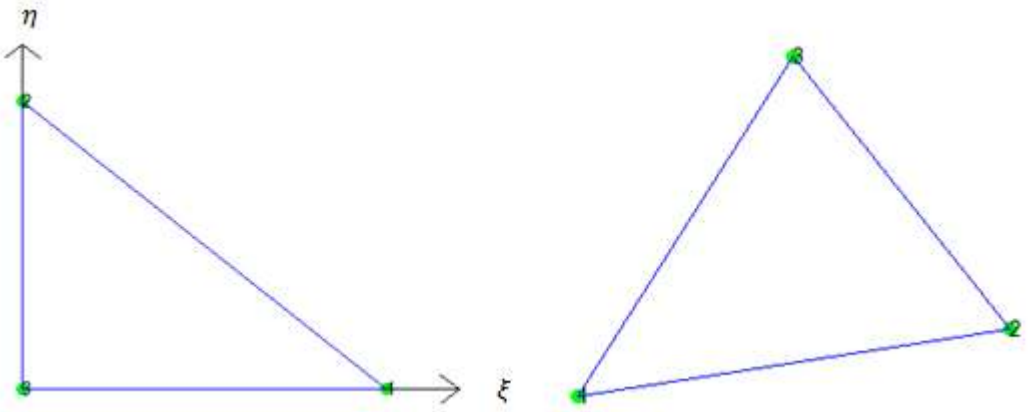
$$N_{Yüzey} \coloneqq \begin{bmatrix} \frac{1-\xi}{2} \\ \frac{1+\xi}{2} \end{bmatrix} \quad (2\text{-}6)$$

$$N_{Yüzey} \coloneqq \begin{bmatrix} \frac{\xi(\xi-1)}{2} \\ \frac{\xi(\xi+1)}{2} \\ 1-\xi^2 \end{bmatrix} \quad (2\text{-}7)$$

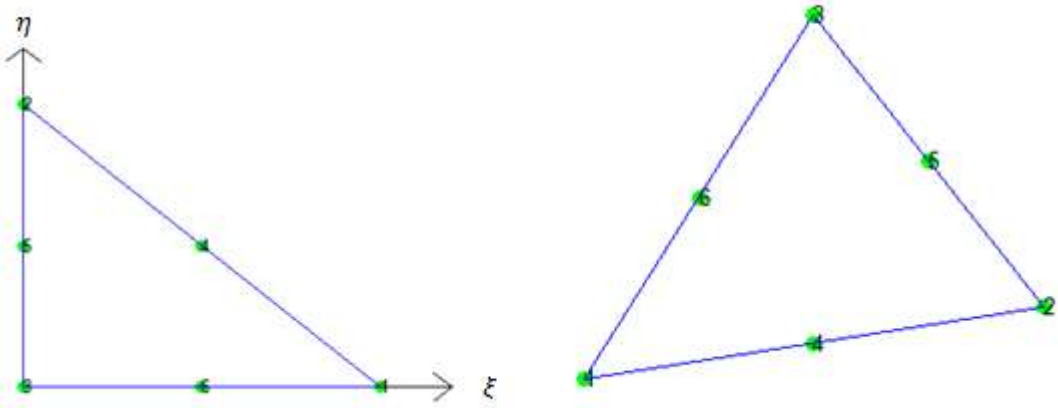
$-1 < \xi < 1$

Üçgen Isı Elemanları (TriM3 & TriM6)

Üçgen kabuk elemanlarıyla beraber veya tek başına kullanılabilen iki boyutlu bu eleman ısı transferi çözümlemelerinde kullanılmak amacıyla geliştirilmiştir. Üç ve altı düğüm noktasına sahip iki tip üçgen eleman sırasıyla Şekil 2-7 ve Şekil 2-8'de gösterilmiştir.



Şekil 2-7 Doğrusal Üçgen Isı Elemanı



Şekil 2-8 İkinci Derece Üçgen Isı Elemanı

Isı elemanlarının her bir düğüm noktasında tek bir sıcaklık değeri tanımlanmıştır. Elemanların ısı kondüksiyonu, ısı konveksiyonu, ısı üretimi ve ısı kapasitesi matrislerinin oluşturulması aşamasında kullanılan üç düğüm noktalı eleman için doğrusal ve altı düğüm noktalı eleman için ikinci derece şekil fonksiyonları (Cook 2001) sırasıyla Denklem 2-8 ve Denklem 2-9'da gösterilmiştir.

$$N3 := \begin{bmatrix} \xi \\ \eta \\ 1-\xi-\eta \end{bmatrix} \tag{2-8}$$

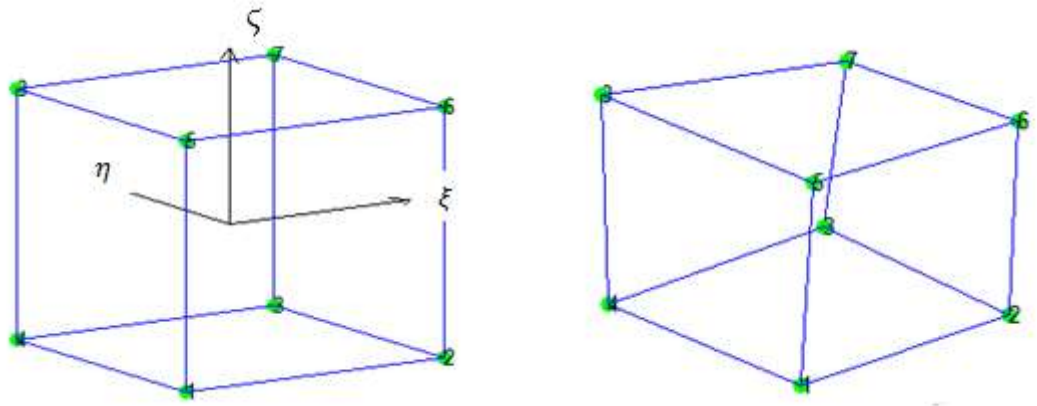
$$N6 := \begin{bmatrix} \xi(2\xi-1) \\ \eta(2\eta-1) \\ (1-\xi-\eta)(2(1-\xi-\eta)-1) \\ 4\xi\eta \\ 4\eta(1-\xi-\eta) \\ 4\xi(1-\xi-\eta) \end{bmatrix} \tag{2-9}$$

$0 < \xi < 1$ ve $0 < \eta < 1$

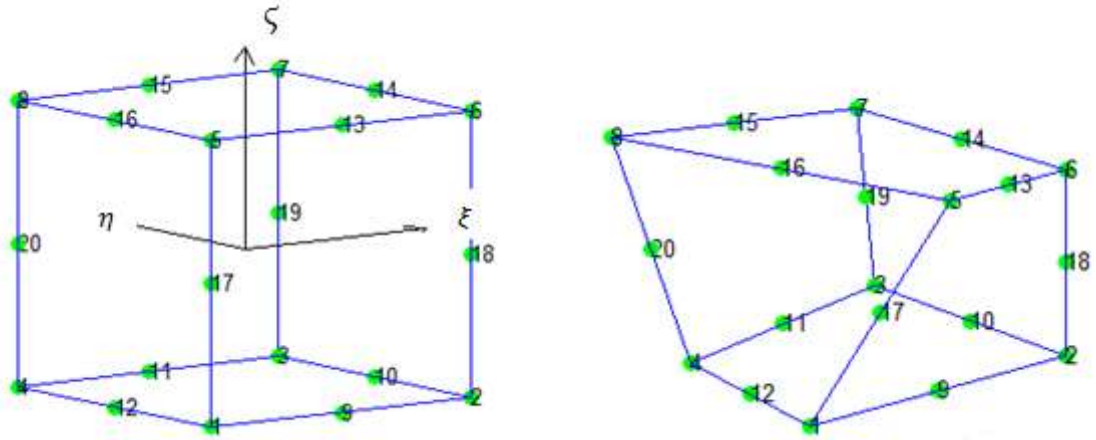
Bu elemanın yüzey akımı yük vektörü ve ısı konveksiyonu direngenlik matrisi ve yük vektörü hesaplamalarında yüzey integrali alınırken dörtgen ısı elemanlarında olduğu gibi doğrusal ve ikinci derece şekil fonksiyonları kullanılmıştır.

Dörtgen Prizma Elemanı (Brick8 & Brick20)

Üç boyutlu bu elemanın doğrusal, ikinci derece ve doğrusal şekil fonksiyonlarına ikinci derece fonksiyonlar eklenerek elde edilmiş üç farklı tipi bulunmaktadır. Doğrusal elemanın kesme ve eğilme yükleme durumlarında rijit davranmasından dolayı bu elemanın formülasyonuna ikinci derece fonksiyonlar eklenerek geliştirilmiş Brick8 (Cook 2001) elemanı oluşturulmuştur. Bu eleman geometrik olarak Brick8 eleman ile aynı olup davranışı Brick8 elemanının davranışından biraz daha esnek hale getirilmiştir. Doğrusal ve ikinci derece şekil fonksiyonlarına sahip elemanlar sırasıyla Şekil 2-9 ve Şekil 2-10'da gösterilmiştir. Bu elemanın doğrusal ve ikinci derece şekil fonksiyonları kullanılarak oluşturulmuş tipleri hem mekanik hem de ısı transferi problemi özelliklerine sahip iken doğrusal şekil fonksiyonlarına ikinci derece fonksiyonların eklenmesiyle oluşturulan tipi yalnızca mekanik özelliklere sahiptir.



Şekil 2-9 Doğrusal Dörtgen Prizma Elemanı



Şekil 2-10 İkinci Derece Dörtgen Prizma Elemanı

Doğrusal ve doğrusal şekil fonksiyonlarına ikinci derece fonksiyonların eklenmesiyle oluşturulan eleman tiplerinde sekiz, ikinci derece şekil fonksiyonları ile oluşturulan eleman da ise yirmi düğüm noktası bulunmaktadır. Bu elemanın her tipinde her bir düğüm noktasında mekanik olarak üç eksende serbestlik derecesine sahiptir. Ayrıca ısı transferi özelliği olan tiplerde her bir düğüm noktasında tek sıcaklık değeri mevcuttur. Eleman formülasyonunda kullanılan doğrusal ve ikinci derece şekil fonksiyonları sırasıyla Denklem 2-10 ve Denklem 2-11'de gösterilmiştir.

$$N8 \coloneqq 0.125 \begin{bmatrix} (1-\xi)\,(1-\eta)\,(1-\zeta) \\ (1+\xi)\,(1-\eta)\,(1-\zeta) \\ (1+\xi)\,(1+\eta)\,(1-\zeta) \\ (1-\xi)\,(1+\eta)\,(1-\zeta) \\ (1-\xi)\,(1-\eta)\,(1+\zeta) \\ (1+\xi)\,(1-\eta)\,(1+\zeta) \\ (1+\xi)\,(1+\eta)\,(1+\zeta) \\ (1-\xi)\,(1+\eta)\,(1+\zeta) \end{bmatrix} \tag{2-10}$$

$$N20 := 0.125 \begin{bmatrix} (-1+\xi)\,(1-\eta)\,(1-\varsigma)\,(2+\xi+\eta+\varsigma) \\ (-1-\xi)\,(1-\eta)\,(1-\varsigma)\,(2-\xi+\eta+\varsigma) \\ (-1-\xi)\,(1+\eta)\,(1-\varsigma)\,(2-\xi-\eta+\varsigma) \\ (-1+\xi)\,(1+\eta)\,(1-\varsigma)\,(2+\xi-\eta+\varsigma) \\ (-1+\xi)\,(1-\eta)\,(1+\varsigma)\,(2+\xi+\eta-\varsigma) \\ (-1-\xi)\,(1-\eta)\,(1+\varsigma)\,(2-\xi+\eta-\varsigma) \\ (-1-\xi)\,(1+\eta)\,(1+\varsigma)\,(2-\xi-\eta-\varsigma) \\ (-1+\xi)\,(1+\eta)\,(1+\varsigma)\,(2+\xi-\eta-\varsigma) \\ 2\,(1-\xi)\,(1+\xi)\,(1-\eta)\,(1-\varsigma) \\ 2\,(1+\xi)\,(1-\eta)\,(1+\eta)\,(1-\varsigma) \\ 2\,(1-\xi)\,(1+\xi)\,(1+\eta)\,(1-\varsigma) \\ 2\,(1-\xi)\,(1-\eta)\,(1+\eta)\,(1-\varsigma) \\ 2\,(1-\xi)\,(1+\xi)\,(1-\eta)\,(1+\varsigma) \\ 2\,(1+\xi)\,(1-\eta)\,(1+\eta)\,(1+\varsigma) \\ 2\,(1-\xi)\,(1+\xi)\,(1+\eta)\,(1+\varsigma) \\ 2\,(1-\xi)\,(1-\eta)\,(1+\eta)\,(1+\varsigma) \\ 2\,(1-\xi)\,(1-\eta)\,(1-\varsigma)\,(1+\varsigma) \\ 2\,(1+\xi)\,(1-\eta)\,(1-\varsigma)\,(1+\varsigma) \\ 2\,(1+\xi)\,(1+\eta)\,(1-\varsigma)\,(1+\varsigma) \\ 2\,(1-\xi)\,(1+\eta)\,(1-\varsigma)\,(1+\varsigma) \end{bmatrix} \quad (2\text{-}11)$$

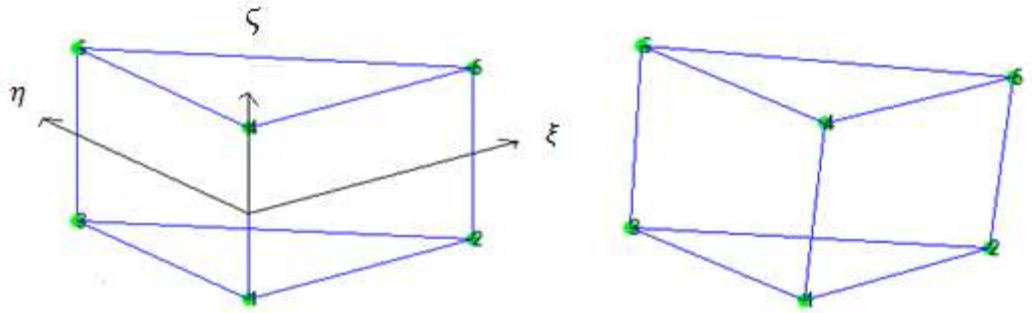
-1 < ξ < 1, -1 < η < 1 ve -1 < ς < 1

Bu elemanın gerek mekanik problemlerde yüzey yüklerinin hesaplanması gerekse ısı transferi problemlerinde ısı konveksiyonu ve yüzey akımı hesapları için gerekli olan yüzey integralinde iki boyutlu dörtgen elemanın doğrusal ve ikinci derece şekil fonksiyonları kullanılmıştır.

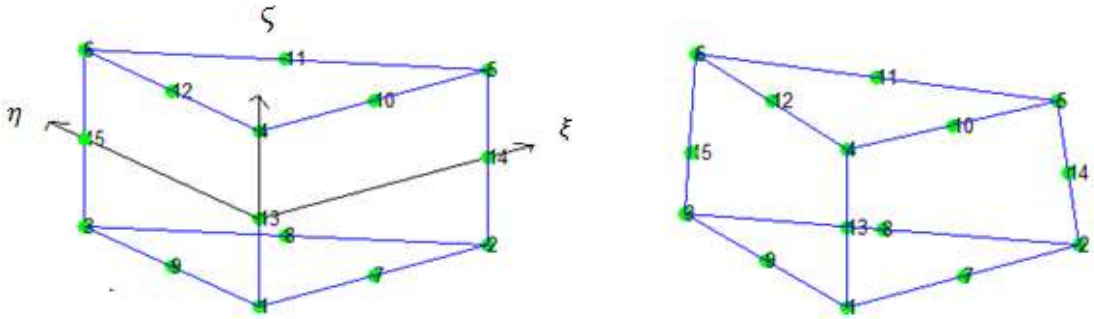
Düzgün olmayan geometrik yapılarda dörtgen prizma elemanının geometrisinin bozulması nedeniyle performansında düşüş gözlenmiş ve bu tarz yapıları modellemek için üçgen prizma ve düzgün dört yüzlü elemanlara gerek duyulmuştur.

Üçgen Prizma Elemanı (Wedge6 & Wedge15)

Üç boyutlu bu eleman doğrusal ve ikinci derece şekil fonksiyonları ile oluşturulmuş iki farklı tipe sahiptir. Doğrusal ve ikinci derece şekil fonksiyonlarına sahip eleman tipleri sırasıyla Şekil 2-11 ve Şekil 2-12'de gösterilmiştir. Bu elemanın her iki tipi de hem mekanik özelliklere hem de ısı transferi analizi özelliklerine sahiptir.



Şekil 2-11 Doğrusal Üçgen Prizma Elemanı



Şekil 2-12 İkinci Derece Üçgen Prizma Elemanı

Doğrusal şekil fonksiyonlarına sahip eleman altı, ikinci derece şekil fonksiyonlarına sahip elaman ise on beş düğüm noktasına sahiptir. Bu elemanın her iki tipinde de her bir düğüm noktasında mekanik olarak üç eksende serbestlik derecesi ve ısı transferi analizinde tek sıcaklık değeri mevcuttur. Üçgen prizma elemanının doğrusal ve ikinci derece şekil fonksiyonları denklem 2-12 ve denklem 2-13'te gösterilmiştir.

$$N6 := 0.5 \begin{bmatrix} (1-\xi-\eta)\,(1-\varsigma) \\ \xi\,(1-\varsigma) \\ \eta\,(1-\varsigma) \\ (1-\xi-\eta)\,(1+\varsigma) \\ \xi\,(1+\varsigma) \\ \eta\,(1+\varsigma) \end{bmatrix} \qquad (2\text{-}12)$$

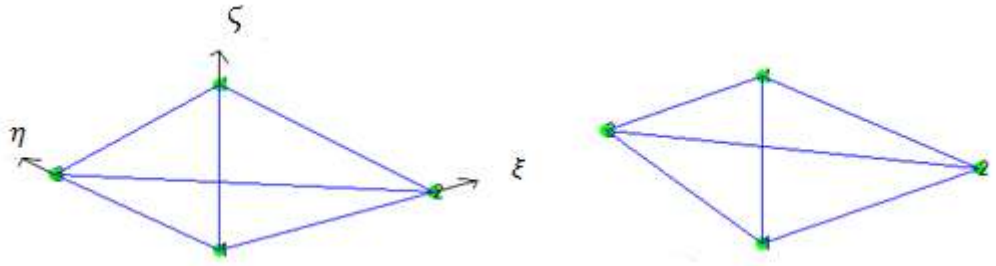
$$N15 := \begin{bmatrix} 0.5\,(-1+\xi+\eta)\,(1-\varsigma)\,(2\,\xi+2\,\eta+\varsigma) \\ 0.5\,\xi\,(1-\varsigma)\,(2\,\xi-\varsigma-2) \\ 0.5\,\eta\,(1-\varsigma)\,(2\,\eta-\varsigma-2) \\ 0.5\,(-1+\xi+\eta)\,(1+\varsigma)\,(2\,\xi+2\,\eta-\varsigma) \\ 0.5\,\xi\,(1+\varsigma)\,(2\,\xi+\varsigma-2) \\ 0.5\,\eta\,(1+\varsigma)\,(2\,\eta+\varsigma-2) \\ 2\,\xi\,(1-\xi-\eta)\,(1-\varsigma) \\ 2\,\xi\,\eta\,(1-\varsigma) \\ 2\,\eta\,(1-\xi-\eta)\,(1-\varsigma) \\ 2\,\xi\,(1-\xi-\eta)\,(1+\varsigma) \\ 2\,\xi\,\eta\,(1+\varsigma) \\ 2\,\eta\,(1-\xi-\eta)\,(1+\varsigma) \\ (1-\xi-\eta)\,(1-\varsigma^2) \\ \xi\,(1-\varsigma^2) \\ \eta\,(1-\varsigma^2) \end{bmatrix} \qquad \text{(2-13)}$$

$-1 < \xi < 1$, $-1 < \eta < 1$ ve $0 < \varsigma < 1$

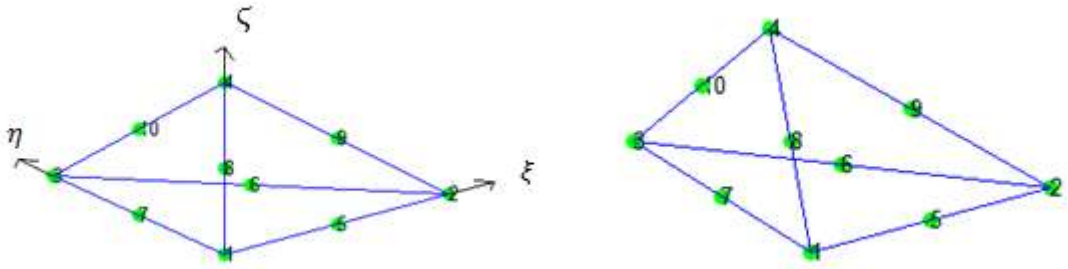
Bu elemanın kütüphanedeki diğer elemanlardan farkı iki farklı yüzey geometrisine (üçgen ve dörtgen) sahip olmasıdır. Bu nedenle mekanik problemlerde yüzey yükü, ısı transferi problemlerinde ise ısı konveksiyonu ve yüzey akımı hesabında kullanılacak yüzey integrali için üçgen ve dörtgen elemanların şekil fonksiyonlarından yararlanılmıştır. Diğer elemanlarda olduğu gibi doğrusal ve ikinci derece eleman için sırasıyla doğrusal ve ikinci derece şekil fonksiyonları ile geliştirilmiş yüzey elemanları kullanılmıştır.

Düzgün Dört Yüzlü Elemanı (Tet4 & Tet10):

Üç boyutlu bu elemanın doğrusal ve ikinci derece şekil fonksiyonlarına sahip iki ayrı tipi sırasıyla Şekil 2-13 ve Şekil 2-14'te gösterilmiştir. Her iki tip hem mekanik hem de ısı transferi analizi özelliklerine sahiptir. Bu elemanlar dörtgen ve üçgen prizma elemanlarının model geometrisinin ihtiyaçlarını karşılayamadığı eğrisel bölgelerde kullanılmak üzere geliştirilmiştir.



Şekil 2-13 Doğrusal Düzgün Dört Yüzlü Elemanı



Şekil 2-14 İkinci Derece Düzgün Dört Yüzlü Elemanı

Doğrusal şekil fonksiyonları kullanılarak oluşturulmuş elemanda dört, ikinci derece şekil fonksiyonları kullanılarak oluşturulmuş elemanda ise on düğüm noktası bulunmaktadır. Her iki tip elemanın her bir düğüm noktası mekanik olarak üç eksende serbestlik derecesine, ısı transferi analizlerinde ise tek sıcaklık değerine sahiptir. Eleman formülasyonunda kullanılan doğrusal ve ikinci derece şekil fonksiyonları [Dhont] sırasıyla Denklem 2-14 ve Denklem 2-15'te gösterilmiştir.

$$N4 := \begin{bmatrix} 1-\xi-\eta-\zeta \\ \xi \\ \eta \\ \zeta \end{bmatrix} \tag{2-14}$$

$$N10 := \begin{bmatrix} (2(1-\xi-\eta-\zeta)-1)(1-\xi-\eta-\zeta) \\ \xi(2\xi-1) \\ \eta(2\eta-1) \\ \zeta(2\zeta-1) \\ 4\xi(1-\xi-\eta-\zeta) \\ 4\xi\eta \\ 4\eta(1-\xi-\eta-\zeta) \\ 4\zeta(1-\xi-\eta-\zeta) \\ 4\xi\zeta \\ 4\eta\zeta \end{bmatrix} \tag{2-15}$$

$0 < \xi < 1$, $0 < \eta < 1$ ve $0 < \zeta < 1$

Düzgün dört yüzlü elemanının mekanik olarak yüzey yükü hesabında, ısı transferi problemlerinde ise yüzey akımı ve ısı konveksiyonu hesaplarında kullanılan yüzey integralinde doğrusal ve ikinci derece şekil fonksiyonları ile kurulmuş iki boyutlu üçgen elemanların şekil fonksiyonları kullanılmıştır.

Malzeme Modeli Kütüphanesinin Geliştirilmesi

Doğrusal, doğrusal olmayan çözümlemelerle ısı çözümlemeleri sırasında kullanılacak malzeme modellerinin tanımlarının yapıldığı malzeme modeli kütüphanesi de yazılım platformuna eklenmiştir. Farklı çözümleme tipleri ve farklı malzeme ortamları için şu aşamada kütüphaneye tek bir model eklenmiş olsa da gerek yazılımın gerekse kütüphanenin yapısı farklı malzeme modellerinin eklenerek

platformun çözümleme yeteneğinin arttırılmasına olanak vermektedir. Aşağıda kütüphanede bulunan malzeme modelleri özetlenmiştir. Eklenen malzeme modellerinin her birine mekanik analizler için gerilim-gerinim ilişkisi matrisleri ve ısı transferi analizleri için de kondüksiyon matrisi hesapları eklenmiştir. Doğrusal olmayan ve ısı transferi problemi için eklenen malzeme modellerinin doğrulukları takip eden bölümlerde detaylıca incelenmiştir.

Doğrusal Elastik Malzeme Modeli

Doğrusal elastik malzeme modeli izotropik elastik malzeme modeli formülasyonu kullanılarak geliştirilmiştir. Bu malzeme bir boyutlu, düzlem gerilim (plane stress), düzlem gerinim (plane strain) ve üç boyutlu malzeme tiplerinin formülasyonunu içermektedir.

Elastoplastik Malzeme Modeli

Malzeme kaynaklı doğrusal olmayan davranışın incelenebilmesi için malzeme modeli kütüphanesine elasto-plastik malzeme modeli eklenmiştir. Elasto-plastik malzeme modeli yalnızca Von Mises akma ölçütü ve izotropik/kinematik pekleşme kuralı kullanılarak geliştirilmiştir. Bu model yalnızca düzlem gerilim ve üç boyutlu malzeme tipleri için geliştirilmiştir.

Von Mises Akma Kriteri ve Izotropik/Kinematik Pekleşme Kuralı

Von Mises modeli üç boyutlu gerilme ve düzlem gerilme durumu için çalışmakta, doğrusal izotropik veya kinematik pekleşmenin yanı sıra doğrusal olmayan pekleşmeyi de modelleyebilmektedir. Genel Von Mises eşitliği denklem 2-16'da gösterilmiştir.

$$\phi = |\sigma - \beta_k| - \sqrt{\frac{2}{3}}(y_0 + \beta_i) \leq 0 \tag{2-16}$$

Denklem 2-16'da ϕ akma fonksiyonunu, $\underline{\sigma}$ gerilme matrisinini deviatoric kısmını, β_k kinematik pekleşme ile alakalı termodinamik yükü, y_0 malzemenin çekme testinde elde edilen akma gerilmesini ve β_i izotropik pekleşme ile alakalı termodinamik yükü temsil etmektedir. Bu denkleme göre akma fonksiyonu negatif değer aldığında malzeme elastik davranış, sıfır olduğunda ise malzeme elastoplastik davranış göstermektedir. Gerilme matrisi hacimsel ve deviatoric olmak üzere iki kısımdan oluşmaktadır (denklem 2-17).

$$\sigma = \sigma_{hacimsel} + \sigma_{deviatoric} \tag{2-17}$$

$$\sigma_{hacimsel} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \qquad m = \frac{\sigma_1+\sigma_2+\sigma_3}{3} \tag{2-18}$$

$$\sigma_{deviatoric} = \sigma - \sigma_{hacimsel} = J_2(\sigma) \tag{2-19}$$

Denklem 2-16'da gerilme matrisinin kendisini kullanmak yerine deviatoric kısmı kullanılmıştır. Bunun nedeni matrisin hacimsel kısmının metallerde akma davranışına katkısı olmadığı yaklaşımıdır. Bu nedenle Von Mises akma kriteri, J_2akma kriteri olarak da adlandırılır.

Eşdeğer Doğrusal Malzeme Modeli

Toprak malzemesi küçük gerilimler altında dahi doğrusal olmayan malzeme özellikleri gösterir. Eşdeğer doğrusal malzeme modeli doğrusal olmayan toprak malzeme özelliklerini basit ve gerçek problemlere kolayca uygulanabilir şekilde uygulamaya olanak sağlar. Modelin görece basitliği doğrusal olmayan toprak malzeme parametrelerinin eşdeğer doğrusal olan parametrelerle açıklanmasında yatar. Bu parametreler araziden alınan örneklerin testleriyle ya da literatürde değişik toprak türleri için hesaplanmış değerlerden belirlenebileceklerinden dolayı malzeme modelinin uygulanabilirliği yüksektir.

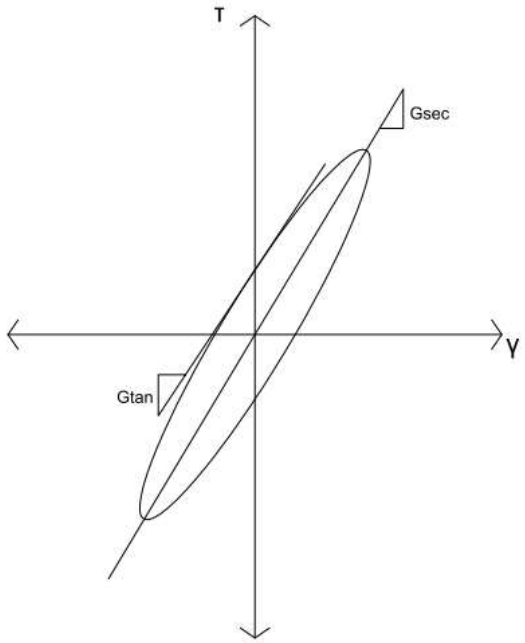
Toprak malzemesi periyodik kuvvetler (deprem) altında Şekil 2-15'deki gibi bir histeresis döngüsü şekliyle açıklanabilecek bir davranış gösterir. Şekilden görüldüğü gibi teğet kesme modülü G_{tan} her kesme gerilimi değeri için değişik bir değerdedir. Eşdeğer doğrusal malzeme modeli G_{tan} parametresini her kesme gerilimi değeri için aynı kalan sekant kesme modülüyle (G_{sec}) açıklar. G_{sec} Denklem 2-20'den bulunabilir:

$$G_{sec} = \frac{\tau_c}{\gamma_c} \tag{2-20}$$

Bu denklemde τ_c ve γ_c çevrimde kesme gerilimi ve kesme birim uzamasının aldığı en yüksek değerlere karşılık gelir. Şekil 2-15'de gösterilen şeklin içinde kalan alan (A) toprak malzemesinin yük altında harcadığı enerjiye denk gelir ve sönme oranı adı verilen λ parametresi (Denklem 2-21) ile ilişkilendirilebilir.

$$\lambda = \frac{W_D}{4\pi W_s} = \frac{1}{2\pi}\frac{A}{G_{sec}\gamma_c^2} \tag{2-21}$$

Denklem 2-21'de W_D sönümleme enerjisini W_s ise maksimum gerilim enerjisini simgeler. Eşdeğer malzeme modelinin kullandığı G_{sec} ve λ parametreleri eşdeğer malzeme parametreleri olarak bilinirler. Bu iki parametrenin bilinmesi eşdeğer malzeme modeli için yeterlidir. Eşdeğer malzeme parametreleri toprağın kesme gerilimine bağlı eğriler olarak belirlenir ve sonlu elemanlar analizinde gerekli olan malzeme matrislerinin oluşturulmasında kullanılırlar.



Şekil 2-15 Histeresis Döngüsü

Seri Çözümleme Algoritmaları

Doğrusal Statik Çözümleme

Doğrusal statik çözümleme algoritmalarında genel olarak denklem 2-22'de gösterilen sistem çözümü gerçekleştirilmektedir. Çözüm sırasında alt ve üst üçgensel ayrıştırma (LU Decomposition) yöntemi temelli algoritmalar kullanılmaktadır.

$$A\,U = F \tag{2-22}$$

A matrisi simetrik direngenlik matrisi ve F vektörü ise yük vektörü olsun. Ayrıştırma işlemi ilk olarak i'inci sıra için alt üçgensel değerlerin (L) hesaplanmasıyla başlar. Bu işlem Denklem 2-23'te gösterildiği gibi i'inci sıranın üst üçgensel değerlerinin (U) i'inci sıranın köşegen elemanına bölünmesidir. Geri kalan sıralar Denklem 2-24'te gösterildiği gibi i'inci sıranın alt üçgensel değerleri kullanılarak güncellenir.

$$L_{ij} = U_{ij} / U_{ii} \qquad i=1 \text{ to } n\text{-}1 \;\; j=i+1 \text{ to } lz \tag{2-23}$$

$$U_{jk} = A_{jk} - L_{ij} \cdot U_{ik} \qquad j=i+1 \text{ to } lz \quad k=j \text{ to } lz \tag{2-24}$$

Yukarıdaki denklemlerde n toplam sınır denklemi sayısını, *lz i+iz* ya da *n* değerlerinden en küçüğünü ve iz ise i'inci sıranın sıra genişliğini temsil eder.

A matrisinin üçgensel ayrıştırımından sonra, her bir F vektörü aşağıdaki denklem kullanılarak güncellenir:

$$F_j = F_j - L_{ij} \cdot F_i \qquad i=1 \text{ to } n \qquad j=i \text{ to } lz \tag{2-25}$$

Daha sonra, her bir F vektörü için sınır deplasmanları Denklem 2-26 kullanılarak hesaplanır:

$$F_j = F_j - U_{ji} \cdot F_i / U_{ii} \qquad i=n \text{ to } 2 \qquad j=i \text{ to } jz \tag{2-26}$$

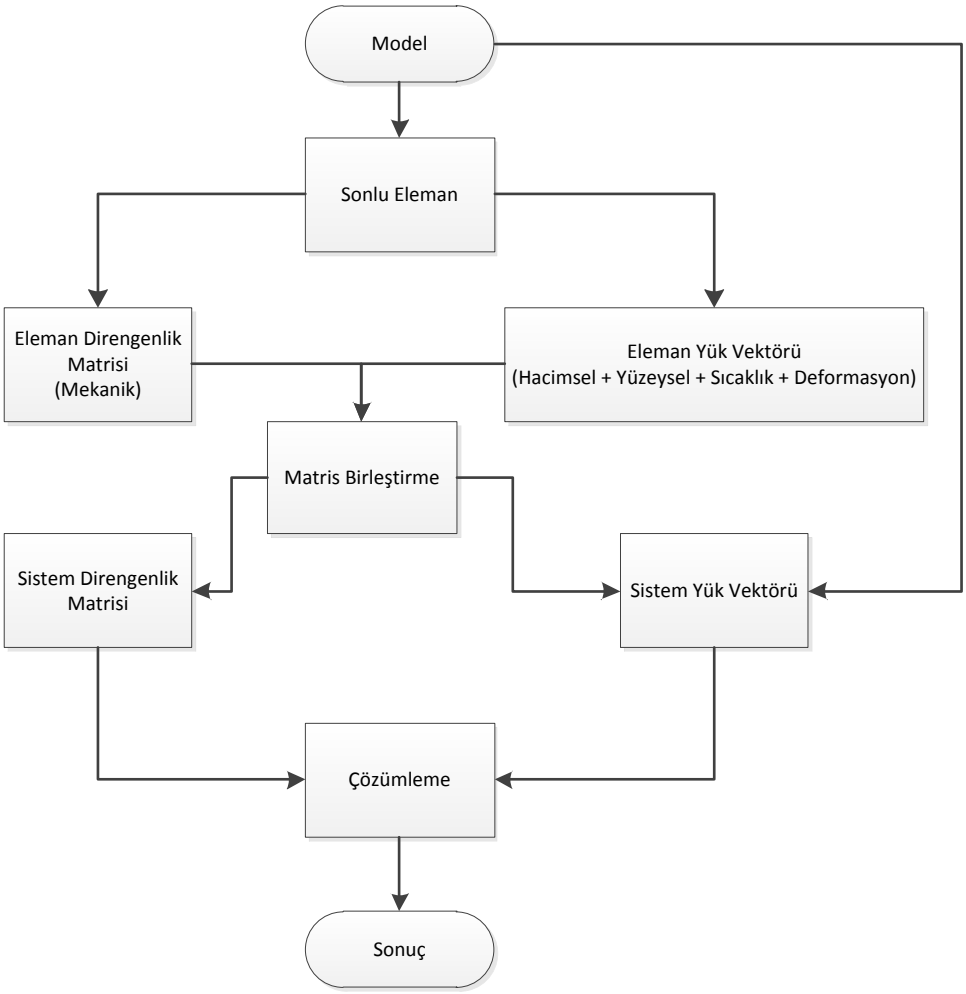
Yukarıdaki denklemde *jz* 1 veya *i-iz*'nin en küçük değere sahip olanını temsil eder.

Proje kapsaminda geliştirilen çözümleme platformu bünyesinde mekanik ve ısı transferi problemlerinin çözümlenebildiği iki ayrı doğrusal statik çözüm algoritması yapısı bulundurmaktadır. Doğrusal statik mekanik problemlerinin ve ısı transferi probleminin sistem matrislerinin eşitlikleri sırasıyla Denklem 2-27 ve Denklem 2-28'de gösterilmiştir.

$$KU = F \tag{2-27}$$

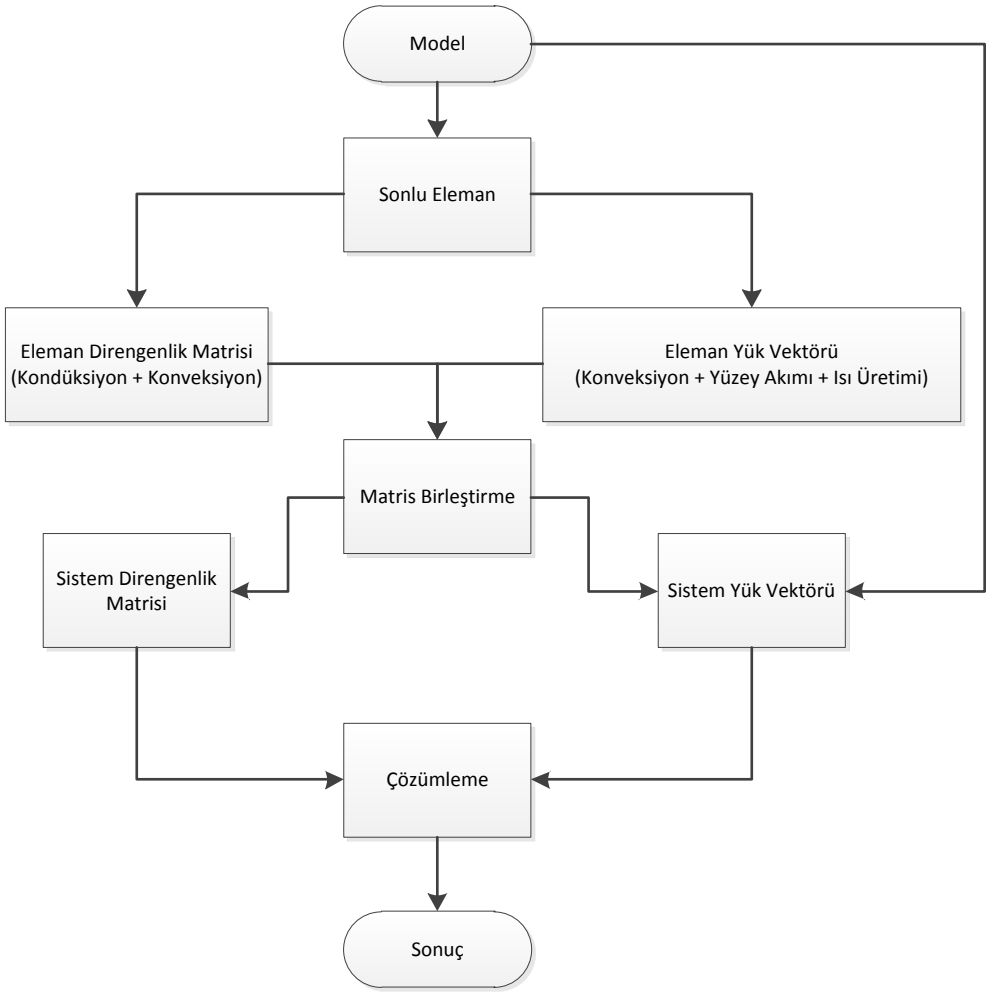
$$K_t T = F_t \tag{2-28}$$

Denklem 2-27'de *K* sistemin direngenlik matrisini, *F* yük vektörünü ve *U* vektörü düğüm noktalarının deplasman değerlerini temsil etmektedir. Denklem 2-28'de ise K_t sistemin konveksiyon ve kondüksiyon özellikleri kullanılarak hesaplanmış ısı direngenlik matrisini, F_t ısı yükü vektörünü ve *T* ise düğüm noktalarındaki sıcaklık değerlerini simgelemektedir. Denklem 2-27 ve Denklem 2-28'in çözümlenmesi için ayrı iki algoritma kullanılmasının sebepleri olarak denklemlerdeki direngenlik matrislerinin ve yük vektörlerinin oluşturulması sırasında oluşan ve yükleme tanımlarındaki farklılıklardır.



Şekil 2-16 Doğrusal Statik Çözümleme Algoritması

Şekil 2-16'da doğrusal statik çözümleme algoritmasının akış diyagramı gösterilmiştir. Doğrusal statik çözümleme algoritması model girdi dosyasından gelen bilgileri kullanarak sonlu eleman kütüphanesindeki ilgili elemandan eleman direngenlik matrisini ve eğer varsa eleman yüklerini alır ve matris birleştirme yöntemiyle sistem direngenlik matrisini ve sistem yük vektörünü oluşturur. Sonlu eleman, yük vektörü olarak hacimsel, yüzeysel, sıcaklık ve deformasyon yüklerinden istenilenleri ekleyerek tek bir eleman yük vektörü olarak çözümleme algoritmasına gönderir. Bu işlemin ardından kullanıcı tarafından girilen düğüm noktalarındaki yük değerleri de sistem yük vektörüne eklenir. Elde edilen sistem çözülerek düğüm noktalarındaki deplasman değerleri elde edilir ve bu değerler modelin sonuç dosyasına kaydedilir.



Şekil 2-17 Doğrusal Zamana Bağlı Olmayan Isı İletimi Çözümleme Algoritması

Şekil 2-17'de ise zamana bağlı olmayan ısı iletimi çözümlemesi için kullanılan algoritmanın akış diyagramı gösterilmiştir. Doğrusal zamana bağlı olmayan ısı iletimi çözümleme algoritması da doğrusal statik çözümleme algoritmasıyla aynı işlemleri içerir. Fakat ısı iletimi problemlerindeki modellerde her düğüm noktasında yalnızca bir bilinmeyen içermesinden dolayı çözümleme algoritmasının sistem direngenlik matrisini ve sistem yük vektörünü oluşturma yapısı doğrusal statik çözümleme algoritmasındaki yapıdan farklı oluşturulmuştur. Direngenlik matrisi hem modelin kondüksiyon yapısını hem de üzerine etki eden konveksiyon yükünü içerir. Aynı konveksiyon yükü yüzey akımı ve ısı üretimi yükü ile birlikte sistemin yük vektörünü oluşturur. Ancak çözümleme aşamasında doğrusal statik çözümleme algoritmasında olduğu gibi sistem direngenlik matrisinin tersi alınır ve sistem yük vektörüyle çarpılıp düğüm noktalarındaki sıcaklık değerleri elde edilir. Son olarak bu sıcaklık değerleri modelin sonuç dosyasına kaydedilir.

Doğrusal Olmayan Statik Çözümleme

Tek bilgisayarda çalışan doğrusal olmayan çözümleme algoritmaları ile hem doğrusal olmayan malzeme davranışı hem de doğrusal olmayan geometri yaklaşımını çözümlemek mümkündür. Sonlu elemanlar kütüphanesindeki tüm mekanik elemanlar doğrusal olmayan malzeme ve doğrusal olmayan geometri için ayrı ayrı direngenlik matrisi hesaplayabilmektedirler. Bu nedenle çözümle algoritması

model dosyasından analiz tipini alarak elemandan o davranışın direngenlik matrisini alır. Böylelikle her iki doğrusal olmayan yöntem için de tek tip algoritma yeterlidir. Doğrusal olmayan mekanik sistemlerin çözümünde Standart Newton Raphson ve Geliştirilmiş Newton Raphson metotları kullanılmıştır. Her iki yöntem yük kontrollü olarak da kullanılabilmektedir.

Standart Newton Raphson Yöntemi

Doğrusal olmayan sistemlerin çözümünde yaygın olarak kullanılan metotlardan birisi de Newton-Raphson yöntemidir. Her ne kadar vurgu, ters-vurgu gibi karmaşık doğrusal olmayan sistemlerde çözüm alınamasa da, çoğu sistem için hızlı bir şekilde sonuç vermektedir.

Standart Newton Raphson yönteminin doğrusal olmayan sistemlerin çözümündeki uygulanışı iki temel bölümde toplanılabilir. Bunlardan ilkinde sisteme uygulanan yük için sistemde oluşan deformasyonların çözülür, bu deformasyonlar sonucu sistemde oluşan iç kuvvet ve sistemin deformasyon sonucu değişmiş olan direngenlik matrisi hesaplanır. Bu aşama sistemin iç kuvveti ile uygulanan kuvvet arasındaki fark, belirlenen tolerans değerinden küçük oluncaya kadar tekrarlanır. Tolerans değerinin küçüklüğü sistemin gerçek çözümüne daha da yakınsamasını sağlarken çözüm zamanını da arttırmaktadır. İkinci temel bölümde ise kuvvet dengesini sağlayan yer değiştirme değerleri kaydedilip, uygulanan kuvvet belirli bir miktarda arttırılır ve tekrar birinci aşama başlatılır. Diğer bir deyişle birinci bölüm kuvvet dengesinin sağlanması için iterasyonların yapıldığı, ikinci bölüm ise kuvvet dengesinin sağlandığı her bir noktanın kaydedilip kuvvet arttırımının yapıldığı aşamalardır.

Standart Newton Raphson metodunun birinci bölümde yapılan işlemler sırasıyla Denklem 2-29, 2-30 ve 2-31'de verilmektedir.

$$\Delta u = K_{j-1}^{-1} \cdot \left(P_i - \mathrm{Pr}_{j-1} \right) \tag{2-29}$$

$$u_j = u_{j-1} + \Delta u \tag{2-30}$$

$$P_i - \mathrm{Pr}_j \approx 0 \tag{2-31}$$

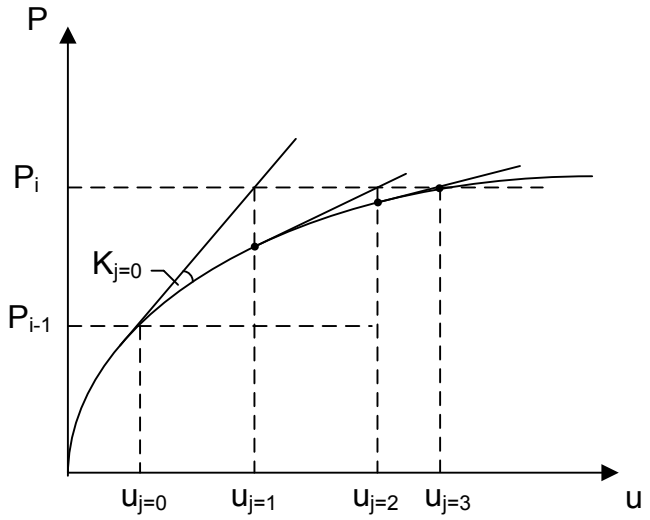
Denklem 2-32 ve 2-33'te *P* sisteme uygulanan kuvveti, *Pr* sistemde oluşan iç kuvveti, *K* sistemin direngenlik matrisini, *u* sistemde oluşan yer değiştirmeleri göstermektedir. Ayrıca *j* indisi iterasyonu, *i* indisi ise kuvvet arttırım adedini belirtmektedir. Denklem 2-31 sağlandığında birinci bölüm tamamlanmış olup Denklem 2-32 ve 2-33'te belirtilen işlemler yapılmak üzere ikinci bölüme geçilir.

$$\lambda_{i+1} = \Delta\lambda + \lambda_i \tag{2-32}$$

$$P_{i+1} = \lambda_{i+1} \cdot P_{ref} \tag{2-33}$$

Denklem 2-32 ve 2-33'te P_{ref} referans kuvvet olup sistemin yüklenileceği maksimum değerdir, $\Delta\lambda$ ise yük parametresi olup 0'dan başlar ve 1 değerine ulaştığında yük arttırılmaz ve algoritma sonlanır. Standart Newton Raphson metodunda $\Delta\lambda$ değeri önceden belirlenmiş sabit bir değerdir bu sebeple

Standart Newton Raphson metoduyla artan deformasyon ile azalan yükün olduğu sistemler çözülemez ayrıca $\Delta\lambda$ değerine göre çözüm süresi ve sistemin gerçek çözüme yakınlığı değişmektedir.



Şekil 2-18 Standart Newton Raphson Yöntemi

Standart Newton Raphson yöntemi yük kontrollü olarak da kullanılabilmektedir. Öncelikle bu yöntemde standart Newton Raphson yönteminden farklı olarak bir direngenlik parametresi S_p, ve başlangıç direngenlik parametresi olan S_{p0} tanımlanmıştır. Bu parametreler yardımıyla sistemin o anki deplasman değişiminin, yük değişimini hangi yönde etkileyeceği belirlenir. S_p denklem 2-34'te olduğu gibi hesaplanır.

$$S_P = \{\mathrm{Pref}\}_n^{\mathrm{T}} [\mathrm{K}_i]^{-1} \{\mathrm{Pref}\}_n \tag{2-34}$$

Denklem 2-34'te i=1 olduğu durumunda bulunan direngenlik parametresi başlangıç direngenlik parametresidir. Ayrıca Standart Newton Raphson da sabit olan $\Delta\lambda$ değeri her yük arttırımı ve her yineleme içerisinde değiştirilmesiyle yük kontrolü sağlanmaktadır. Denklem 2-35'te $\Delta\lambda$ değerinin her yük arttırımında nasıl hesaplanacağı belirtilmiştir.

$$\Delta\lambda_i = sign\left(\det\left(K_1\right)\right) \cdot sign\left(\det\left(K_i\right)\right) \cdot \left|\frac{S_{P0}}{S_P}\right| \cdot \Delta\lambda_0 \tag{2-35}$$

Denklem 2-36'da ise $\Delta\lambda$ değerinin her yinelemede nasıl hesaplanacağı gösterilmiştir.

$$\Delta\lambda_j = \frac{-\{\mathrm{Pref}\}_n^{\mathrm{T}} \left[\mathrm{K}_j\right]^{-1} \left\{P_j - \mathrm{Pr}_j\right\}_n}{\{\mathrm{Pref}\}_n^{\mathrm{T}} \left[\mathrm{K}_j\right]^{-1} \{\mathrm{Pref}\}_n} \tag{2-36}$$

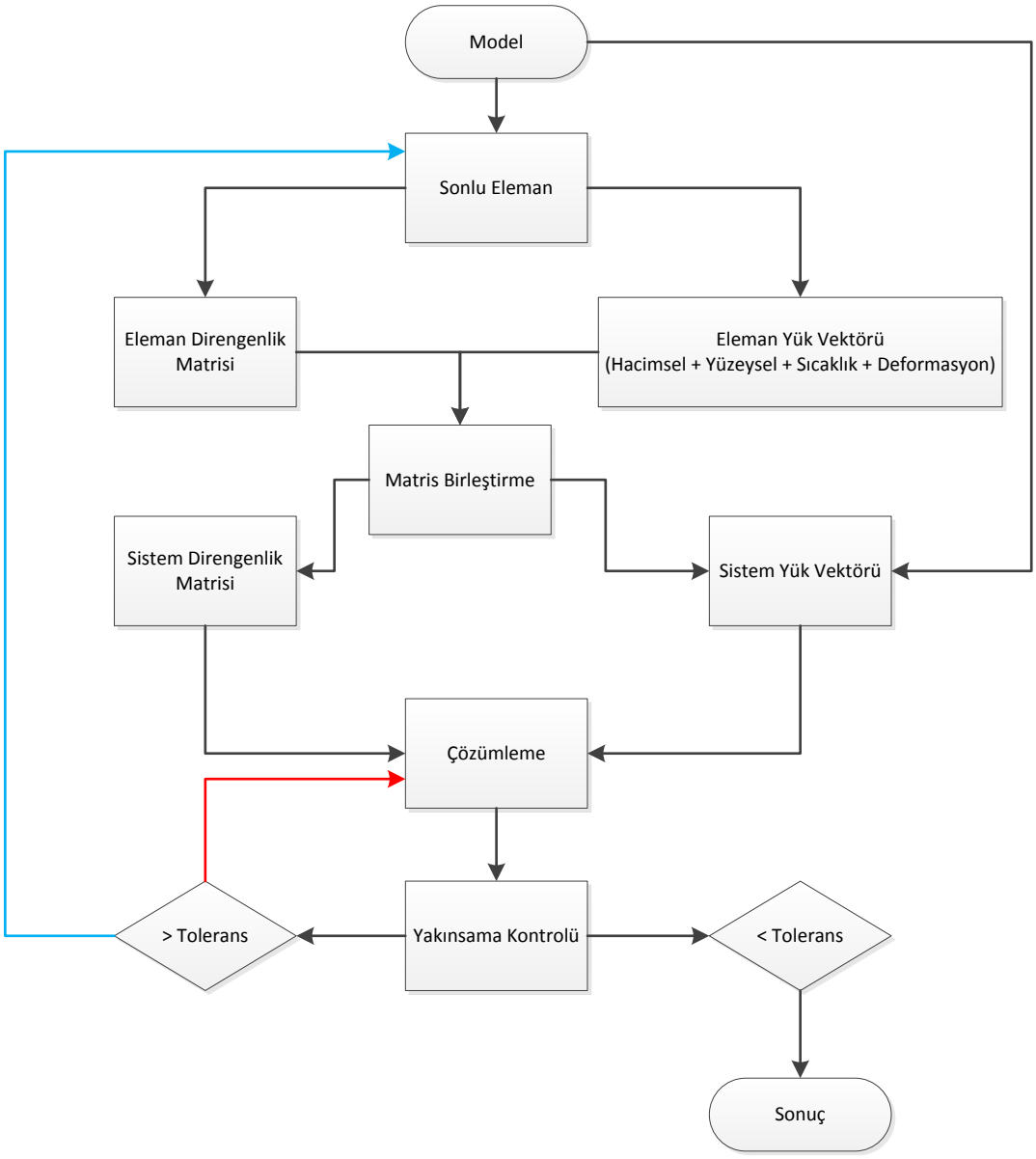
Denklem 2-35 sadece yük arttırımları sırasında kullanıldığı için, direngenlik matrisi ve $\Delta\lambda_i$ değeri "*i*" indisiyle ifade edilmiştir. Ayrıca denklem 2-35'teki $\Delta\lambda_0$ değeri önceden belirlenmiş sabit bir değerdir. Denklem 2-36 ise sadece iterasyonlarda kullanıldığı için, direngenlik matrisi ve $\Delta\lambda_j$ değeri "*j*" indisiyle ifade edilmiştir. Bunların dışında $\Delta\lambda_j$ değerinin sabit olmayıp yinelemeler sırasında da değişmesinden dolayı λ ve *P*, değerleri de her yinelemeler sırasında güncellenmelidir.

Bu yapılan değişikliklerle deplasman monotonik olarak artarken, yükün artıp azaldığı yani yük deplasman grafiğinde bazı bölgelerde pekleşme bazı bölgelerde de yumuşama olan sistemler çözülebilmektedir. Ancak yöntemin sadece yük kontrollü olmasından dolayı, vurgu ve ters-vurgunun olduğu sistemler bu yöntemle çözülememekte ve daha karmaşık bir çözüm yöntemi gerektirmektedir.

Geliştirilmiş Newton Raphson Yöntemi

Geliştirilmiş Newton Raphon metodunun Standart Newton Raphson yönteminden tek farkı her iterasyonda sistemin direngenlik matrisinin değişmemesidir. Böylelikle her çözüm adımında sistem direngenlik matrisi hesaplanmayarak işlem sayısı azalmaktadır ancak Standart Newton Raphson yöntemine göre daha fazla hesaplama adımına ihtiyaç duymaktadır. Standart Newton Raphson yönteminde olduğu gibi Geliştirilmiş Newton Raphson çözümleme algoritmasını da yük kontrollü olarak kullanmak mümkündür.

Şekil 2-19'da doğrusal olmayan statik çözümleme algoritmasının akış diyagramı gösterilmiştir. Doğrusal olamayan statik çözümleme algoritmasında takip edilen işlem sırası düğüm noktalarındaki deplasman değerlerinin elde edilmesi işlemine kadar doğrusal statik çözümleme algoritmasıyla tamamen aynıdır. Ancak elde edilen deplasman değerleri yakınsama kontrolü işlemine gönderilir. Yakınsama kontrolü işleminde modeldeki her eleman için sonlu eleman kütüphanesindeki elemana ilgili düğüm noktasındaki deplasman değerleriyle tekrardan gidilip iç kuvvet değerleriyle geri dönülür. Bu eleman iç kuvvet değerleri sistem iç kuvvet vektörüne eklenir ve elde edilen bu vektör uygulanan yük vektörü ile kıyaslanır. Eğer fark tolerans değerinden küçük ise düğüm noktalarındaki deplasman değerleri modelin sonuç dosyasına kaydedilir, eğer fark tolerans değerinden büyük ise yeniden çözümleme işlemine geçilir. Yeniden çözümleme işleminde Standart Newton Raphson metodu kullanılmış ise sonlu eleman kütüphanesindeki elemandan eleman teğet direngenlik matrisi alınır ve sistem direngenlik matrisi yeniden oluşturulur (mavi ok), Geliştirilmiş Newton Raphson metodu kullanılmış ise ilk başta oluşturulmuş olan sistem direngenlik matrisi değiştirilmeden tekrar çözümleme işleminde kullanılır (kırmızı ok). Yeniden çözümleme işleminde sistemin yeni yük vektörü olarak ugulanan yük vektörü ile sistemin iç kuvvet vektörünün farkı kullanılır ve çözümle sonunda elde edilen deplasman değerleri bir önceki deneme adımında elde edilen deplasman değerlerine eklenir. Bu yeniden çözümleme işlemi mevcut tolerans değeri yakalanana kadar devam eder.



Şekil 2-19 Doğrusal Olmayan Statik Çözümleme Algoritması

Zaman Alanında Doğrusal ve Doğrusal Olmayan Çözümleme

Çözümleme platformu statik çözümleme algoritmalarının yanında dinamik çözümleme algoritmalarını da içerir. Platformda kullanılan dinamik çözümleme algoritmaları Newmark Tümleştirme (İntegrasyon) Algoritma Ailesinin iki üyesidir: Newmark Örtük ve Newmark Belirtik Tümleştirme Algoritmaları. Her iki algoritma da temel dinamik hareket denkleminin (Denklem 2.37) tümleştirme kullanılarak çözülmesine dayanır.

$$M\ddot{u} + C\dot{u} + Ku = R_e(t) \qquad (2\text{-}37)$$

Temel dinamik hareket denlleminde M kütle matrisini, C sönümleme matrisini, K direngenlik matrisini, R^edinamik yükleme vektörünü, $\ddot{u}, \dot{u}, u$ ise sırasıyla ivme, hız ve deplasman vektörlerini temsil eder. Platformdaki eklenen Örtük Newmark Algoritması doğrusal problemleri çözebilirken, Belirtik Newmark Algoritması hem doğrusal hem de doğrusal olmayan problemlerin çözümünde kullanılabilir.

Doğrusal Örtük Çözümleme Algoritması

Denklem 2-37'deki sürekli denklemin tek bir zaman anı için yazılmış hali Denklem 2-38'de gösterilmiştir.

$$[M]\{\ddot{U}\}_n + [C]\{\dot{U}\}_n + \{R^{\text{int}}\}_n = \{R^{ext}\}_n \tag{2-38}$$

ve doğrusal sistemler için,

$$\{R^{\text{int}}\}_n = [K]\{U\}_n \tag{2-39}$$

Bu diferansiyel denklemin çözümü için 1959 yılından Newmark, Örtük Newmark Zaman tümleştirme (integrasyon) yöntemini önermiştir. Bu yöntem aşağıdaki ifadeyle gösterilebilir [1];

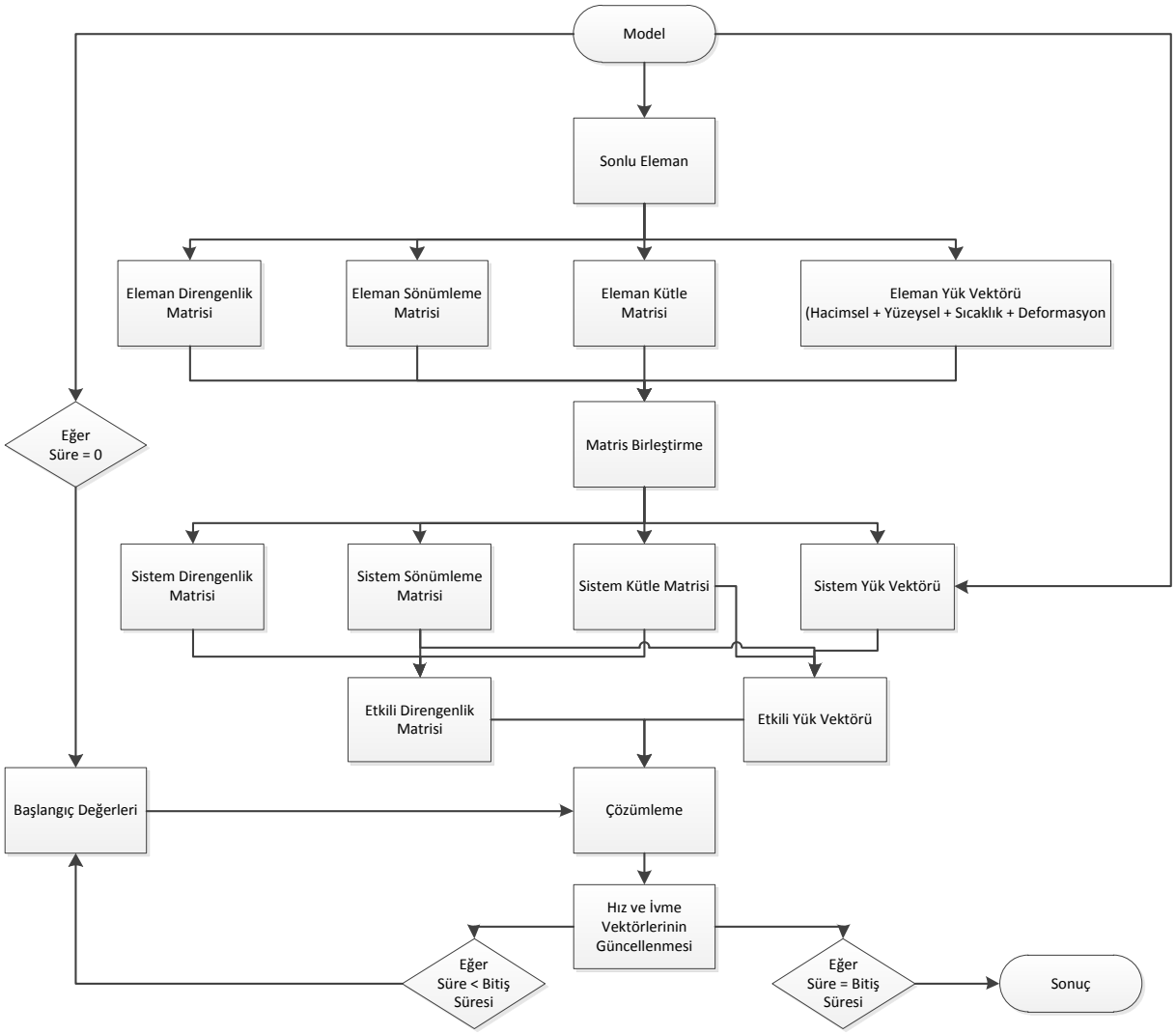
$$\begin{aligned}[\overline{K}]\{U\}_{n+1} = \{R^{ext}\}_{n+1} + [M]\left\{\frac{1}{\beta\Delta t^2}\{U\}_n + \frac{1}{\beta\Delta t}\{\dot{U}\}_n + \left(\frac{1}{2\beta}-1\right)\{\ddot{U}\}_n\right\} \\ + [C]\left\{\frac{\gamma}{\beta\Delta t}\{U\}_n + \left(\frac{\gamma}{\beta}-1\right)\{\dot{U}\}_n + \left(\frac{\gamma}{2\beta}-1\right)\{\ddot{U}\}_n\right\}\end{aligned} \tag{2-40}$$

burada

$$[\overline{K}] = \frac{1}{\beta\Delta t^2}[M] + \frac{\gamma}{\beta\Delta t}[C] + [K] \tag{2-41}$$

Yukarıdaki denklemlerde K, M, C sırasıyla direngenlik, kütle ve sönümleme matrislerini gösterir. $U, \dot{U}, \ddot{U}$ sembolleriyle sırasıyla deplesman, hız ve ivme gösterilmiştir. $\overline{K}$ Matrisi etkin direngenlik matrisi olarak adlandırılır. γ ve β ise algoritmanın kararlılık ve hata özelliklerini değiştiren faktörlerdir. Denklemin uygulandığı zaman aralıkları ise Δt sembolüyle belirtilmiştir.

Denklem 2-40'da görüldüğü üzere n+1 zamanında hesaplanan deplasmanlar hem n hem de n+1 zamanındaki değişkenlere bağlıdır. Algoritma örtük ismini bu özelliğinden alır. Denklem 2.40 ile tanımlanan etkin direngenlik matrisi $\overline{K}$, direngenlik matrisini içerdiği için köşegen matris olarak yazılamaz ve deplasmanların hesabı için bu matrisi çarpanlarına ayırmak zorunludur. Denklem 2.40 ve Denklem 2,41 de geçen γ ve β değerleri zaman aralığı Δt'den bağımsız olarak sayısal kararlılık sağlanabilmesi için $\gamma = \frac{1}{2}$ ve $\beta = \frac{1}{4}$olarak kullanılmıştır (Cook 2001).



Şekil 2-20 Örtük Çözümleme Algoritması

Şekil 2-20 örtük tümleştirme yöntemiyle zaman alanında doğrusal sinamik çözümleme algoritmasının akış diyagramını göstermektedir. Doğrusal statik algoritmadaki yapıya ek olarak çözümleme algoritması sonlu eleman kütüphanesindeki elemandan eleman kütle matrisini ve eleman sönümleme matrisini alır ve matris birleştirme yöntemiyle sistem kütle ve sönümleme matrislerinin oluşturur. Oluşan yük vektörü, kütle matrisi ve sönümleme matrisini kullanarak etkin yük vektörünü, direngenlik matrisi, sönümleme matrisi ve kütle matrisini kullanarak da etkin direngenlik matrisini oluşturur. Oluşturulan bu etkin direngenlik matrisi ve etkili yük vektörü doğrusal statik çözümleme algoritmasında olduğu gibi doğrusal çözülür ve düğüm noktalarındaki deplasman değerleri elde edilir. Bu deplasman değerleri kullanılarak düğüm noktalarındaki hız ve ivme değerleri hesaplanır. Bu değerler bir sonraki çözümleme işlemine dâhil edilir ve çözümleme işlemi son zaman dilimine erişilene kadar tekrarlanır. Elde edilen düğüm noktalarındaki değerler modelin sonuç dosyasına kaydedilir.

Doğrusal ve Doğrusal Olmayan Belirtik Çözümleme Algoritması

Belirtik dinamik çözümleme için Newmark Belirtik Algoritmasını kullanılır. Newmark Belirtik Çözümleme Algoritması, dinamik sonlu elemanlar metodu problemlerinin çözümlenmesinde kullanılan bir algoritmadır. Newmark dinamik çözümleme algoritmalarının bir türevi olan bu algoritma, ilk olarak Hughes ve Liu (1979) tarafından ortaya konulmuştur. Newmark Belirtik Çözümleme Algoritması, temelinde temel dinamik denklemindeki (Denklem 2.37) deplasman türevlerinin merkez diferansiyel formülleriyle (Denklem 2.42) hesaplanmasına dayanır.

$$f'(x) \cong \frac{f(x+h)-f(x-h)}{2h} \tag{2-42}$$

Denklem 2.42'de gösterilen fonksiyonun x noktasında h aralıklarla merkez fark (central difference) formüllerine göre türevini belirtir. Merkez fark yöntemi kullanılarak Denklem 2.37'deki temel hareket sürekli denklemi noktasal çözüm durumuna getirildiğinde Denklem 2-43'de anlatılan ifade elde edilir:

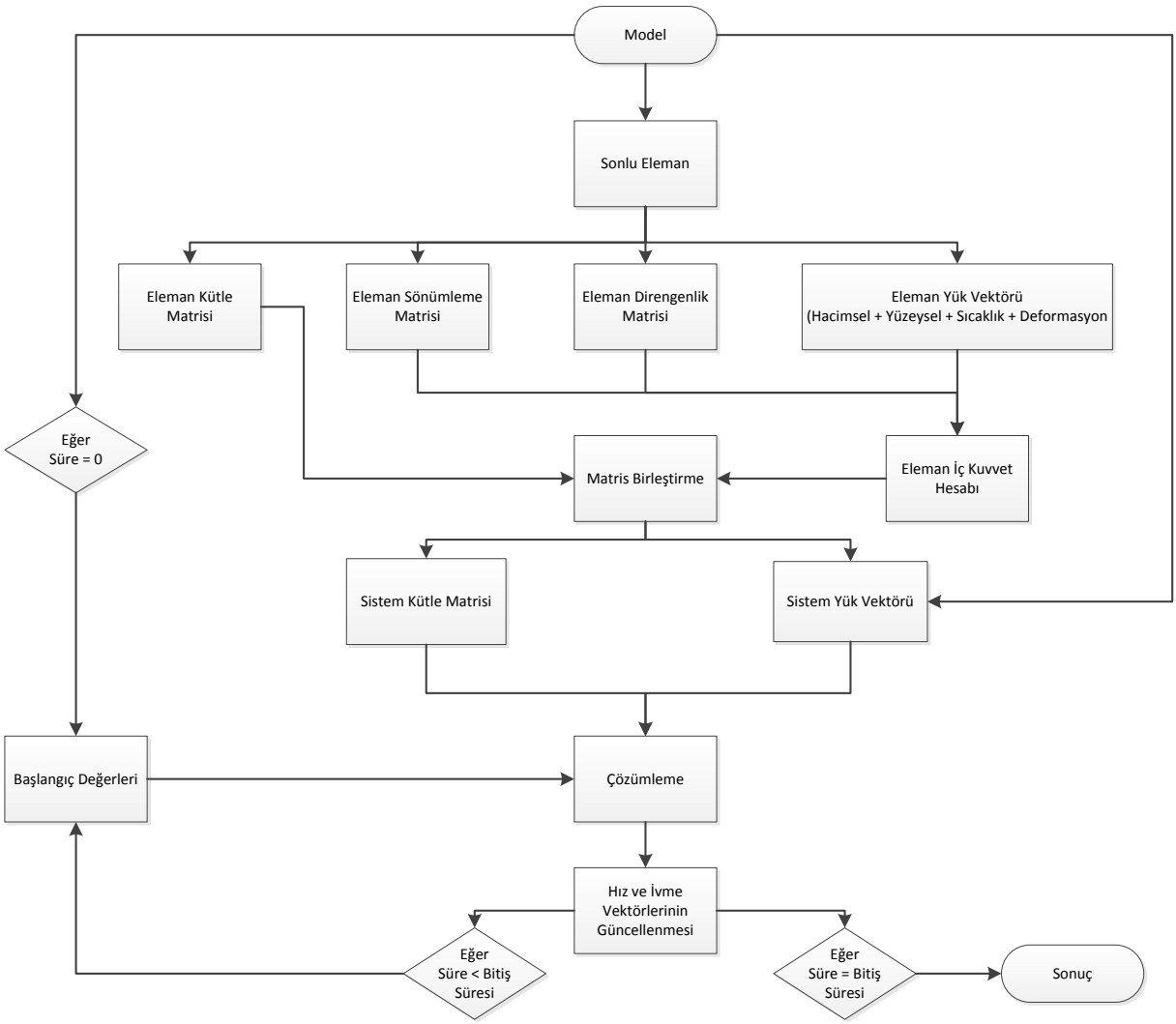
$$\frac{1}{\Delta t^2}[M]\{D\}_{n+1} = \{R^{Dış}\}_n - \{R^{İç}\}_n + \left[\frac{2}{\Delta t^2}M - \frac{1}{\Delta t}C\right]\{D\}_n - \left[\frac{1}{\Delta t^2}M - \frac{1}{\Delta t}C\right]\{D\}_{n-1} \tag{2-43}$$

Doğrusal sistemler için,

$$\{R^{İç}\} = [K]\{D\} \tag{2-44}$$

Denklem 2-43'ün sol tarafı direngenlik matrisi [K]'dan bağımsızdır. Eğer yoğunlaştırılmış (lumped) kütle matrisi varsayımı kullanılırsa, çözülmesi gereken denklemlerin birbirleriyle olan ilişkisi ortadan kalkar ve Denklem 2-43 sistem matrislerini oluşturmadan her sonlu eleman için ayrı ayrı çözülebilir. Ayrıca denklemin çözümü için *n+1* adımındaki direngenlik matrisi gerekmediğinden Denklem 2-43 hem doğrusal hem de doğrusal olmayan sistemlerin çözümünde kullanılabilir. Belirtik dinamik çözümleme algoritması koşullu kararlılık gösterir (Denklem 2-45), denklemin uygulanacağı zaman adımlarının büyüklüğü (Δt) sistemin özelliklerine göre kararlılığı sağlamak için yeterince küçük seçilmelidir. Belirtik dinamik çözümleme algoritmasının örtük dinamik çözümlemesinden temel farkı, belirtik dinamik algoritmasında *n* ve *n-1* zamanındaki terimler kullanılarak *n+1* deki deplasmanların bulunmasıdır. Buna karşı örtük algoritmada *n* ve *n+1* anlarına ait terimlerden *n+1* anındaki deplasmanlar bulunur.

$$\Delta t < \frac{2}{\omega_{maksimum}} \tag{2-45}$$



Şekil 2-21 Belirtik Çözümleme Algoritması

Şekil 2-21'de ise belirtik tümleştirme yöntemi kullanılarak gerçekleştirilen zaman alanında doğrusal dinamik çözümleme algoritmasının akış diyagramı gösterilmiştir. Belirtik çözümleme algoritmasının örtük çözümleme algoritmasından farkı sistem matrisi olarak yalnızca kütle matrisini oluşturmasıdır. Sonlu elemandan gelen eleman direngenlik matrisi, eleman sönümleme matrisi ve eleman yük vektörü sistem matrislerine eklenmek yerine eleman iç kuvvet vektörü hesabında kullanılır. Sonuç olarak sonlu elemandan eleman kütle matrisi ve eleman iç kuvvet vektörü alınır ve matris birleştirme yöntemi ile sistem kütle matrisi ve sistem dinamik yük vektörüne dönüştürülür. Kütle matrisinin köşegen olduğu durumlarda herhangi bir matris ayrıştırma yöntemine ihtiyaç duyulmadan basit aritmetik işlemlerle bir sonraki zaman adımına ait ivme değerleri elde edilir.

Zamana Bağlı Isı İletimi Çözümlemesi

Zaman bağlı ısı iletimi çözümlemesinde genel olarak iki yöntem kullanılmaktadır. Bunlar Örtük ve Belirtik Euler algoritmalarıdır. Her iki algoritma da genel zamana bağlı ısı iletimi denkleminin (Denklem 2-48) zaman alanında tümleştirilmesiyle çözülmektedir.

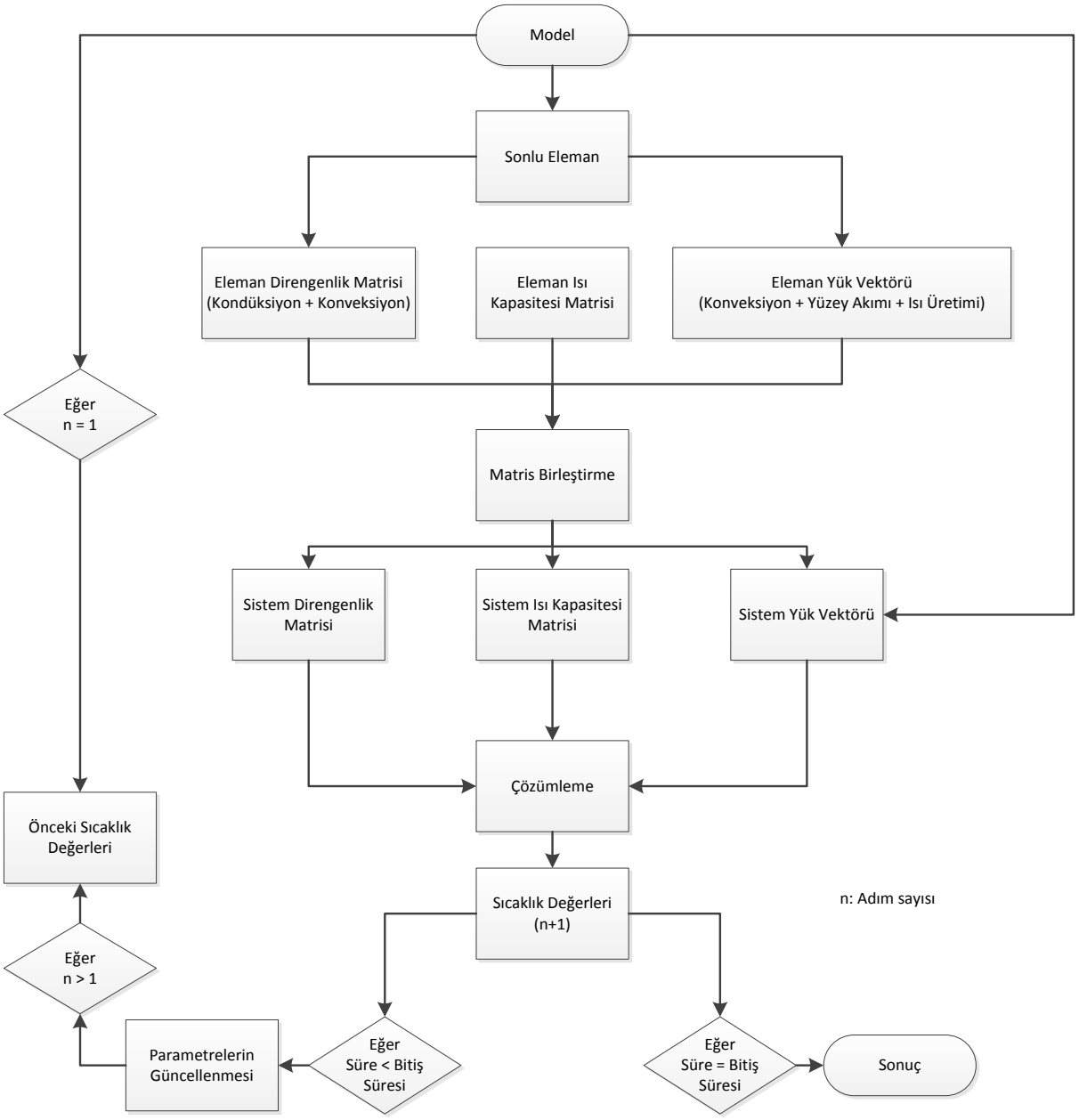
$$C\dot{T} + K_t T = F_t(t) \tag{2-48}$$

Denklem 2-48'de C enerji depolamadaki değişimi ifade eden matrisi, K_t ısı direngenlik matrisini, F_t ısı yükü vektörünü ve T düğüm noktalarındaki sıcaklık değerlerini içeren vektörü temsil etmektedir. Denklem 2-48'in çözümünde Taylor serisi açılımı kullanılarak denklem 2-49 elde edilmiştir.

$$\{\mathrm{T}\}_{n+1} = \{\mathrm{T}\}_n + \{(1 - \beta)\dot{T_n} + \beta(\dot{T_{n+1}})\}\Delta\mathrm{t} \tag{2-49}$$

Denklem 2-49'da T düğüm noktalarındaki sıcaklık değerlerini içeren vektörü, n adım sayısını, Δt zaman adımını ve β çözümleme algoritması yöntemini belirleyen bir katsayıyı ifade etmektedir. Bu aşamada Örtük Euler ve Belirtik Euler yöntemleri için β sırasıyla 1 ve 0 değerleri alır. Bunun yanı sıra kullanıcı istediği β değerini girerek farklı tümleştirme yöntemlerini de kullanabilmektedir.

Şekil 2-22 zamana bağlı ısı iletimi çözümleme yönteminin akış diyagramını göstermektedir. Zamana bağlı ısı iletimi algoritmasında sistem direngenlik matrisi ve sistem yük vektörüne ek olarak sistem ısı kapasitesi matrisi sonlu eleman kütüphanesideki elemandan gelen eleman matris ve vektörlerinin eklenmesiyle oluşturulur. Oluşturulan bu matris ve vektörler bir önceki zaman adımında düğüm noktalarındaki sıcaklık değerleri ve Belirtik ya da Örtük Euler yöntemi kullanılarak çözümlenir ve o zaman adımında düğüm noktalarındaki sıcaklık değerleri elde edilir. Elde edilen bu sıcaklık değerleri bir sonraki zamanın çözümleme işlemine katılarak yeni sıcaklık değerleri elde edilir. Bu döngü işlemi istenilen zamandaki sıcaklık değerleri elde edilene kadar devam eder. En son elde edilen sıcaklık değerleri modelin sonuç dosyasına kaydedilir.



Şekil 2-22 Zamana Bağlı Isı İletimi Çözümleme Algoritması

Doğrulama Testleri

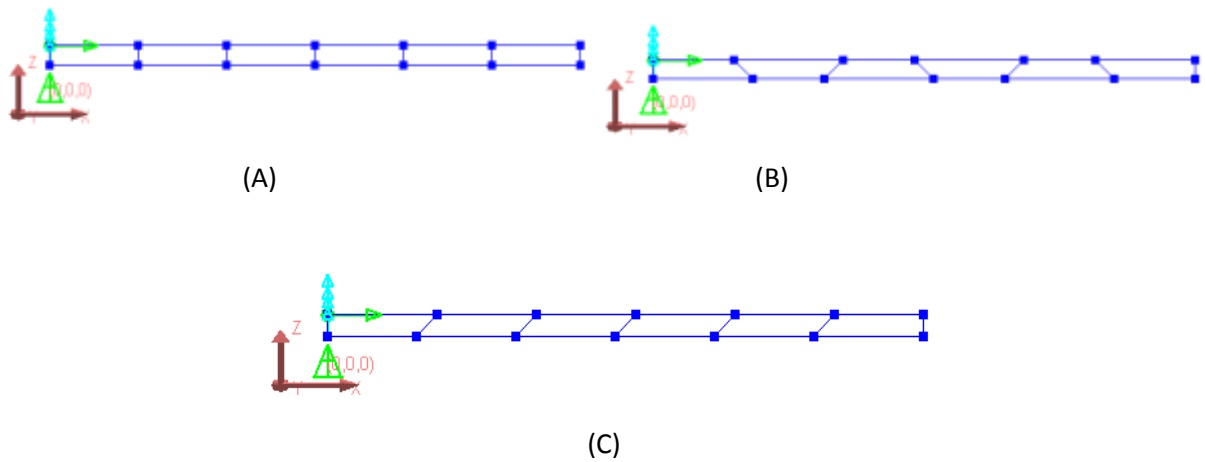
Platformdaki sonlu elemanlar kütüphanesindeki elemanların, malzeme modeli kütüphanesindeki modellerin ve çözümleme algoritmalarının sonuçlarının doğruluğunu sınamak amaçlı çeşitli doğrulama testleri yapılmıştır. Mekanik disiplin için MacNeal ve Harder (1985) tarafından önerilen problemler, ısı transferi disiplini için ise Reddy ve Gartling (2010) tarafından tavsiye ettiği dikdörtgen plaka problemi çözülmüştür ve sonuçlar analitik çözüm sonuçlarıyla karşılaştırılmıştır. Proje çerçevesinde çok sayıda doğrulama problemleri çözülmüş olsa da rapor içerisinde sınırlı sayıda sonuca yer verilmiştir.

Doğrusal Statik Problemler

Bu bölüm platformun mekanik disiplinindeki sonlu elemanlar kütüphanesindeki elemanların, malzeme modeli kütüphanesindeki modellerin ve mekanik çözümleme algoritmalarının çözümleme sonuçlarını içermektedir. Sonlu elemanlar kütüphanesindeki elemanların doğruluğu doğrusal statik çözümleme algoritması kullanılarak MacNeal ve Harder (1985) tarafından önerilen problemlerin çözümlenmesiyle sınanmıştır. Doğrusal olmayan malzeme ve doğrusal olmayan geometri problemlerinde konsol kiriş probleminin farklı eleman ağı sayıları kullanılarak çözümlenmesiyle doğrulanmıştır.

Problem 2-002: 2 Boyutlu Elemanlarla Düz Konsol Kiriş Problemi

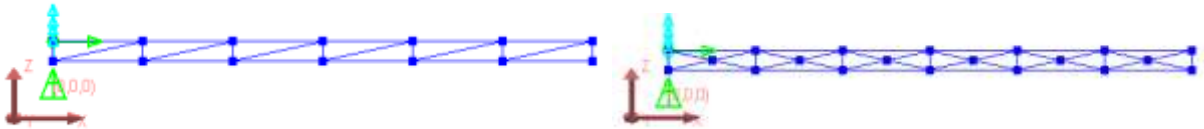
MacNeal ve Harder (1985) tarafından tavsiye edilen düz konsol kiriş problemleri ve özellikleri aşağıda gösterilmiştir.



(A) (B)

(C)

Şekil 2-23 Düz Konsol Kiriş Problemi. (A) Dikdörtgen Kabuk Elemanlar. (B) Yamuk Kabuk Elemanlar. (C) Paralelkenar Kabuk Elemanlar.



(D) (E)

Şekil 2-24 Düz Konsol Kiriş Problemi. (D) Dörtgen Başına 2 Üçgen Eleman. (E) Dörtgen Başına 4 Üçgen Eleman.

Çözümlemelerde kullanılan modelin parametreleri ise aşağıda belirtilmiştir.

Geometrik Özellikler:	Malzeme Özellikleri:	Kesit Özellikleri:
Uzunluk: 6	E: $1{,}00*10^{7}$	Kalınlık: 0.1
Genişlik: 0.2	v: 0.3	
	G: 3,846,154	

Sonlu elemanların performansını test edebilmek amacıyla modele uygulanan farklı yükleme tipleri Tablo 2-1'de gösterilmiştir.

Tablo 2-1 2 Boyutlu Düz Konsol Kiriş Problemi – Yükleme Durumları

Yükleme	Yükleme Tipi	Yük
1	Px Eksenel Uzama	Fx=0.5 serbest uçtaki tüm düğüm noktalarında
2	Vz ve My, Kesme ve Eğilme	Fz=0.5 serbest uçtaki tüm düğüm noktalarında
3	Vy ve Mz, Kesme ve Eğilme	Fy=0.5 serbest uçtaki tüm düğüm noktalarında
4	Mx Burulma	Fy=-5 serbest uçtaki alt düğüm noktalarında Fy=5 serbest uçtaki üst düğüm noktalarında
5	My Moment	Fx=-5 serbest uçtaki alt düğüm noktalarında Fx=5 serbest uçtaki üst düğüm noktalarında
6	Mz Moment	Mz=0.5 serbest uçtaki tüm düğüm noktalarında

Dörtgen ShellQ elemanının membran ve plak (ince ve kalın) tipleri kullanılarak çözümlenen düz konsol kirişin serbest uç düğüm noktalarındaki sehim değerleri sırasıyla Tablo 2-2 ve Tablo 2-3'te gösterilmiştir.

Tablo 2-2 ShellQ Elemanı ile 2 Boyutlu Düz Konsol Kiriş Problemi (Membran Etkisi) – Serbest Uç Düğüm Noktalarındaki Sehim Değerleri

Yükleme Tipi	Model	Sonuç Parametresi	ANSYS	Tip1	Tip2	Tip3	Analitik Sonuç
Px Eksenel Uzama	A	Ux	$3.00*10^{-5}$	$3.00*10^{-5}$	$3.33*10^{-5}$	$3.00*10^{-5}$	$3.00*10^{-5}$
	B		$3.00*10^{-5}$	$3.00*10^{-5}$	$3.45*10^{-5}$	$3.00*10^{-5}$	
	C		$3.00*10^{-5}$	$3.00*10^{-5}$	$3.36*10^{-5}$	$3.00*10^{-5}$	
Vz ve My Kesme ve Eğilme	A	Uz	0.107	0.010	0.108	0.105	0.108
	B		0.0227	0.003	0.102	0.017	
	C		0.0804	0.004	0.107	0.070	
My Moment	A	Ux	$8.99*10^{-4}$	$8.40*10^{-5}$	$9.00*10^{-4}$	$8.84*10^{-4}$	$9.00*10^{-4}$
	B		$1.40*10^{-4}$	$2.06*10^{-5}$	$8.43*10^{-4}$	$1.04*10^{-4}$	
	C		$7.23*10^{-4}$	$2.82*10^{-5}$	$8.98*10^{-4}$	$6.44*10^{-4}$	

Tablo 2-2'de belirtilen Tip1, Tip2 ve Tip3 sırasıyla doğrusal, ikinci derece ve dönme serbestlik derecesine sahip şekil fonksiyonları ile türetilmiş membran elemanı temsil etmektedir. Tablodaki sonuçlar incelendiğinde Tip 1 elemanı eksenel yükleme durumları dışındaki durumlarda yeterli değildir. Bu nedenle diğer tipler geliştirilmiştir. İkinci derece şekil fonksiyonlarına sahip eleman kullanılarak (Tip 2) kesme, eğilme ve moment durumlarında kabul edilebilir seviyede başarılı sonuçlar elde edilmesine rağmen eksenel yükleme durumunda elemanın Tip 1'den daha kötü sonuç elde ettiği görülebilmektedir. Bu nedenle doğrusal şekil fonksiyonlarına sahip elemana dönme direngenliğinin eklenmesiyle elde edilen Tip 3 elemanın performansı eksenel yükleme durumu için Tip 2'den ve kesme, eğilme, moment durumlarında da Tip 1'den daha başarılı olduğu görülmüştür. Bu nedenle tüm yükleme durumları göz önüne alındığında Tip 3 elemanının diğer tiplerden daha avantajlı olduğu sonucuna ulaşılabilir. Ayrıca eleman geometrisinin bozulmasıyla tüm tipler için doğruluk seviyesinin düştüğü görülmüştür.

Tablo 2-3 ShellQ Eleman ile 2 Boyutlu Düz Konsol Kiriş Problemi (Plak Etkisi) - Serbest Uç Düğüm Noktalarındaki Sehim Değerleri

Yükleme Tipi	MODEL	Sonuç Parametresi	ANSYS	ShellQ (ince Plak)	ShellQ (Kalın Plak)	Analitik Sonuç
Vy ve Mz Kesme ve Eğilme	A	Uy	0.430	0.432	0.429	0.432
	B		0.430	0.432	0.421	
	C		0.429	0.432	0.428	
Mx Burulma	A	Uy	$2.31*10^{-3}$	$2.33*10^{-3}$	$3.02*10^{-3}$	$3.41*10^{-3}$
	B		$2.32*10^{-3}$	$2.33*10^{-3}$	$2.80*10^{-3}$	
	C		$2.33*10^{-3}$	$2.33*10^{-3}$	$2.70*10^{-3}$	
Mx Moment	A	Rz	$3.60*10^{-2}$	$3.60*10^{-2}$	$3.60*10^{-2}$	$3.60*10^{-2}$
	B		$3.60*10^{-2}$	$3.60*10^{-2}$	$3.60*10^{-2}$	
	C		$2.33*10^{-3}$	$3.60*10^{-2}$	$3.60*10^{-2}$	

Tablo 2-3 incelendiğinde ShellQ elemanının ince plak tipinin performansının eleman geometrisinin bozulmasıyla değişmediği ve burulma durumu dışında (kesme, eğilme ve moment durumları) analitik

sonuç ile aynı sonucu verdiği görülebilmektedir. Burulma durumunda ise elemanın direngen davrandığı söylenebilir. Kalın plak tipinde ise burulma durumunda ince plak durumuna göre daha gerçekçi sonuçlar elde edilirken kesme ve eğilme durumda daha direngen davranış gözlemlenmiştir. Moment yüklemesi altında ise analitik değerlerle aynı sonuçlar elde edilmiştir. Bunun yanı sıra eleman geometrisinin bozulması kalın kabuk elemanının performansını olumsuz yönde etkilemektedir.

ShellT elemanı kullanılarak oluşturulan düz konsol kiriş modeli Şekil 2-24'te gösterilmiştir. ShellT elemanının membran, plak (ince ve kalın) tiplerinin kullanılmasıyla oluşturulan kirişin serbest uç noktalarındaki sehim değerleri sırasıyla Tablo 2-4 ve Tablo 2-5'te gösterilmiştir.

Tablo 2-4 ShellT Elemanı ile 2 Boyutlu Düz Konsol Kiriş Problemi – Serbest Uç Düğüm Noktalarındaki Sehim Değerleri

Yükleme Tipi	MODEL	Sonuç Parametresi	SAP2000	Tip1	Analitik Sonuç
Px Eksenel Uzama	D	Ux	$3.00*10^{-5}$	$3.00*10^{-5}$	$3.00*10^{-5}$
	E		$3.00*10^{-5}$	$3.00*10^{-5}$	
Vz ve My Kesme ve Eğilme	D	Uz	$3.20*10^{-3}$	$2.48*10^{-2}$	$1.08*10^{-1}$
	E		$6.60*10^{-3}$	$4.98*10^{-2}$	
My Moment	D	Ux	$2.65*10^{-5}$	$2.08*10^{-4}$	$9.00*10^{-4}$
	E		$5.50*10^{-5}$	$4.17*10^{-4}$	

Tablo 2-4'teki Tip1 doğrusal şekil fonksiyonlarıyla oluşturulmuş membran elemanını temsil etmektedir. Tablodaki değerler incelendiğinde eksenel yükleme altında analitik sonuçlar ile aynı değerler elde edilirken kesme, eğilme ve moment durumlarında performans olarak dörtgen membran elemanlarından daha düşük doğruluk seviyesine sahiptir. Bu nedenle geometrinin zorladığı durumlar dışında dörtgen membran elemanı kullanmak daha uygundur.

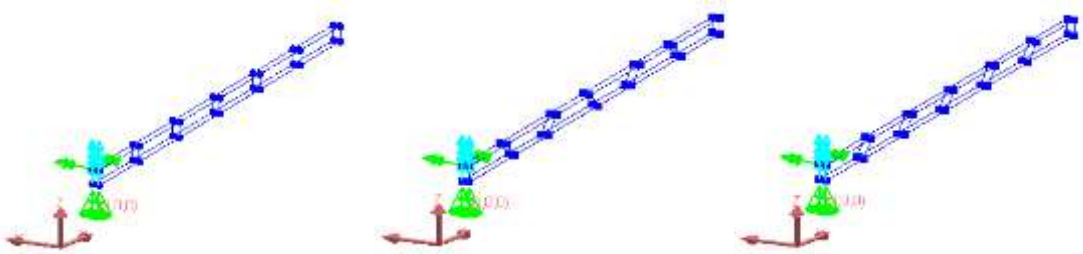
Tablo 2-5 ShellT Elemanı ile 2 Boyutlu Düz Konsol Kiriş Problemi (Plak Etkisi) - Serbest Uç Düğüm Noktalarındaki Sehim Değerleri

Yükleme Tipi	MODEL	Sonuç Parametresi	SAP2000	ShellT İnce Plak	ShellT Kalın Plak	Analitik Sonuç
Vz ve My Kesme ve Eğilme	D	Uy	0.430	0.430	0.428	0.432
	E		0.431	0.431	0.429	
Mx Burulma	D	Uy	$2.31*10^{-3}$	$2.30*10^{-3}$	$2.00*10^{-3}$	$3.41*10^{-3}$
	E		$2.30*10^{-3}$	$2.30*10^{-3}$	$2.27*10^{-3}$	
Mz Moment	D	Rz	$3.60*10^{-2}$			$3.60*10^{-2}$
	E		$3.60*10^{-2}$			

Tablo 2-5'te Shell T elemanının ince ve kalın plak tiplerinin performansları incelendiğinde ince plak elemanının performansının kalın plak elemanının performansından daha başarılı olduğu görülebilmektedir. Ancak membran elemanlarda olduğu gibi geometrinin zorladığı durumlar dışında dörtgen plak eleman kullanmak daha uygundur.

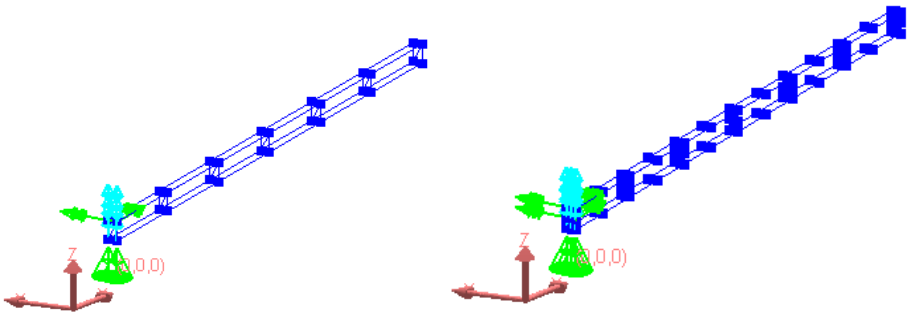
Problem 5-002: 3 Boyutlu Elemanlarla Düz Konsol Kiriş Problemi

Bir önceki bölümde çözümlenen düz konsol kiriş problemi bu aşamada üç boyutlu elemanlar kullanılarak farklı eleman ağı yoğunluklarıyla tekrar çözülmüştür. Çözümlemelerde kullanılan yapısal modeller Şekil 2-25 ve Şekil 2-26'da gösterilmiştir.



(A) (B) (C)

Şekil 2-25 Düz Konsol Kiriş Problemi. (A) Dikdörtgen Prizmatik Eleman. (B) Yamuk Prizmatik Eleman. (C) Paralelkenar Prizmatik Eleman



(D) (E)

Şekil 2-26 Wedge Elemanlar ile Düz Konsol Kiriş Problemi. (D) Wedge6. (E) Wedge15.

Çözümlemeler sırasında kullanılan model özellikleri de aşağıdaki gibidir.

Geometrik Özellikler:	Malzeme Özellikleri:
Uzunluk: 6	E: $1*10^7$
Genişlik: 0.2	v: 0.3
	G: 3,846,154

Sonlu elemanlar kütüphanesindeki 3 boyutlu elemanların performanslarını görmek için modeldeki konsol kirişe farklı yükleme durumları etki ettirilmiştir ve bu durumlar Tablo 2-6'da gösterilmiştir.

Tablo 2-6 3 Boyutlu Düz Konsol Kiriş Problemi – Yükleme Durumları

Yükleme	Yükleme Tipi	Yük
1	Px Eksenel Uzama	Fx=0.25 serbest uçtaki tüm düğüm noktalarında
2	Vz & My Kesme ve Eğilme	Fz=0.25 serbest uçtaki tüm düğüm noktalarında
3	Vy & Mz Kesme ve Eğilme	Fy=0.25 serbest uçtaki tüm düğüm noktalarında
4	Mx Burulma	Fy=-2.5 serbest uçtaki alt düğüm noktalarında Fy=2.5 serbest uçtaki üst düğüm noktalarında
5	My Moment	Fx=-2.5 serbest uçtaki alt düğüm noktalarında Fx=2.5 serbest uçtaki üst düğüm noktalarında
6	Mz Moment	Fx=5 serbest uçtaki y=0 düzlemindeki düğüm noktalarında Fx=-5 serbest uçtaki y=0.1 düzlemindeki düğüm noktalarında

Bu bölümde Brick8, geliştirilmiş Brick8, Brick20, Wedge6, Wedge15, Tet4 ve Tet10 elemanları ile oluşturulmuş konsol kirişin serbest uç düğüm noktalarındaki sehim değerleri sırasıyla Tablo 2-7, Tablo2-8, Tablo 2-9, Tablo 2-10, Tablo 2-11, Tablo 2-12 ve Tablo 2-13'te gösterilmiştir.

Tablo 2-7'deki Brick 8 elemanının sonuçları incelendiğinde eksenel yüklemede analitik sonuçlara çok yakın değerler elde edildiği ancak diğer yükleme durumlarında analitik sonuçlardan uzaklaşıldığı görülmektedir. Ayrıca elemanların geometrisi bozuldukça eleman performansı tüm yükleme durumları için olumsuz olarak etkilenmiştir.

Brick8 elemanının eksenel yükleme durum dışındaki yükleme durumlarındaki performansını iyileştirmek adına Brick8 elemanının doğrusal şekil fonksiyonuna ikinci derece denklemler ekleyerek geliştirilmiş Brick8 elemanı elde edilmiştir. Tablo 2-8'de geliştirilmiş Brick8 elemanının sonuçları incelendiğinde eksenel yükleme ve burulma durumlarında eleman performansında ciddi bir değişiklik olmadığı ancak kesme, eğilme ve moment durumlarında performansın iyileştiği görülmektedir. Ancak Brick8 elemanında olduğu gibi eleman geometrisindeki bozulma elemanın performansını olumsuz olarak etkilemektedir.

İkinci derece şekil fonksiyonlarının kullanılmasıyla oluşturulmuş Brick20 elemanının sonuçları incelendiğinde (Tablo 2-9) eksenel yükleme durumu dışındaki tüm yükleme durumlarında geliştirilmiş Brick8 elemanından daha iyi performans gösterdiği görülmektedir.

Tablo 2-10'da belirtilen Wedge6 elemanının sonuçları incelendiğinde diğer doğrusal elemanlarda olduğu gibi eksenel yükleme durumu dışındaki yükleme durumlarında elde edilen sonuçların analitik sonuçlara yeterince yakın olmadığı gözlemlenmiştir. Bu durumdan dolayı bu yükleme durumlarında daha iyi performans elde etmek amacıyla ikinci derece şekil fonksiyonları kullanılarak Wedge15 elemanı geliştirilmiştir. Ayrıca doğrusal üçgen prizma (Wedge6) elemanının performansı ile doğrusal dikdörtgenler prizması (Brick8) elemanının performansları kıyaslandığında genel olarak dikdörtgenler prizması elemanının sonuçlarının analitik sonuçlara biraz daha yakın olduğu gözlemlenmektedir.

Tablo 2-7 Brick8 Elemanın ile 3 Boyutlu Düz Konsol Kiriş Problemi – Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yüklem e Durumu	Mode l	Eleman Ağı	Sonuç Parametre si	SAP2000	Brick8	Analitik Sonuç
1 Px Eksenel Uzama	A	6x1x1	Ux	$2.91*10^{-5}$	$2.99*10^{-5}$	$3.00*10^{-5}$
		30x1x1		$2.98*10^{-5}$	$3.00*10^{-5}$	
		30x4x8		$3.00*10^{-5}$	$3.71*10^{-5}$	
	B	6x1x1		$2.92*10^{-5}$	$2.99*10^{-5}$	
	C	6x1x1		$2.92*10^{-5}$	$2.99*10^{-5}$	
2 Vz & My Kesme ve Eğilme	A	6x1x1	Uz	$1.00*10^{-2}$	$1.00*10^{-2}$	$1.08*10^{-1}$
		30x1x1		$7.18*10^{-2}$	$7.18*10^{-2}$	
		30x4x8		$1.07*10^{-1}$	$1.07*10^{-1}$	
	B	6x1x1		$2.80*10^{-3}$	$2.84*10^{-3}$	
	C	6x1x1		$3.40*10^{-3}$	$3.46*10^{-3}$	
3 Vy & Mz Kesme ve Eğilme	A	6x1x1	Uy	$1.09*10^{-2}$	$1.09*10^{-2}$	$4.32*10^{-1}$
		30x1x1		$1.56*10^{-1}$	$1.56*10^{-1}$	
		30x4x8		$4.17*10^{-1}$	$4.17*10^{-1}$	
	B	6x1x1		$4.50*10^{-3}$	$4.65*10^{-3}$	
	C	6x1x1		$6.20*10^{-3}$	$6.23*10^{-3}$	
4 Mx Burulma	A	6x1x1	Uy	$2.80*10^{-3}$	$2.86*10^{-3}$	$3.41*10^{-3}$
		30x1x1		$2.89*10^{-3}$	$2.92*10^{-3}$	
		30x4x8		$3.31*10^{-3}$	$3.39*10^{-3}$	
	B	6x1x1		$1.59*10^{-3}$	$1.62*10^{-3}$	
	C	6x1x1		$1.08*10^{-3}$	$1.10*10^{-3}$	
5 My Moment	A	6x1x1	Ux	$8.37*10^{-5}$	$8.38*10^{-5}$	$9.00*10^{-4}$
		30x1x1		$5.98*10^{-4}$	$5.99*10^{-4}$	
		30x4x8		$8.91*10^{-4}$	$9.55*10^{-4}$	
	B	6x1x1		$1.88*10^{-5}$	$1.94*10^{-5}$	
	C	6x1x1		$2.52*10^{-5}$	$2.56*10^{-5}$	
6 Mz Moment	A	6x1x1	Ux	$5.00*10^{-5}$	$4.53*10^{-5}$	$1.80*10^{-3}$
		30x1x1		$6.50*10^{-4}$	$6.50*10^{-4}$	
		30x4x8		$1.74*10^{-3}$	$1.87*10^{-3}$	
	B	6x1x1		$2.00*10^{-5}$	$1.72*10^{-5}$	
	C	6x1x1		$2.00*10^{-5}$	$2.51*10^{-5}$	

Tablo 2-8 Geliştirilmiş Brick8 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi – Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	SAP2000	GeliştirilmişBrick8	Analitik Sonuç
1 Px Eksenel Uzama	A	6x1x1	Ux	$2.93*10^{-5}$	$2.99*10^{-5}$	$3.00*10^{-5}$
		30x1x1		$2.99*10^{-5}$	$3.00*10^{-5}$	
		30x4x8		$2.98*10^{-5}$	$3.37*10^{-5}$	
	B	6x1x1		$2.97*10^{-5}$	$2.99*10^{-5}$	
	C	6x1x1		$2.97*10^{-5}$	$2.99*10^{-5}$	
2 Vz & My Kesme ve Eğilme	A	6x1x1	Uz	$1.06*10^{-1}$	$1.06*10^{-1}$	$1.08*10^{-1}$
		30x1x1		$1.08*10^{-1}$	$1.08*10^{-1}$	
		30x4x8		$1.07*10^{-1}$	$1.07*10^{-1}$	
	B	6x1x1		$5.10*10^{-3}$	$2.35*10^{-2}$	
	C	6x1x1		$6.73*10^{-2}$	$8.54*10^{-2}$	
3 Vy & Mz Kesme ve Eğilme	A	6x1x1	Uy	$4.20*10^{-1}$	$4.23*10^{-1}$	$4.32*10^{-1}$
		30x1x1		$4.30*10^{-1}$	$4.30*10^{-1}$	
		30x□x8		$4.29*10^{-1}$	$4.29*10^{-1}$	
	B	6x1x1		$1.29*10^{-2}$	$3.31*10^{-2}$	
	C	6x1x1		$2.28*10^{-1}$	$2.46*10^{-1}$	
4 Mx Burulma	A	6x1x1	Uy	$2.80*10^{-3}$	$2.86*10^{-3}$	$3.41*10^{-3}$
		30x1x1		$2.89*10^{-3}$	$2.92*10^{-3}$	
		30x4x8		$3.31*10^{-3}$	$3.34*10^{-3}$	
	B	6x1x1		$1.72*10^{-3}$	$1.82*10^{-3}$	
	C	6x1x1		$2.55*10^{-3}$	$3.36*10^{-3}$	
5 My Moment	A	6x1x1	Ux	$8.91*10^{-4}$	$8.95*10^{-4}$	$9.00*10^{-4}$
		30x1x1		$8.98*10^{-4}$	$8.99*10^{-4}$	
		30x4x8		$8.96*10^{-4}$	$9.31*10^{-4}$	
	B	6x1x1		$3.24*10^{-5}$	$1.46*10^{-4}$	
	C	6x1x1		$6.37*10^{-4}$	$7.59*10^{-4}$	
6 Mz Moment	A	6x1x1	Ux	$1.77*10^{-3}$	$1.79*10^{-3}$	$1.80*10^{-3}$
		30x1x1		$1.79*10^{-3}$	$1.80*10^{-3}$	
		30x4x8		$1.79*10^{-3}$	$1.86*10^{-3}$	
	B	6x1x1		$4.00*10^{-5}$	$1.64*10^{-4}$	
	C	6x1x1		$1.12*10^{-3}$	$1.23*10^{-3}$	

Tablo 2-9 Brick20 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi - Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	ANSYS	GeliştirilmişBrick8	Brick20	Analitik Sonuç
1 Px Eksenel Uzama	A	6x1x1	Ux	$3.00*10^{-5}$	$2.99*10^{-5}$	$3.01*10^{-5}$	$3.00*10^{-5}$
	B	6x1x1		$3.00*10^{-5}$	$2.99*10^{-5}$	$3.01*10^{-5}$	
2 Vz & My Kesme ve Eğilme	A	6x1x1	Uz	$1.05*10^{-1}$	$1.06*10^{-1}$	$1.07*10^{-1}$	$1.08*10^{-1}$
	B	6x1x1		$9.63*10^{-2}$	$2.35*10^{-2}$	$9.72*10^{-2}$	
3 Vy & Mz Kesme ve Eğilme	A	6x1x1	Uy	$4.15*10^{-1}$	$4.23*10^{-1}$	$4.16*10^{-1}$	$4.32*10^{-1}$
	B	6x1x1		$3.98*10^{-1}$	$3.31*10^{-2}$	$3.98*10^{-1}$	
4 Mx Burulma	A	6x1x1	Uy	$2.90*10^{-3}$	$2.86*10^{-3}$	$2.90*10^{-3}$	$3.41*10^{-3}$
	B	6x1x1		$3.14*10^{-3}$	$1.82*10^{-3}$	$3.09*10^{-3}$	
5 My Moment	A	6x□x1	Ux	$8.93*10^{-4}$	$8.95*10^{-4}$	$9.00*10^{-4}$	$9.00*10^{-4}$
	B	6x1x1		$8.31*10^{-4}$	$1.46*10^{-4}$	$8.30*10^{-4}$	
6 Mz Moment	A	6x1x1	Ux	$1.78*10^{-3}$	$1.79*10^{-3}$	$1.80*10^{-3}$	$1.80*10^{-3}$
	B	6x1x1		$1.72*10^{-3}$	$1.64*10^{-4}$	$1.66*10^{-3}$	

Wedge15 elemanının performans değerleri incelendiğinde (Tablo 2-11) genel olarak tüm yükleme durumlarında elde edilen sonuçların geliştirilmiş Brick8 elemanı kullanılarak elde edilen sonuçlara çok yakın olduğu görülmektedir. Ayrıca elemanın ikinci derece şekil fonksiyonları içermesinden dolayı kesme, eğilme ve moment durumlarında Wedge6 elemanından daha iyi performans sergilediği söylenebilir.

Tablo 2-10 Wedge6 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi - Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	ANSYS	Brick8	Wedge6	Analitik Sonuç
1 Px Eksenel Uzama	D	6x1x1	Ux	$2.99*10^{-5}$	$2.99*10^{-5}$	$2.98*10^{-5}$	$3.00*10^{-5}$
2 Vz & My Kesme ve Eğilme	D	6x1x1	Uz	$9.93*10^{-3}$	$1.00*10^{-2}$	$9.93*10^{-1}$	$1.08*10^{-1}$
3 Vy & Mz Kesme ve Eğilme	D	6x1x1	Uy	$1.09*10^{-2}$	$1.09*10^{-2}$	$1.09*10^{-2}$	$4.32*10^{-1}$
4 Mx Burulma	D	6x1x1	Uy	$2.12*10^{-3}$	$2.86*10^{-3}$	$2.11*10^{-3}$	$3.41*10^{-3}$
5 My Moment	D	6x1x1	Ux	$8.29*10^{-5}$	$8.38*10^{-5}$	$8.29*10^{-5}$	$9.00*10^{-4}$
6 Mz Moment	D	6x1x1	Ux	$4.56*10^{-5}$	$4.53*10^{-5}$	$1.65*10^{-4}$	$1.80*10^{-3}$

Tablo 2-11 Wedge15 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi - Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	ABAQUS	Geliştirilmiş Brick8	Wedge15	Analitik Sonuç
1 Px Eksenel Uzama	E	6x1x1	Ux	$3.02*10^{-5}$	$2.99*10^{-5}$	$3.01*10^{-5}$	$3.00*10^{-5}$
2 Vz & My Kesme ve Eğilme	E	6x1x1	Uz	$1.06*10^{-1}$	$1.06*10^{-1}$	$1.06*10^{-1}$	$1.08*10^{-1}$
3 Vy & Mz Kesme ve Eğilme	E	6x1x1	Uy	$4.17*10^{-1}$	$4.23*10^{-1}$	$4.17*10^{-1}$	$4.32*10^{-1}$
4 Mx Burulma	E	6x1x1	Uy	$2.82*10^{-3}$	$2.86*10^{-3}$	$2.85*10^{-3}$	$3.41*10^{-3}$
5 My Moment	E	6x1x1	Ux	$8.99*10^{-4}$	$8.95*10^{-4}$	$8.99*10^{-4}$	$9.00*10^{-4}$
6 Mz Moment	E	6x1x1	Ux	$1.67*10^{-6}$	$1.79*10^{-3}$	$1.80*10^{-3}$	$1.80*10^{-3}$

Tablo 2-12 Tet4 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi - Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	Ansys	Tet4	Analitik Sonuç
1 Px Eksenel Uzama	A	288	Ux	$2.99*10^{-5}$	$2.98*10^{-5}$	$3.00*10^{-5}$
2 Vz & My Kesme ve Eğilme	A	288	Uz	$1.29*10^{-2}$	$1.28*10^{-2}$	$1.08*10^{-1}$
3 Vy & Mz Kesme ve Eğilme	A	288	Uy	$1.45*10^{-2}$	$1.44*10^{-2}$	$4.32*10^{-1}$
4 Mx Burulma	A	288	Uy	$9.01*10^{-2}$	$8.99*10^{-5}$	$3.41*10^{-3}$
5 My Moment	A	288	Ux	$1.07*10^{-2}$	$1.07*10^{-4}$	$9.00*10^{-4}$
6 Mz Moment	A	288	Ux	$6.08*10^{-5}$	$6.07*10^{-5}$	$1.80*10^{-3}$

Tablo 2-13 Tet10 Elemanı ile 3 Boyutlu Düz Konsol Kiriş Problemi - Serbest Uçtaki Düğüm Noktalarındaki Sehim Değerleri

Yükleme Durumu	Model	Eleman Ağı	Sonuç Parametresi	Ansys	Tet10	Analitik Sonuç
1 Px Eksenel Uzama	A	288	Ux	$3.06*10^{-5}$	$3.01*10^{-5}$	$3.00*10^{-5}$
2 Vz & My Kesme ve Eğilme	A	288	Uz	$1.03*10^{-1}$	$9.55*10^{-2}$	$1.08*10^{-1}$
3 Vy & Mz Kesme ve Eğilme	A	288	Uy	$3.77*10^{-1}$	$3.57*10^{-1}$	$4.32*10^{-1}$
4 Mx Burulma	A	288	Uy	$4.85*10^{-3}$	$3.10*10^{-3}$	$3.41*10^{-3}$
5 My Moment	A	288	Ux	$8.80*10^{-4}$	$7.39*10^{-4}$	$9.00*10^{-4}$
6 Mz Moment	A	288	Ux	$1.64*10^{-3}$	$1.29*10^{-3}$	$1.80*10^{-3}$

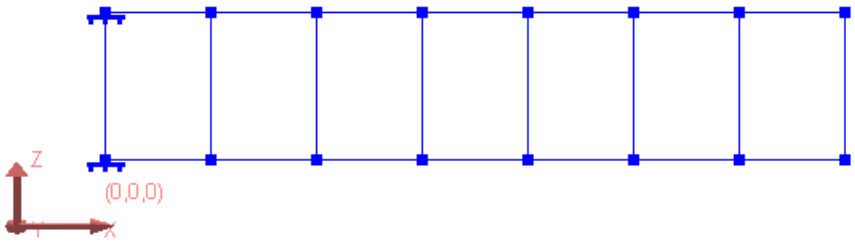
Aynı kirişin doğrusal şekil fonksiyonlarına sahip düzgün dörtyüzlü (Tet4) eleman kullanılarak çözümünden elde edilen sonuçlara göre (Tablo 2-12) eksenel yükleme durumu dışındaki yükleme durumlarında eleman performansının yeterince başarılı olmadığı söylenebilmektedir. Bu eleman geometrinin zorladığı yerlerde Brick elemanlarının geometrilerinin bozulmasıyla performansının düşmesi

nedeniyle bu tür geometrilerde kullanılmak amacıyla geliştirildiğinden düzgün geometrik şekilleri modellemek için bu elemanı kullanmak uygun değildir.

Tet10 elemanının sonuçları incelendiğinde elemanın Tet4 elemanından daha iyi performans sunduğu görülmektedir (Tablo 2-13). Ancak Tet4 elemanında olduğu gibi bu eleman düzgün geometrik şekillerin modellenmesi için uygun değildir.

Doğrusal Olmayan Problemler

Hem doğrusal olmayan geometrik davranışı anlatmak için uygulanan eleman formülasyonunu hem de doğrusal olmayan çözümleme algoritmasını doğrulamak için Louie L. Yaw (2009) tarafından tavsiye edilen düz konsol kiriş problemi büyük deformasyon (large strain) teorisi kullanılarak çözümlenmiştir. Bu aşamada Brick8 elemanı kullanılarak modellenen kirişin (Şekil 2-27) serbest uç düğüm noktasındaki sehim değerleri ANSYS sonuçları ile kıyaslanmıştır.



Şekil 2-27 Düz Konsol Kiriş Problemi (Louie L. Yaw)

Şekil 2-27'de gösterilen kirişin özellikleri aşağıdaki gibidir.

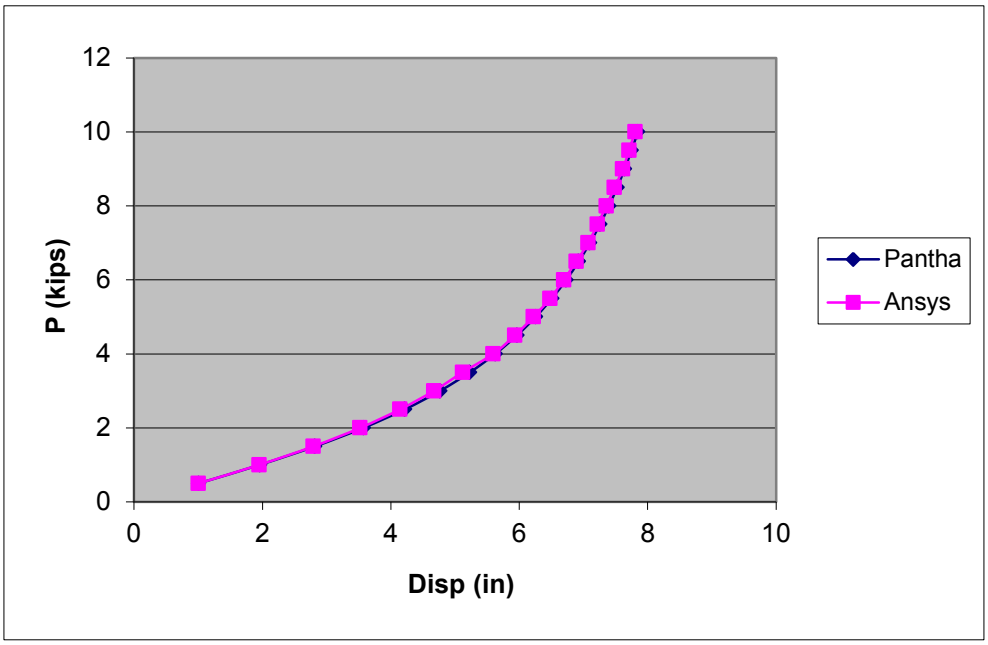
Geometrik Özellikler:	Malzeme Özellikleri:	Kesit Özellikleri:
Uzunluk: 10	E: 100	Kalınlık: 2
Genişlik: 2	v: 0	
Eleman Sayısı: 7		

Yükleme Özellikleri:

Aşamalı Yükleme Sayısı: 20

Tolerans: 1e-3

Brick8 elemanı ile oluşturulmuş kiriş elemanların serbest uç noktasındaki sehim değerleri farklı yük aşamaları için Şekil 2-28'de gösterilmiştir.

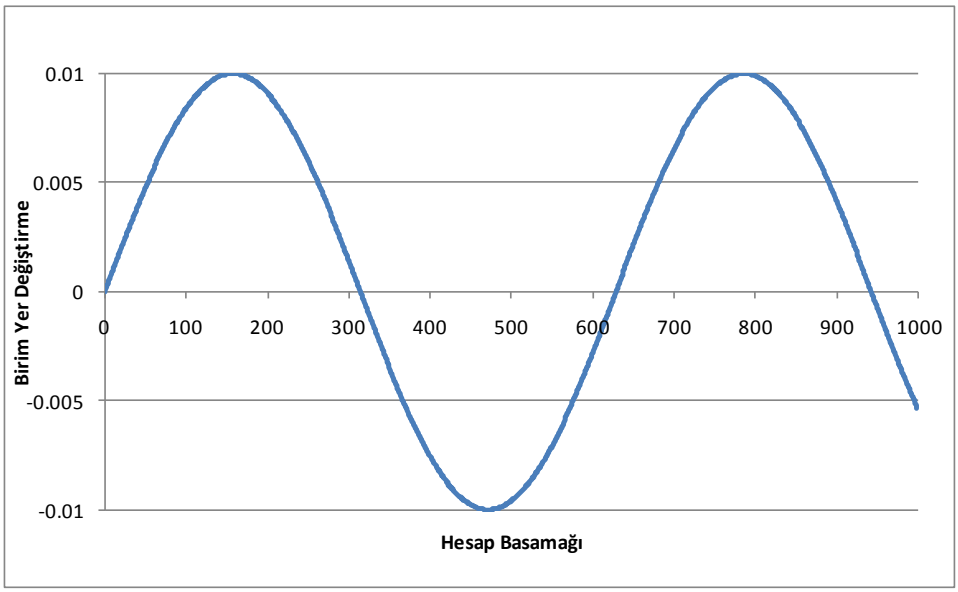


Şekil 2-28 Brick8 Elemanı ile Düz Konsol Kiriş Problemi – Serbest Uç Noktalarındaki Sehim Değerlerinin Kıyaslanması

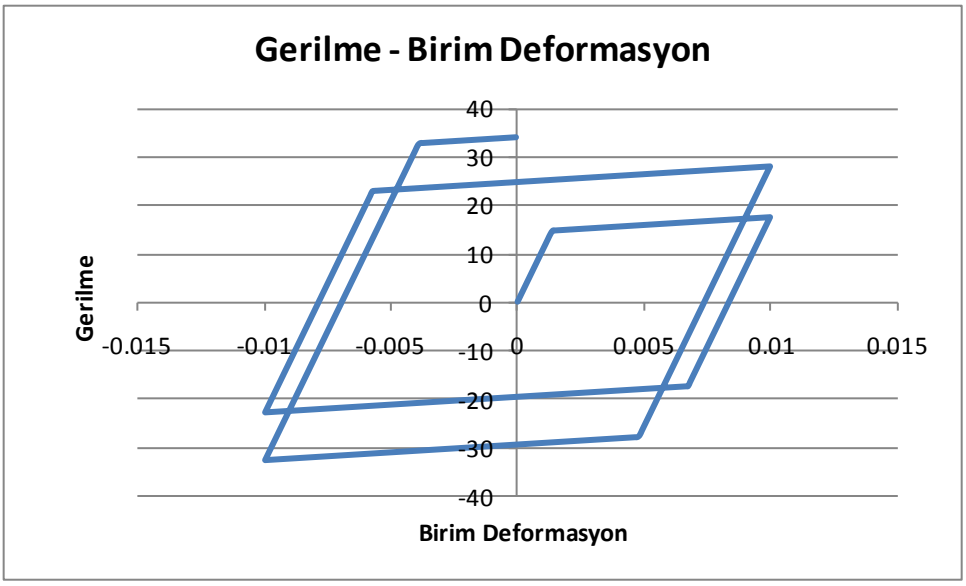
Şekil 2-28'deki sonuçlara göre doğrusal olmayan çözümleme algoritmasının performansı ANSYS ticari yazılımının doğrusal olmayan geometri performansına oldukça yakındır.

Doğrusal Olmayan Malzeme Davranışı

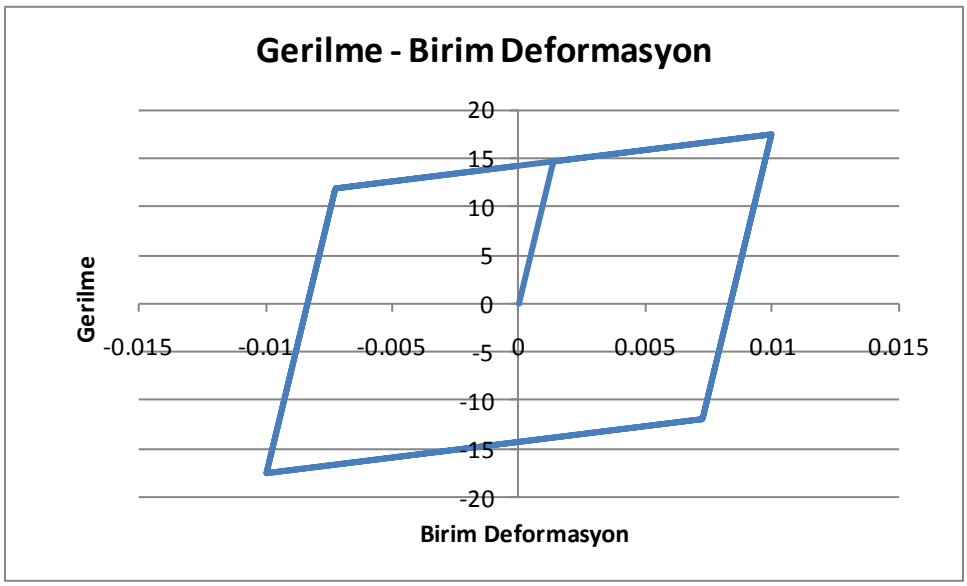
İlk olarak, Von Mises akma kuralı kullanılarak formüle edilen doğrusal olmayan malzeme modelinin doğruluğunu sınamak amacıyla aşağıdaki şekilde gösterilen (Şekil 2-29) birim yer değiştirme tanımı gerilme noktasında sadece kesme deformasyonu yapacak şekilde, diğer bir deyişle ε_{xy} ve ε_{xz} değerlerine toplam 1000 adımda aşağıdaki yer değiştirmeler uygulanmış ve elde edilme gerilme değerleri takip eden şekillerde sadece izotropik pekleşme (Şekil 2-30), sadece kinematik pekleşme (Şekil 2-31) ve aynı anda izotropik ve kinematik pekleşmenin olduğu durumlardaki (Şekil 2-32) gerilme-birim yer değiştirme grafikleri elde edilmiştir. Hesaplarda akma gerilmesi olarak 36 Mpa kullanılmıştır.



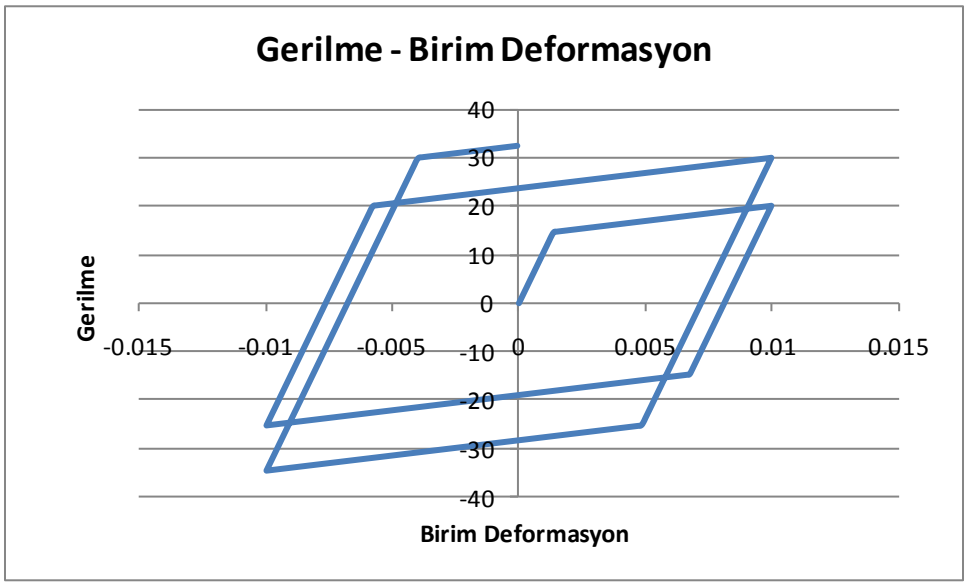
Şekil 2-29 Gerilme Noktasına Uygulanan Birim Yer değiştirme



Şekil 2-30 Kesme Birim Deformasyon Uygulaması – İzotropik Pekleşme



Şekil 2-31 Kesme Birim Deformasyon Uygulaması – Kinematik Pekleşme



Şekil 2-32 Kesme Birim Deformasyon Uygulaması – İzotropik / Kinematik Pekleşme

Sadece izotropik pekleşmenin olduğu durum için gerilme noktası her dönüşte daha büyük bir gerilmede akmıştır. Bu durum hesaplarda akma yüzeyinin devamlı genişlediğini göstermekte ve izotropik pekleşme tanımının doğru bir şekilde uygulandığını anlatmaktadır. Kinematik pekleşme durumunda ise gerilme noktası ilk olarak yaklaşık 15 civarında akmış, ancak tersi yönde akmak daha küçük, yaklaşık 12.5 gibi bir değerde gerçekleşmiştir. Bu durum akma yüzeyinin merkezinin kaydığını göstermekte ve kinematik pekleşme tanımının doğru yapıldığını göstermektedir. Son durumda ise hem izotropik hem de kinematik pekleşmenin olduğu durum incelenmiş ve tutarlı sonuçlar elde edilmiştir.

İkinci aşamada, hem malzeme modelinin sonlu elemanlarda uygulanışının hem de doğrusal olmayan çözümleme algoritmasının doğruluğunu sınamak için önceki bölümlerde çözümlenen McNeal ve Harder (1985) tarafından tavsiye edilen ve Mekanik Doğrulama Testleri bölümünde parametreleri belirtilmiş olan düz konsol kiriş modeli (Şekil 2-25a) Brick8 elemanı kullanılarak modellenmiş ve sonuçları ANSYS programının sonuçları ile kıyaslanmıştır. Konsol kirişin serbest uç düğüm noktalarındaki sehim değerleri ve serbest olmayan uçtaki elemanın düğüm noktalarındaki gerilme değerleri kıyaslanmıştır.

Brick8 elemanı ile oluşturulmuş konsol kirişin serbest uç noktalarındaki sehim değerleri Tablo 2-14'te gösterilmiştir.

Tablo 2-14 Brick8 Elemanı ile Düz Konsol Kiriş Problemi – Serbest Uç Düğüm Noktasındaki Sehim Değerleri

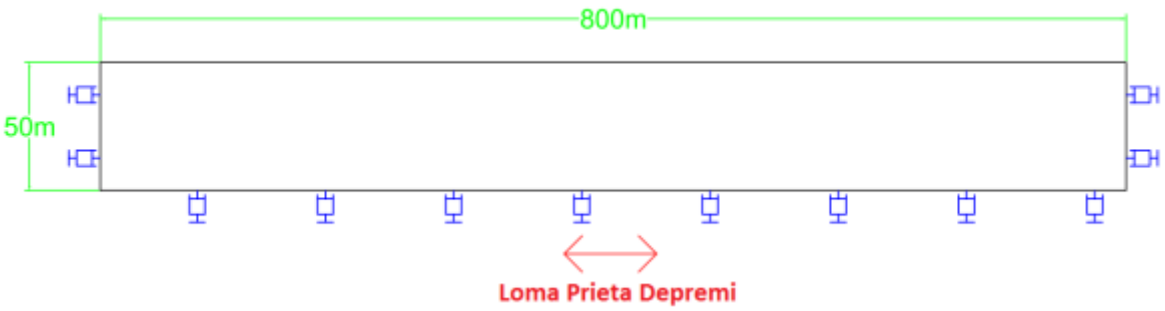
Yükleme Tipi	Model	Eleman Ağı	Sonuç Parametresi	ANSYS	Brick8
1 Px Eksenel Uzama	A	6x1x1	Ux	$3.93*10^{-5}$	$3.93*10^{-5}$
2 Vz & My Kesme ve Eğilme	A	6x1x1	Uz	$1.48*10^{-2}$	$1.48*10^{-2}$
3 Vy & Mz Kesme ve Eğilme	A	6x1x1	Uy	$1.61*10^{-2}$	$1.61*10^{-2}$
4 Mx Burulma	A	6x1x1	Uy	$4.27*10^{-3}$	$4.27*10^{-3}$
5 My Moment	A	6x1x1	Ux	$1.22*10^{-4}$	$1.22*10^{-4}$
6 Mz Moment	A	6x1x1	Ux	$6.66*10^{-5}$	$6.66*10^{-5}$

Tablo 2-14'teki sonuçlara göre Brick8 elemanıyla birlikte kullanılan doğrusal olmayan malzeme performansı ANSYS ticari yazılımının doğrusal olmayan malzeme davranışı ile tamamen aynı olduğu görülmektedir.

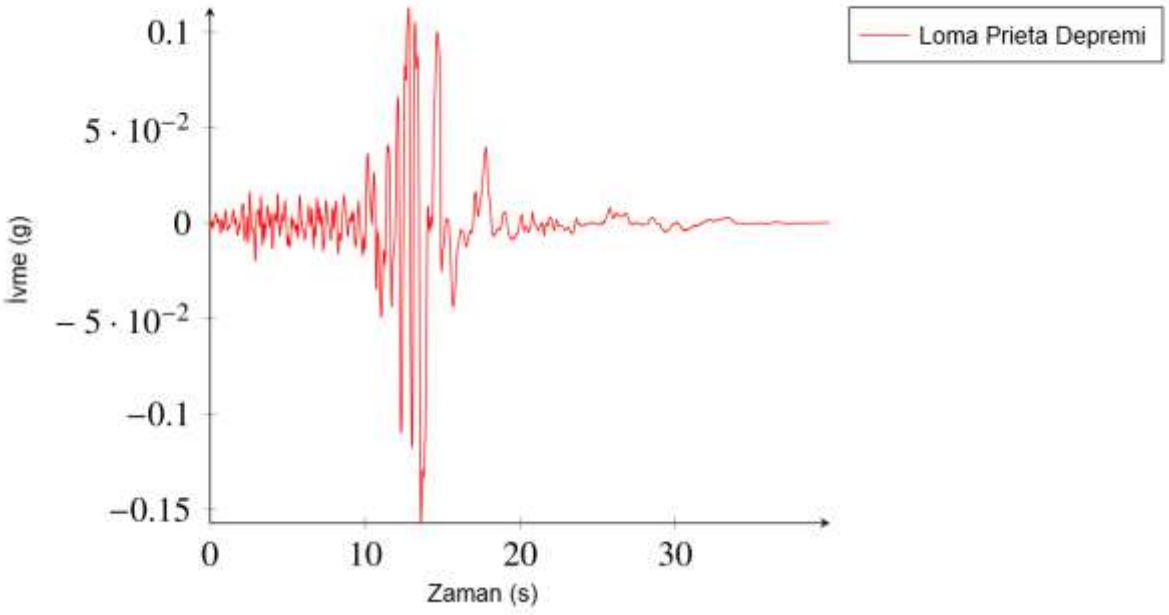
Eşdeğer Doğrusal Malzeme Modeli Doğrulaması

Bu bölümde platforma eklenen eşdeğer doğrusal malzeme modelinin doğrulaması yapılmıştır. Doğrulamada eşdeğer doğrusal malzeme modelinin verdiği sonuçlar EduShake programından alınan sonuçlar ile karşılaştırılmıştır. EduShake programı eşdeğer malzeme modelini malzeme kütüphanesinde uygulanan şekliyle içerir, ancak problem çözümünde zaman alanını değil frekans alanını kullanır. EduShake frekans alanında probleme uygulanan deprem yüklerini (EduShake programı sadece deprem yüklerini kabul etmektedir.) sinüs dalgalarının kombinasyonu olarak temsil eder ve toprağın depreme verdiği toplam tepkiyi her sinüs dalgasına verilen tepkileri toplayarak hesaplar. Edushake'in platformdaki çözüm yaklaşımından diğer farkları modeli tek boyutlu olarak çözmesi ve çözüm sırasında frekansdan bağımsız sönümleme kullanmasıdır. Oysaki, platformda analizler iki ya da üç boyutlu yapılabilir ve frekansa bağımlı olarak çalışan Reyleigh sönümleme tekniğini kullanılabilir.

Şekil 2-33 doğrulama sırasında kullanılan modeli sergiler. 50 metrelik bir toprak katmanına Loma Prieta depremi uygulanmıştır (Şekil 2-34). Gerçekte toprak modeli sonsuz uzunlukta yatay uzunluğa sahiptir ve yatayda sonsuza doğru dalgalanır. Toprak katmanının bu özelliğini karşılamak için model sönümleyicilerle (damper) çevrilmiştir. Ayrıca model yatayda yeterince (800 metre) geniş tutulmuştur.



Şekil 2-33 Eşdeğer Soğrusal Malzeme Modeli Doğrulamasında Kullanılan Model



Şekil 2-34 Loma Prieta Depremi İvme Değerleri

Toprağın ve modelde kullanılan eleman ağının özellikleri Tablo 2-15'de gösterilmiştir. Eşdeğer doğrusal malzeme modelinde kullanılan sönümleme ve kesme modülü değerleri Vucetic ve Dobry'den (1991) alınmıştır. Modelde kullanılan toprak meshinin özellikleri denklem 2-50 kullanırak belirlenmiştir.

$$\Delta l = \frac{V_s}{10f} \tag{2-50}$$

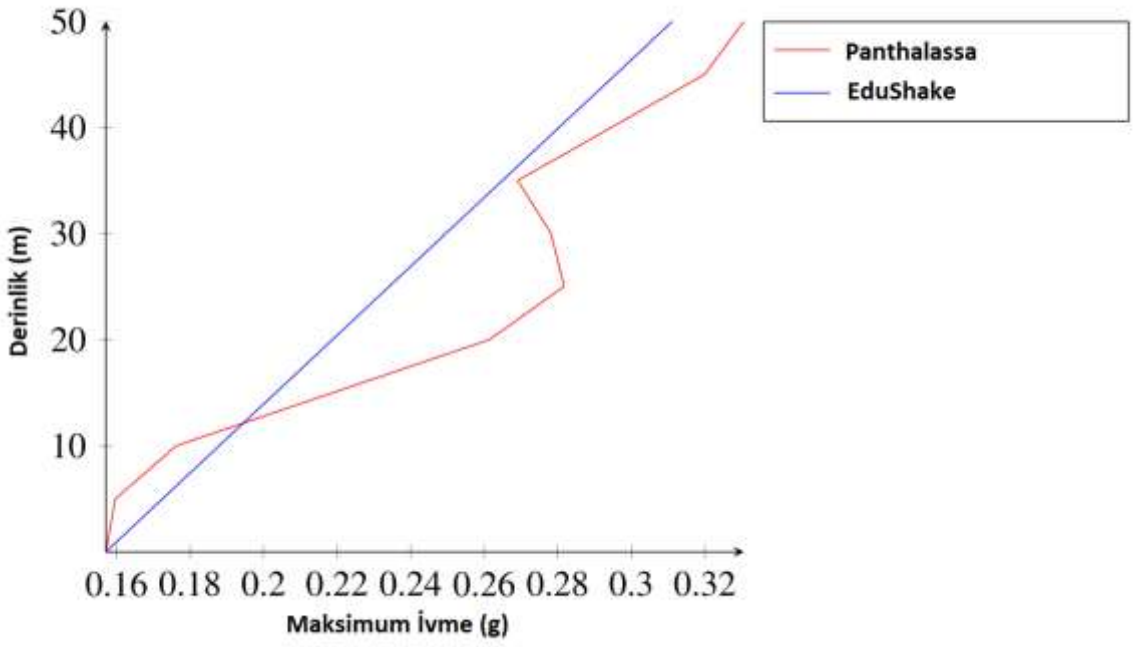
Bu denklemde Δl deprem dalgalarının aktarımı için gerekli en büyük mesh elemanı büyüklüğünü, V_s toprağın kesme hızını f ise deprem dalgalarının içerdiği en büyük frekansı temsil eder.

Loma Prieta depremi 10 Hz'den büyük frekansta dalgalar içermediği için dikeyde 2 metrelik eleman ağı denklem 2-50'ye göre uygundur. Ancak eşdeğer doğrusal malzeme modeli kesme modülüsünde azalmayı öngördüğünden denklem 2-50 kesme modülünde 25 kat kesme modülü azalması öngürelerek hesaplanmış ve dikeyde 0.5 metrelik eleman boyu kullanılmıştır.

Tablo 2-15 Eşdeğer Doğrusal Malzeme Modeli Doğrulama Testi Model Özellikleri

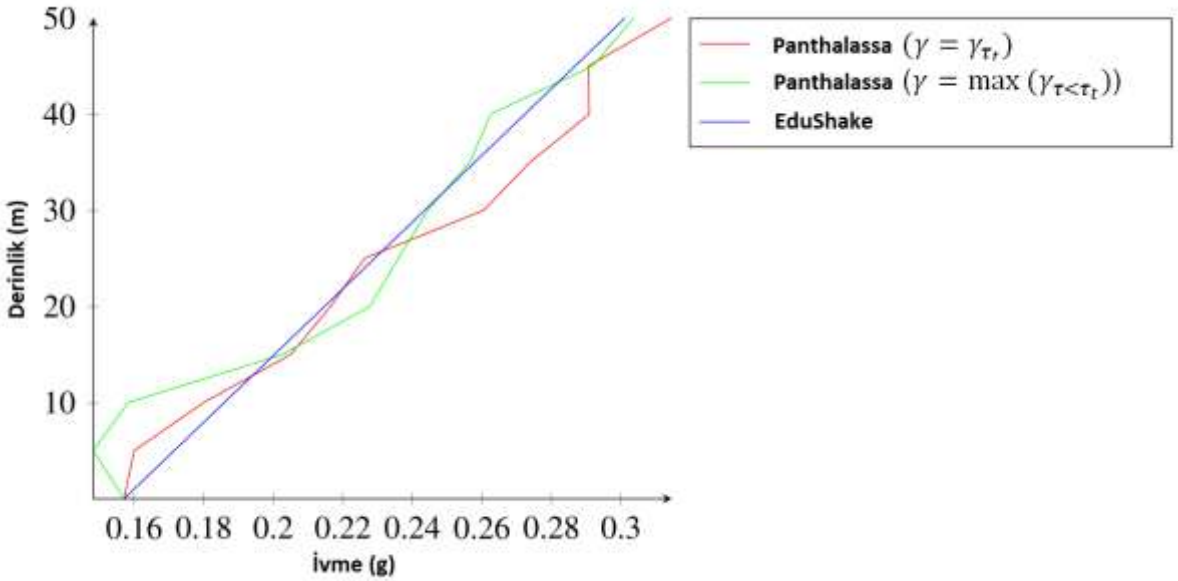
Parametre	Değer
E (Elastik Modülüs)	200,000 kPa
v (Poisson Oranı)	0.25
V_s (Kesme Dalgası Hızı)	200 m/s
γ(Toprak Ağırlığı)	19.6 kN/m^2
g (Yerçekimi İvmesi)	9.81 m/s^2
PI (Plastisite İndisi)	100
Rayleigh Sönümleme Frekansı 1	1.5
Rayleigh Sönümleme Frekansı 2	4.5
Model Büyüklüğü	50m x 800m
Eleman Ağı Büyüklüğü	0.5m x 10m

Modelin eleman ağı özelliklerini test etmek ve zaman ve frekans alanları arasında temel bir karşılaştırma yapmak için %5'lik bir sönümleme kullanılaral doğrusal bir analiz yapılmıştır. Şekil 2-35 bu analizin sonucunda toprağın değişik katmanları için hesaplanan (Panthalassa) ve EduShake'in hesapladığı maksimum ivme değerlerini gösterir. Panthalassa'nın zaman alanında yaptığı analiz ortalama olarak 0.02g daha büyük ivme değerlerine ulaşmıştır. Zaman alanında yapılan analizin doğrusal olmayan sonuçları iki boyutlu modelin ve Rayleigh sönümleme tekniğinin sonucudur. İki analiz farklılar gösterse de sonuçlar yakındır.



Şekil 2-35 Panthalassa ve EduShake karşılaştırması: Toprak Katmanında Değişik Derinlikte Ulaşılan Maksimum İvme Değerleri, Doğrusal Analiz

Doğrusal analizden sonra eşdeğer doğrusal analiz yapılmıştır. Eşdeğer doğrusal analiz hem her zaman adımındaki kesme gerinimini ($\gamma = \gamma_{\tau_t}$) hem de o ana kadar ulaşılan maksimum kesme gerinimi ($\gamma = \max(\gamma_{\tau<\tau_t})$) kullanılarak yapılmıştır. Şekil 2-36 bu analizler sonucunda toprağın değişik katmanlarında ulaşılan maksimum ivme değerlerini gösterir. Doğrusal analiz sonuçlarına benzer şekilde zaman alanında yapılan analizler hafifçe daha büyük ivme değerleri ile sonuçlanmaktadır. Bütün analiz tipleri için eşdeğer doğrusal analiz doğrusal analize göre daha büyük ivme değerleri ortaya çıkarmıştır (Tablo 2-16). Bu sonuç Eşdeğer doğrusal analiz sırasında toprağın kesme direnci azaldığı için beklenen bir durumdur. Bu sonuçlarla eşdeğer doğrusal malzeme modeli doğrulanmıştır.



Şekil 2-36 Panthalassa ve EduShake karşılaştırması: Toprak Katmanında Değişik Derinlikte Ulaşılan Maksimum İvme Değerleri, Eşdeğer Doğrusal Analiz

Tablo 2-16 Eşdeğer Doğrusal Malzeme Modeli Doğrulama Testi: Toprak katmanının Üstünde Elde Edilen Maksimum İvme Değerleri

	Zaman Alanı (Panthalassa)	Frekans Alanı (EduShake)
Doğrusal	0.3306 g	0.3112 g
Eşdeğer Doğrusal ($\gamma = \gamma_{\tau_t}$)	0.3146 g	0.3012 g
Eşdeğer Doğrusal ($\gamma = \max(\gamma_{\tau<\tau_t})$)	0.3036 g	

Örtük ve Belirtik Algoritmaların Doğrulanması

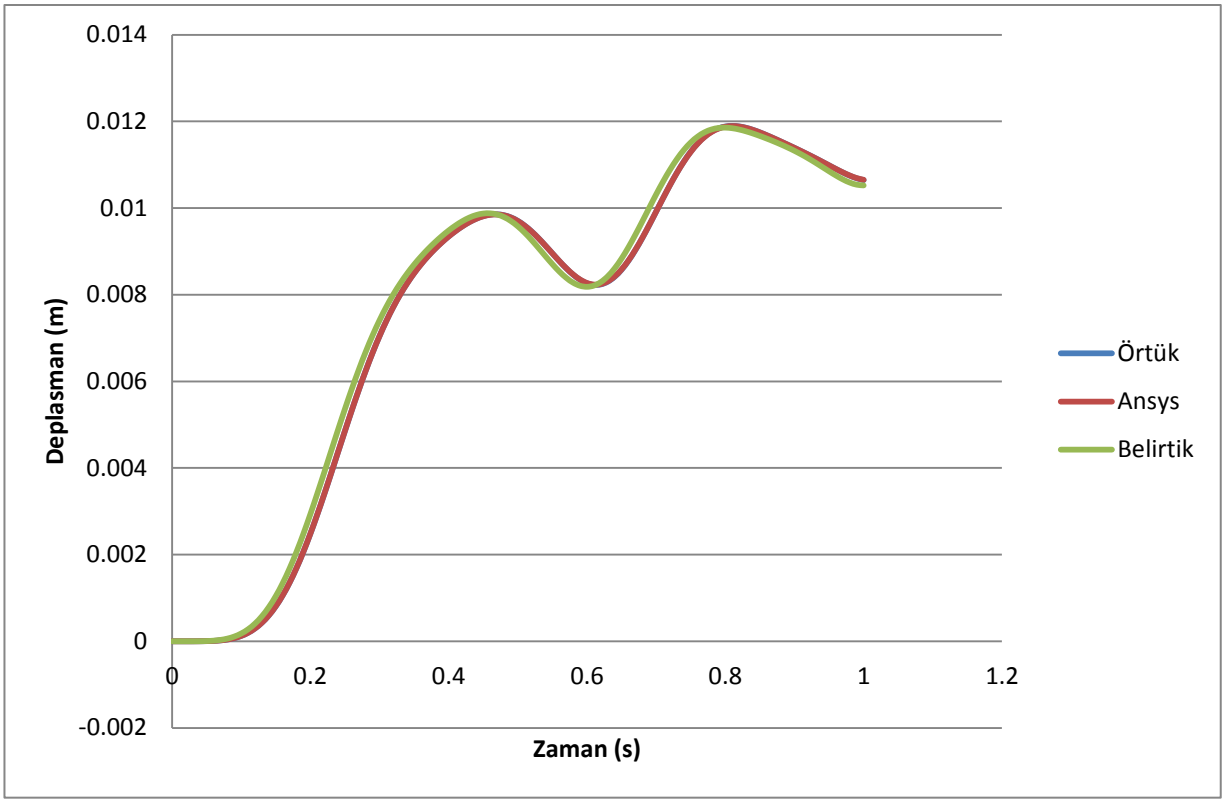
Dinamik çözümleme algoritmalarının doğrulanması için aşağıda Şekil 2-37'de gösterilen dinamik kiriş problemi hem ANSYS hem de platform bünyesindeki dinamik çözümleme algoritmaları ile çözülmüş ve sonuçlar karşılaştırılmıştır. Bu amaçla, doğrusal dörtgen membran elemanlardan oluşan modelin sol üst köşesine ani bir kuvvet 0.01 saniye boyunca uygulanmış ve bir saniye boyunca modelin üst sağ köşesinde oluşan x yönündeki deplasmanlar hesaplanmıştır. Kirişin sağ üst köşesindeki deplasmanların zamana göre değişimi Şekil 2-38'de ve maksimum deplasmanlar ise Tablo 2-17'de gösterilmiştir.

Şekil 2-37 Örnek Kiriş Modeli

Tablo 2-17 Modelin Sağ Üst Köşesinde X Yönünde Oluşan Maksimum Deplasmanlar

Ansys	Belirtik	Örtük
0.011892 m	0.011856 m	0.011897 m

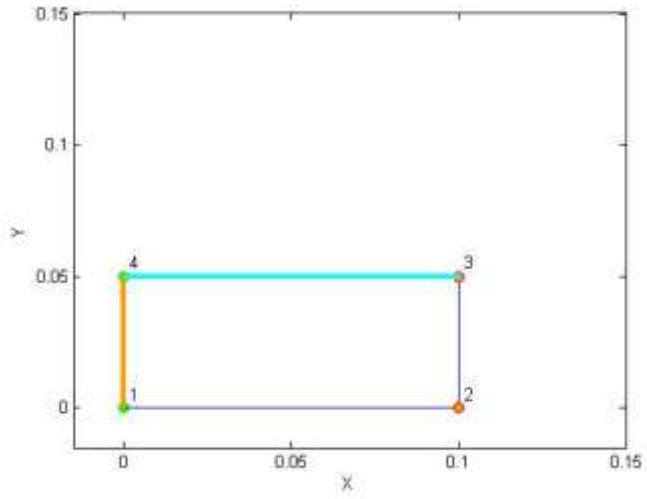
Yukarıdaki sonuçlardan da görülebileceği gibi hem belirtik hem de örtük algoritma sonuçları ANSYS programının sonuçlarıyla uyuşmaktadır.



Şekil 2-38: Modelin Sağ Üst Köşesinde X Yönünde Oluşan Deplasman Grafiği

Isı İletimi Problemleri

Isı transferi disiplininin doğrulama testleri için Reddy ve Gartling (2010) tarafından tavsiye edilen dikdörtgen plaka problemi çözülmüştür. Bu aşamada sonlu elemanlar kütüphanesindeki eleman modelleri ve doğrusal zamana bağlı olmayan ve doğrusal zamana bağlı çözümleme algoritmalarının başarımı sınanmıştır. Doğrulama için kullanılan ve Reddy ve Gartling (2010) tarafından önerilen dikdörtgen plaka problemi Şekil 2-39'de gösterilmiştir.



Şekil 2-39 Reddy and Gartling (2010) Tarafından Önerilen Dikdörtgen Plaka

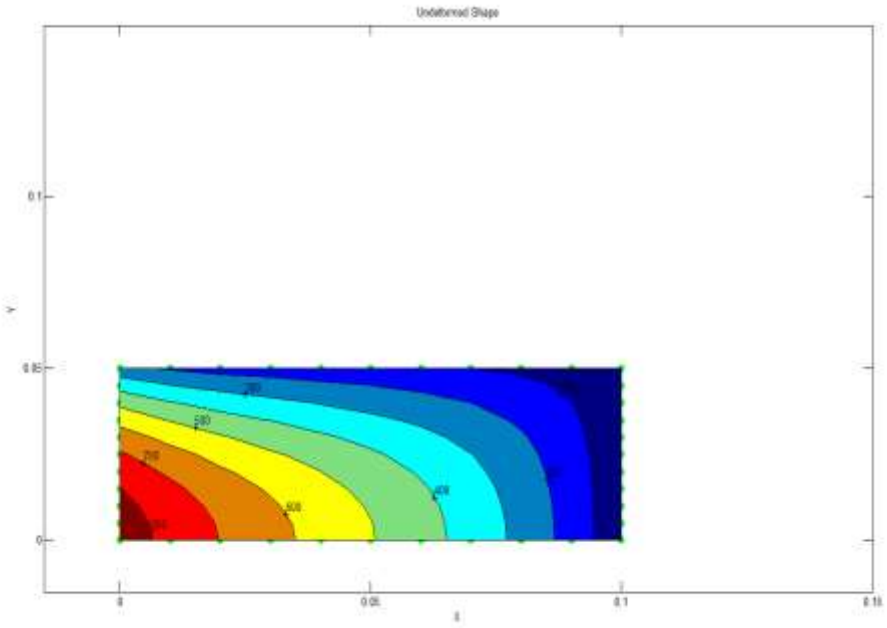
Şekil 2-39'da gösterilen dikdörtgen plakanın turuncu renkli köşe noktalarının (2 ve 3) sıcaklık değerleri sabitlenmiş, yine turuncu renkle gösterilmiş kenarında ısı akımı ve açık mavi ile çizilmiş kenarında ise ısı konveksiyonu tanımlanmıştır. Ayrıca plaka içerisinde ısı üretimi mevcuttur. Plakanın kondüksiyon değeri her yönde eşit olarak alınmıştır.

Isı iletimi çözümlemesi sırasında kullanılan model özellikleri aşağıda belirtildiği gibidir.

Geometrik Özellikler:	Malzeme Özellikleri:	Kesit Özellikleri:
Uzunluk: 0.1m	Isı Kondüksiyonu: 0.4 W/m.^{o}C	Kalınlık: 1m
Genişlik: 0.05m	Isı Konveksiyonu: 60 W/m^2	
	Dış Akışkan Sıcaklığı: 100 ^{o}C	
	Isı Akımı: 3500 W/m^2	
	Isı Üretimi (birim hacim): 135300 W/m^3	

Sabit Sıcaklık: 25 ^{o}C

Yukarıda tanımlanan problem sonlu elemanlar kütüphanesindeki tüm elemanlar için çözülmüş ve sonuçlar analitik sonuçlar ile karşılaştırılmıştır. Elde edilen sonuçlar Tablo 2-18'de gösterilmiştir. Ayrıca plakanın sıcaklık dağılımı Şekil 2-40'da gösterilmiştir.



Şekil 2-40 Dikdörtgen Plakanın Sıcaklık Dağılımı

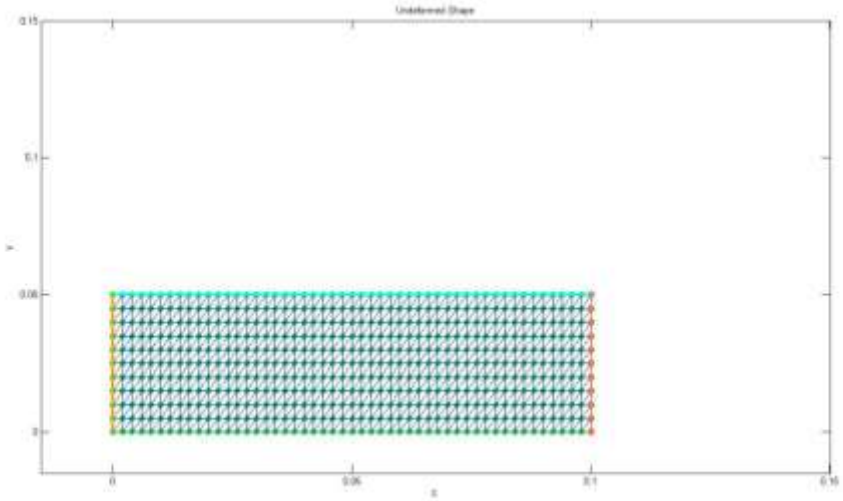
Tablo 2-18 Sonlu Elemanlar Kütüphanesindeki Elemanların Performans Değerleri

Eleman Tipi	Eleman Sayısı	1		3 ve 4'ün tam ortası		4
		Sıcaklık (°C)	Akım (x) (W/m^2)	Sıcaklık (°C)	Akım (y) (W/m^2)	Sıcaklık (°C)
TriM3	50	$8.55*10^2$	$3.21*10^3$	$1.28*10^2$	-	$2.32*10^2$
TriM6	50	$8.54*10^2$	$3.49*10^3$	$1.28*10^2$	-	$2.33*10^2$
RecM4	25	$8.55*10^2$	$3.12*10^3$	$1.28*10^2$	$6.20*10^3$	$2.27*10^2$
RecM8	25	$8.54*10^2$	$3.47*10^3$	$1.28*10^2$	$6.17*10^3$	$2.33*10^2$
Brick8	25	$8.56*10^2$	$3.04*10^3$	$1.29*10^2$	$5.38*10^3$	$2.24*10^2$
Brick20	25	$9.17*10^2$	$3.46*10^3$	$1.98*10^2$	$5.95*10^3$	$3.06*10^2$
Wedge6	32	$8.24*10^2$	$2.81*10^3$	$1.28*10^2$	$5.27*10^3$	$2.37*10^2$
Wedge15	32	$8.54*10^2$	$3.48*10^3$	$1.28*10^2$	$6.19*10^3$	$2.33*10^2$
Analitik Çözüm	-	$8.54*10^2$	$3.50*10^3$	$1.28*10^2$	$6.19*10^3$	$2.34*10^2$

Tablo 2-18'deki sonuçlar incelendiğinde mekanik elemanlarda olduğu gibi geometri zorlamadığı sürece dörtgen elemanlar kullanmak daha uygundur. İkinci derece şekil fonksiyonlarına sahip elemanların sonuçları doğrusal şekil fonksiyonlarına sahip elemanların sonuçlarına göre analitik değerlere daha yakındır.

Zamana bağlı ısı iletimi çözümleme algoritmalarının doğrulanması için TriM3 elemanı kullanılarak yine aynı problem 1000 eleman kullanılarak (Şekil 2-41) çözülmüştür. Yukarıda belirtilen model özelliklerine

ek olarak 40000 saniyelik süre için hesaplama yapılmıştır. Zamana bağlı çözümleme algoritmalarının sonuçları Tablo 2-19'de gösterilmiştir.



Şekil 2-41 1000 TriM3 Elemanlı Dikdörtgen Plaka

Tablo 2-19 Isı İletimi Çözümleme Algoritmalarının Performans Sonuçları

TriM3 1000 Eleman t = 40000 saniye	Düğüm Noktası 1'deki Sıcaklık Değeri (°C)
Zamana Bağlı Olmayan	854.51
Zamana Bağlı (Belirtik _ Lumped)	853.55
Zamana Bağlı (Belirtik)	853.57
Zamana Bağlı (Örtük _ Lumped)	853.53
Zamana Bağlı (Örtük)	853.55

Tablo 2-19'deki sonuçlara göre belirtik ya da örtük algoritma kullanmanın ve ısı kapasite matrisini yoğunlaşmış (lumped) ya da dağıtık (consistent) olarak hesaplamanın sonuçlarda çok ufak değişikliğe sebep olduğu görülmektedir.

3. Bölüm

Paralel Sonlu Elemanlar Çözümleme Platformu Tasarımı

Giriş

Yeni nesil yazılımların geliştirilmesinde önemli konular arasında, yazılım büyüdükçe farklı araştırmacıların yazılımı geliştirmekte zorlanmaları, yazılımın bir bölümünde yapılan bir değişikliğin yazılımın diğer parçalarını bozabilme riskinin var olması ve paralel hesaplama ihtiyacının getirdiği ek karmaşılıklar olarak sayılabilir. Dolayısıyla, yapı mühendisliği için genişletilebir yeni bir sonlu elemanlar platformunun geliştirilmesindeki en önemli ihtiyaç, platformun veri yapısının anlaşılabilir, kullanılabilir ve modüler bir yapıda olup, yeni parçaların eklenmesine olanak sağlamasıdır. Bu amaçla, proje çerçevesinde Panthalassa isimli bir yazılım geliştirilmiş, yazılımın nesne yönelimli veri yapısı özel olarak tasarlanmış ve genel bir çözümleme motoru geliştirilmiştir. Bu çözümleme motoruna "plug-in" teknolojisi kullanılarak yeni parçaların eklenmesi sağlanmıştır. Günümüzdeki çeşitli ticari yazılımlardan farklı olarak, çözümleme moturuna eleman, malzeme modeli gibi parçaların yanı sıra, çözümleme algoritması, bölümleme algoritması ve hatta farklı veri dosyalarını okuyup platform yapısına dönüştüren çeviriciler gibi farklı parçaların, çözümleme motorunu değiştirmeden, motorun kaynak koduna ihtiyaç duymadan eklenmesi sağlanmıştır.

Panthalassa, Sonlu Elemanlar Çözümleme Platformu

Panthalassa sonlu elemanlar yöntemiyle mühendislik problemlerinin analizi için geliştirilmiş çözümleme platformudur. Panthalassa çok işlemcili veya çok bilgisayarlı diğer bir deyişle paylaşımlı veya dağıtık bellekli, paralel sistemlerde yüksek başarımlı çözümleme için tasarlanmıştır. Panthalassa geliştirilirken yüksek başarım kıstasına ek olarak, platformun isteğe göre yeni özelliklerle geliştirilebilmesi, genişletilebilmesi ve günümüz kodlama standartlarına uygun olarak yazılması üstünde durulmuştur. Bu amaçlarla oluşturulan Panthalassa platformu, nesne yönelimli mimariyle hazırlanıp, sonradan geliştirilebilecek eklerle (plug-in) genişletilebilecek şekilde hazırlanmıştır.

Panthalassa nesne yönelimli mimarileri olanaklı kılan ve başarım odaklı programların yazılmasındaki başarısıyla genel olarak kabul görmüş C++ diliyle yazılmıştır. C++ programlama dili JAVA, Python gibi programlama dillerinin aksine her hangi bir yorumlayıcıya ihtiyaç duymaz ve bilgisayar işlemcisinin doğrudan işleyebileceği yönergeler (instruction) ortaya çıkartır. Bu şekilde sadece piyasadaki en son teknoloji işlemcilerin algılayabileceği yüksek performanslı yönergeler kullanılabilir. C, FORTRAN gibi diğer yorumlayıcıya ihtiyaç duymayan programlama dilleri ise C++ dilinin sahip olduğu yüksek seviyeli programlama dili özelliklerine (nesne yönelimli programlama, meta-programlama vs.) sahip değillerdir. Panthalassa çözümleme platformu C++ programlama dilinin getirdiği performans ve yüksek seviyeli

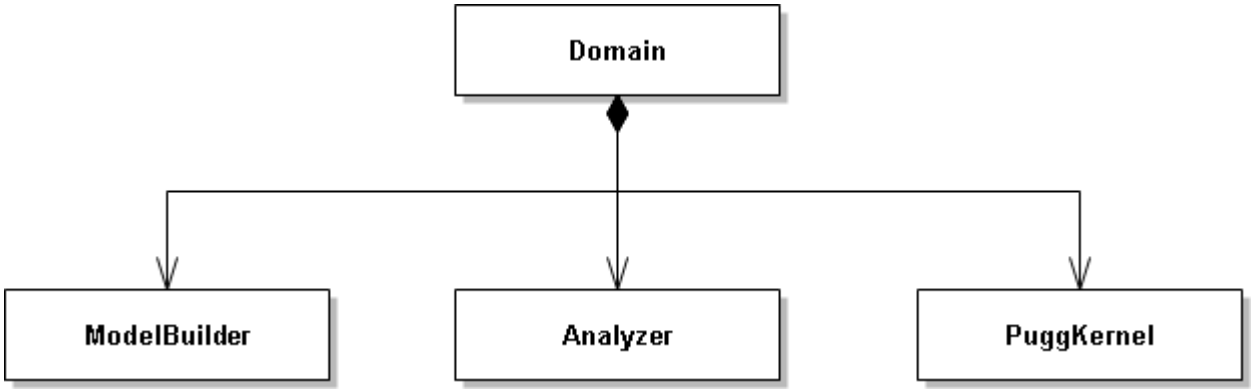
programlama imkanlarını kullanarak yüksek başarımlı, geliştirebilir bir çözümleme platformu oluşturmaktadır.

Panthalassa nesne yönelimli mimarinin gerektirdiği üzere hiyerarşik bir dizi sınıf kullanılarak tasarlanmıştır. Bu sınıflar kullanıcıdan giriş dosyasının okunması, bu dosyaya göre sonlu elemanlar yönteminin değişik öğelerine karşılık gelen farklı sınıfları içeren veri tabanının oluşturulması, daha sonra da istenen sonlu elemanlar algoritmasının bu veri tabanı üzerinde çalışmasını sağlar.

Nesne Yönelimli Veri Yapısı

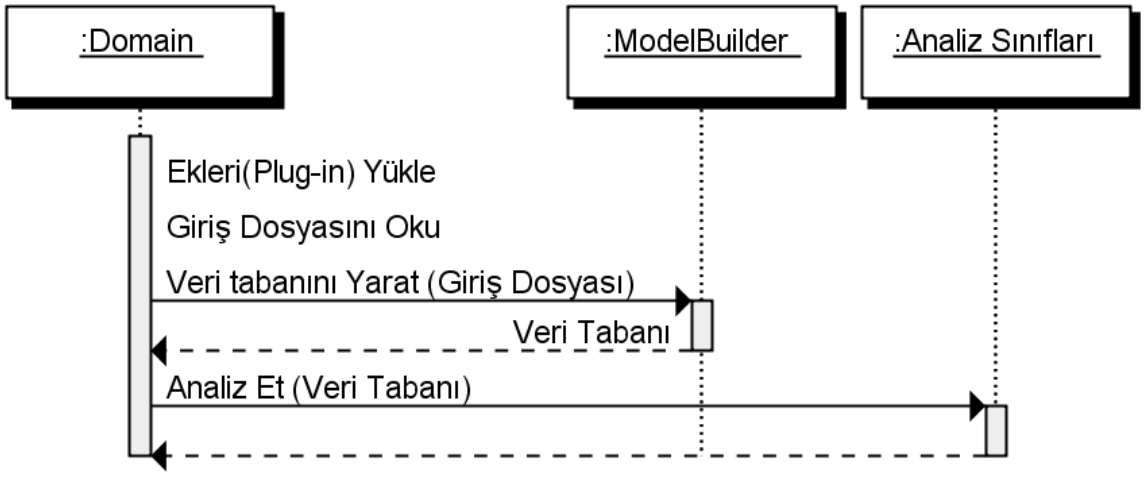
Domain Sınıfı

Domain Sınıfı kullanıcıdan sonlu elemanlar analizini tanımlayan giriş dosyasını alır ve bu dosyaya göre sonlu elemanlar analizini yürütür. *Domain* Sınıfı bu işlemler için Şekil 3.1'de gösterilen üç sınıftan yararlanır: *ModelBuilder*, *Analyzer* ve *PuggKernel*. *ModelBuilder* Sınıfı giriş dosyasının okunmasından, *Analyzer* Sınıfı sonlu elemanlar analizinden, *PuggKernel* Sınıfı ise Panthalassa'ya yeni özellikler ekleyen eklerin sisteme yüklenmesinden sorumludur.



Şekil 3.1: Domain Sınıf Diyagramı

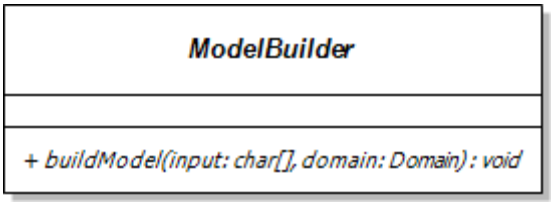
Domain Sınıfının işleyiş şekli Şekil 3.2'de gösterilmiştir. *Domain* Sınıfı ilk iş olarak ekleri *PuggKernel* Sınıfı vasıtasıyla sisteme yükler, bu şekilde kullanıcının Panthalassa'ya eklenenen yeni işlevlere ulaşabilmesini sağlar. Bu işlemden sonra *Domain* Sınıfı kullanıcı giriş dosyasını okur ve kullanıcının belirlediği *ModelBuilder nesnesini* kullanarak hafızada sonlu elemanlar modeli ve analizini simgeleyen veri tabanını oluşturur. Son olarak *Analyzer* Sınıfını kullanarak analizi gerçekleştirir.



Şekil 3.2: Panthalassa İşleyiş Düzeni

ModelBuilder Sınıfı

ModelBuilder Sınıfı (Şekil 3.3) kullanıcı giriş dosyasından sonlu elemanlar modeli ve analizi veri tabanının oluşturulmasından sorumludur. Soyut *ModelBuilder* Sınıfına işlerlik bu sınıftan türeyen sınıflar ile kazandırılır. *ModelBuilder* sınıfından türeyen her sınıf *buildModel* fonksiyonunu yeniler ve bu fonksiyonda sınıfa özel biçimdeki giriş dosyasından sonlu elemanlar modeli ve analizi veri tabanını oluşturur. Kullanıcı Panthalassa'ya parametre olarak giriş dosyasını okumasını istediği *ModelBuilder* Sınıfından türeyen bir sınıfın ismini vererek giriş dosyasına özel *ModelBuilder* nesnesinin kullanılmasını sağlar.



Şekil 3.3: ModelBuilder Sınıf Diyagramı

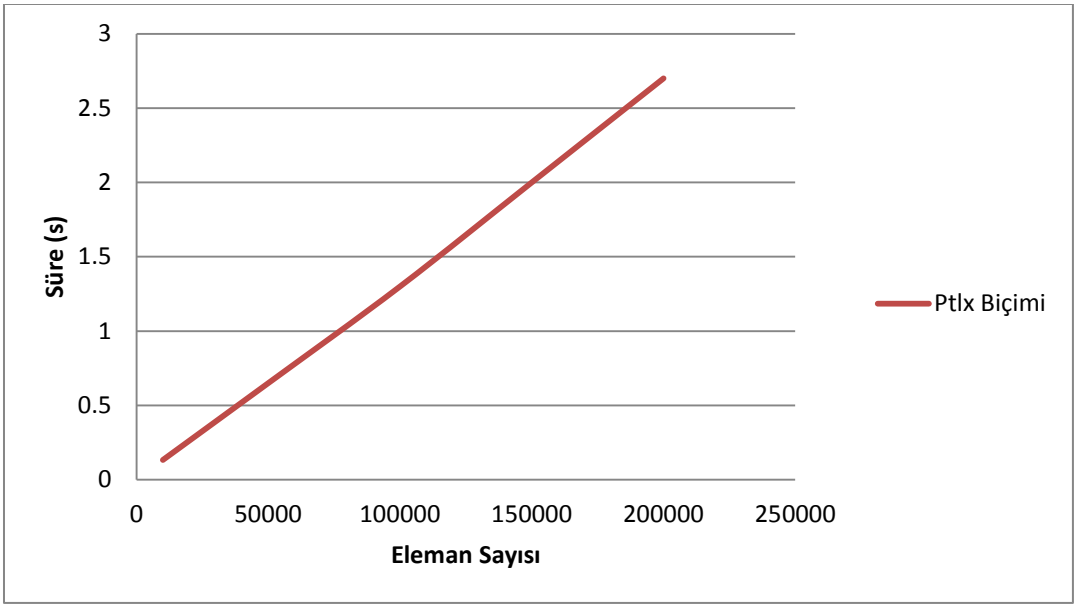
Ptlx Panthalassa'nın özel giriş dosyası biçimidir ve *PtlxBuilder* Sınıfıyla Panthalassa'ya uygulanmıştır. *PtlxBuilder* Sınıfı Ptlx biçimine uygun hazırlanmış dosyaları okur ve bu dosyadaki tanımlara göre sonlu elemanlar modeli ve analizi veri tabanını oluşturur. Ptlx giriş dosyası biçimi, XML (Extensible Markup Language: Genişletilebilir İşaretleme Dili) tabanlıdır ve veri tabanı oluşturma işleminin az bellek kullanarak hızlı bir biçimde gerçekleşmesi için geliştirilmiştir. XML dosya sisteminin okunması için RapidXml Kütüphanesi (2012) kullanılmıştır. RapidXml kütüphanesi XML dosyalarını ayrıştırırken bellekte dosyanın ikinci bir kopyasını oluşturmamakta, bu tekniğin yerine XML dosyasını oluşturan elemanlar ve bu elemanlara bağlanan özelliklerin dosya içindeki yerlerini belirlemeye dayanan bir teknik kullanmaktadır. Bu şekilde RapidXML az bellek kullanarak çok büyük XML dosyalarını hızlı bir şekilde ayrıştırabilmektedir. Şekil 3.4'de *Ptlx* biçimli değişik sayıda elemandan oluşan sonlu elemanlar modellerini betimleyen giriş dosyalarından sonlu elemanlar veri tabanı yapısı oluşturulması için gereken süreler sergilenmiştir. Panthalassa *Ptlx* biçimli 200,000 elemanlı bir sonlu elemanlar modeli giriş dosyasından sonlu elemanlar veri tabanını yaklaşık üç saniye içinde oluşturabilmektedir.

Liste 3.1'de *Ptlx* biçimli bir giriş dosyasından alınan örnek sergilenmiştir. Ptlx biçiminde kullanıcı tanımladığı nesnelere, bu nesnelere dair özel seçenekleri XML dosya formatının özellik (attribute) nesnelerini kullanarak atayabilir. Karakter dizisi olarak tanımlanan bu özellikler yeni sistemde nesnelerin yapıcı metodu kullanılarak nesnelere iletilir. Liste 3.1'de verilen örnekte *MaterialModel* nesnesinin parametreleri bu şekilde belirlenmiştir.

Analyzer Sınıfı

*Analyzer S*ınıfı sonlu elemanlar modelini ve sonlu elemanlar analizini simgeleyen sınıflardan sorumludur. *Analyzer* Sınıfı bu sınıfların nesnelerini sahiplenir ve işleyişlerini kontrol eder. Şekil 3.5'de *Analyzer* Sınıf diyagramı sergilenmiştir. *Analyzer* Sınıfına bağlı olan sınıflardan *Structure* sınıfı sonlu elemanlar modeli veri yapısını içinde barındırır. *MaterialModel* Sınıfı sonlu elemanlar modelinde kullanılan malzemeleri simgeler. *TimeTable* Sınıfı analiz süresince zamanın değişimini kontrol eder. *Algorithm* Sınıfı analiz sırasında kullanılan algoritmadan, *PartitioningJob* Sınıfı ise analiz sırasında kullanılabilecek olan bölümleme işlemlerinden sorumludur. Son olarak *Tracker* Sınıfı analiz sonucunda oluşan çıktıların kullanıcıya aktarımını sağlar.

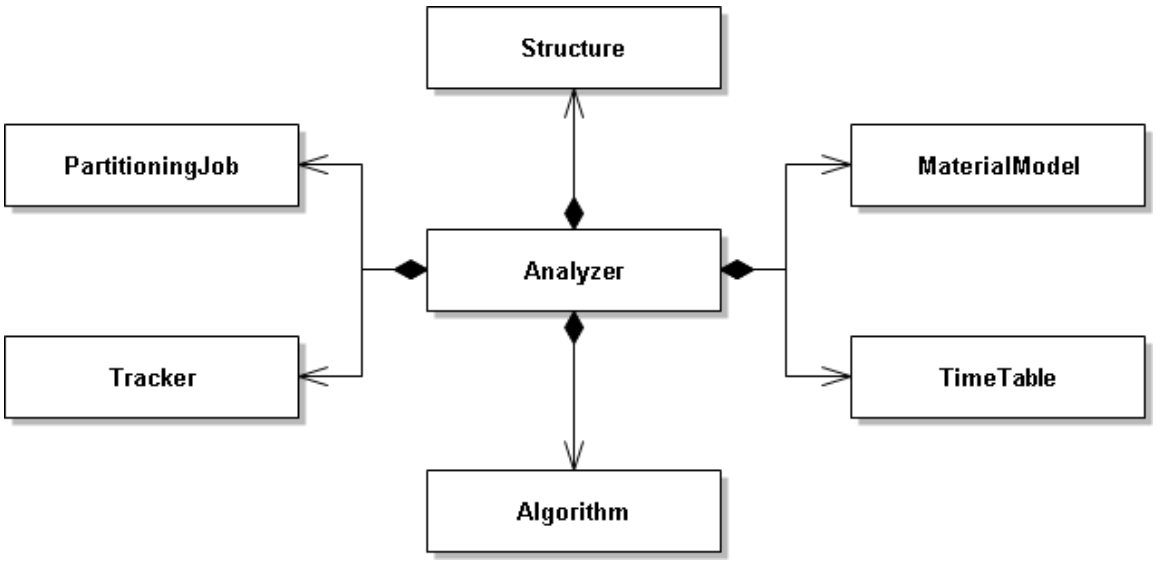
Domain, Analyzer nesnesine analizi başlatma emrini verir. *Analyzer* gerekiyorsa *PartitioningJob* nesnesi ile sonlu elemanlar modelini bölümledikten sonra kullanıcının giriş dosyasında belirlediği ve *TimeTable* nesnesinde tutulan zaman aralıklarının her biri için, *Algorithm* nesnesi ile analizin yürütülmesini *Tracker* nesneleri ile analiz sonuçlarının kaydedilmesini sağlar. Bu işlemler sırasında sonlu elemanlar modelini simgeleyen *Structure* nesnesi ve modelde kullanılan materyal modellerini simgeleyen *MaterialModel* nesneleri kullanılır.



Şekil 3.4: Ptlx Biçimli Giriş Dosyasından Sonlu Elemanlar Veri Tabanı Yapısı Oluşturulması İşlemi için Performans Testi

Liste 3.1: Ptlx Giriş Dosyası Okuma Biçimi Örneği

```
<MaterialModel plugin="linear_material" id="1" E="25e6" Mu=”0.3” p=”100”/>
<Structure id="0">
<Nodes>
<Node id="1" x="0" y="0" z="4" />
<Node id="2" x="4" y="0" z="4" />
<Node id="3" x="4" y="4” z="4" />
<Node id="4" x="0" y="4" z="4" />
<Node id="5" x="0" y="0" z="0" />
<Node id="6" x="4" y="0" z="0" />
<Node id="7" x="4" y="4” z="0" />
<Node id="8" x="0" y="4" z="0" />
</Nodes>
<Elements plugin="brick" material="1">
<Element id="1" nodes="1,2,3,4,5,6,7,8" />
</Elements>
...
```



Şekil 3.5: Analyzer Sınıf Diyagramı

Structure Sınıfı

Structure Sınıfı sonlu elemanlar çözümlemesinde kullanılacak olan modeli simgeler. Panthalassa'da sonlu elemanlar modelinin değişik öğeleri (düğüm noktaları, elemanlar vb.) değişik sınıflarla simgelenmiştir. Bu sınıflara ait nesneler *Structure* nesnesinde tutulurlar. Şekil 3.6'da *Structure* Sınıf yapısı sergilenmiştir. Sonlu elemanlar modeli nesnelerini simgeleyen sınıflar aşağıda detaylandırılmıştır.

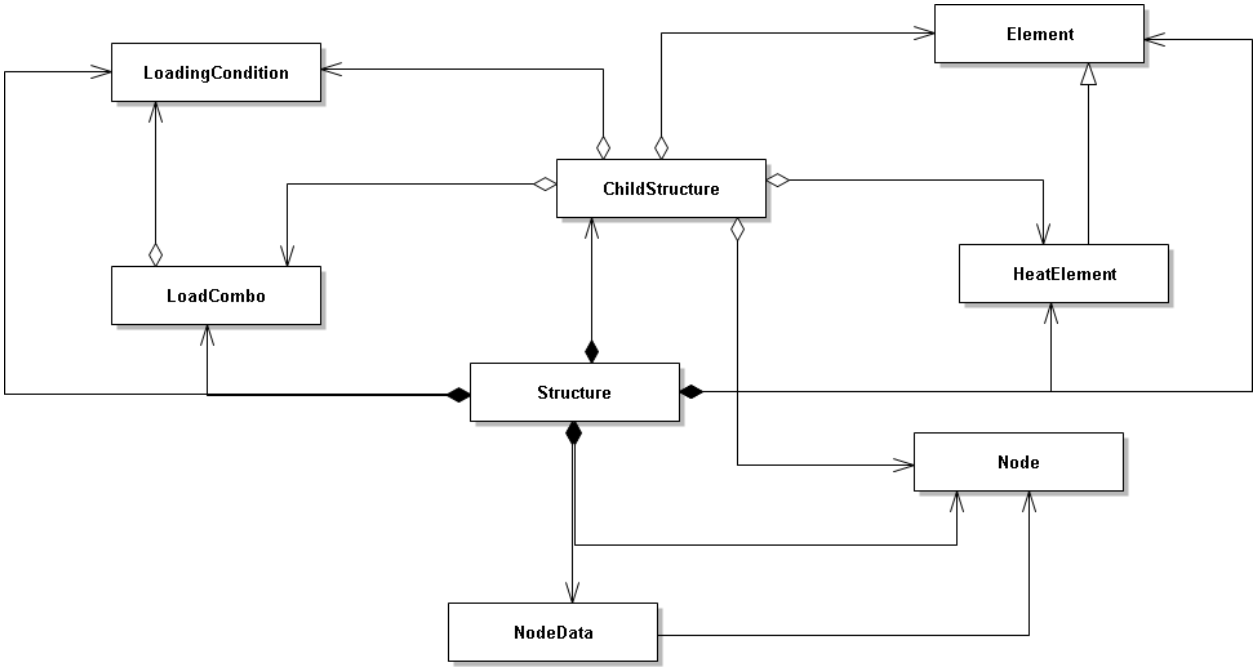
Node Sınıfı

Node Sınıfı sonlu elemanlar modelindeki düğüm noktalarını simgeler. Bu düğüm noktaları uzaydaki belli koordinatlarla ilişkilendirilir ve bu şekilde modelin ve model içindeki sonlu elemanların uzaydaki yeri belirlenir.

NodeDataHolder Sınıfı

NodeDataHolder Sınıfı sonlu elemanlar modelinde düğüm noktaları ile eşleştirilen bilgileri hafızada tutmaktan sorumludur. *NodeDataHolder* Sınıfının bellekte tuttuğu bilgiler düğüm noktalarındaki deplasman, hız, ivme, sıcaklık bilgileri ve her düğüm noktasının serbestlik dereceleri ile eşleştirilen denklem numaralarıdır.

Bu bilgilerin *Node* Sınıfında değil *NodeDataHolder* Sınıfında toplanmasının sebebi analiz sırasında işlemcinin önbellek performansını arttırmaktır. İşlemci hafızadaki bilgilere ulaşırken çalışan programın istediği bilgiye ek olarak hafızada bu bilgiye yakın olarak tutulan bilgileri de önbelleğe atar. Önbelleğin erişim süresi normal hafızaya göre çok daha kısa olduğu için programlar hafızada birbirine yakın bilgilere art arda erişmek için tasarlanırlarsa önbellek performansının düşünülmediği tasarımlara göre yüksek performansa ulaşılabilir. Şekil 3.7'de düğüm noktaları ile eşleştirilen bilgilerin ayrı *Node* nesnelerinde ve tek bir *NodeDataHolder* nesnesinde tutulması arasındaki farkı gösterir. Analiz sırasında farklı *Node* nesnelerine ait bilgilere art arda erişilmesi gerekeceği için bu bilgilerin bir arada tutulması bu bilgilere erişim performansını artırır.



Şekil 3.6: Structure Sınıf Diyagramı

Element ve *HeatElement* Sınıfları

Sonlu elemanlar yönteminin kalbinde elemanlar olarak tabir edilen ve algoritmada kullanılacak olan matrislerin oluşturulmasını sağlayan birimler yatar. Panthalassa'da bu birimlerin karşılığı *Element* Sınıfıdır (Şekil 3.8). *Element* Sınıfı kullanıcıdan kendisine verilen koordinatlar ve çeşitli özellikleri kullanarak algoritmada kullanılacak olan direngenlik, kütle vb. matrislerini oluşturur, eşdeğer düğüm noktası yüklerini ve analiz sırası ve sonrasında gerekli olan gerinim (strain) ve gerilim (stress) gibi çıkış bilgilerini hesaplar. *Element* Sınıfı kullanıcıdan koordinat bilgilerini *Node* nesneleriyle (*nodes* Değişkeni) ve elemanın kullandığı malzeme özelilerini de kendisine atanan *MaterialModel* nesnesinden (*materialModel* Değişkeni) alır.

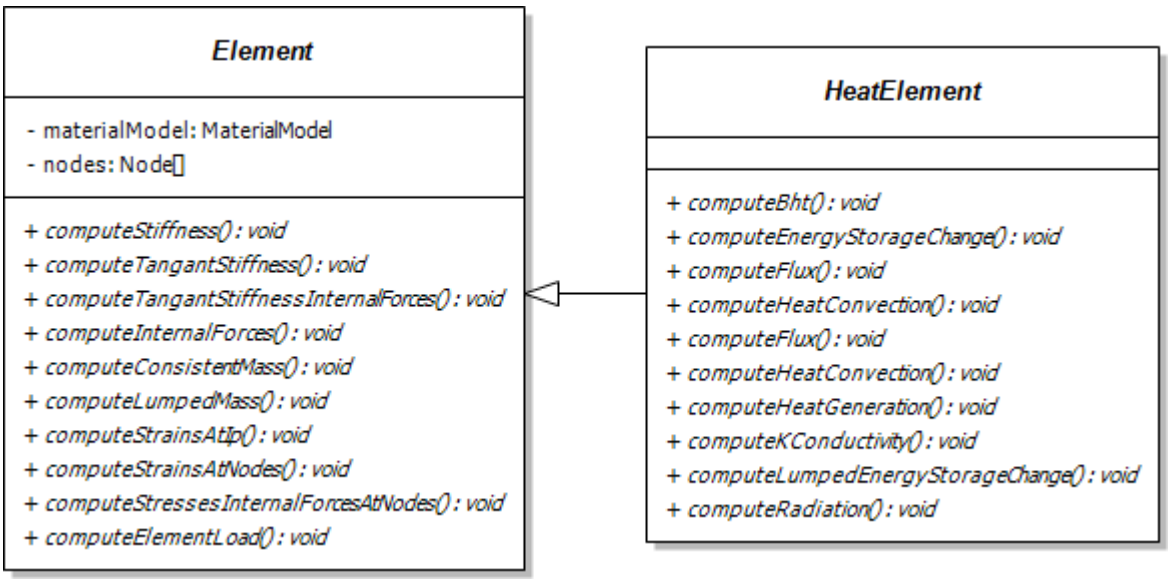
Node 1	Deplasman
	Hız
	İvme
	...
Node 2	Deplasman
	Hız
	İvme
	...
Node 3	Deplasman
	Hız
	İvme
	...
...	

(a)

Node 1	Deplasman	Hız	İvme
Node 2	Deplasman	Hız	İvme
Node 3	Deplasman	Hız	İvme
...	...	...	...

(b)

Şekil 3.7: Node Nesneleri ile Eşleştirilen Bilgilerin Ayrı *Node* Nesnelerinde (a) ve Tek Bir *NodeDataHolder* Nesnesinde (b) Tutulmasının Oluşturduğu Hafıza Şablonları



Şekil 3.8: *Element* ve *HeatElement* Sınıf Diyagramları

Şekil 3.8'de gösterildiği üzere *Element* Sınıfı sanal bir sınıftır ve elemanların analiz sırasında kullanılması için gereken fonksiyonları belirtir. *Element* Sınıfından türeyen sınıflar bu fonksiyonları uygular ve gerekli işlerliği sağlarlar. Tablo 3.1'de bu sanal fonksiyonlar açıklanmıştır. Element Sınıfından türeyen sınıflar ancak mekanik analizlerde kullanılabilirler. Isı analizlerinde kullanılacak elemanlar *HeatElement* sınıfından türerler. *HeatElement* sınıfı *Element* sınıfından türetilmiştir ve *Element* Sınıfının içerdiği sanal fonksiyonlara ek olarak ısı analizi ile ilgili sanal fonksiyonları da içerir. *HeatElement*

Sınıfından türeyen sınıflar bu sanal fonksiyonları uygular ve ısı analizi ile ilgili işlevliği sağlarlar. *HeatElement* sınıfının özel fonksiyonları Tablo 3.2'de açıklanmıştır.

Tablo 3.1: Element Sınıfı Sanal Fonksiyonları

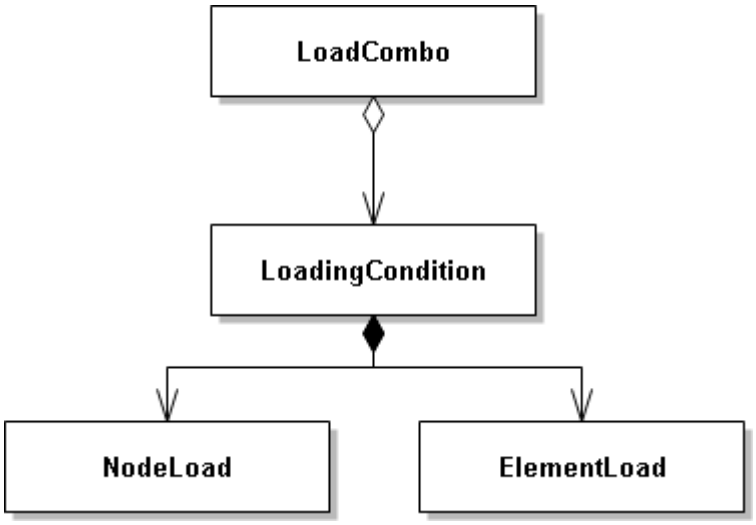
Fonksiyon	Açıklama
computeStiffness	Direngenlik matrisini hesapla
ComputeTangentStiffness	Teğet direngenlik matrisini esapla
computeTangentStiffnessInternalForces	Teğet direngenlik matrisi ve iç kuvvetleri hesapla
computeInternalForces	İç kuvvetleri hesapla
computeConsistentMass	Tutarlı kütle matrisini hesapla
computeLumpedMass	Topaklanmış kütle matrisini hesapla
comnputeStrainsIp	Gerilimleri tümleştirme noktalarında hesapla
computeStrainsAtNodes	Gerinimleri düğüm noktalarında hesapla
computeStressesInternalForcesAtNodes	Gerilimleri ve İç kuvvetleri düğüm noktalarında hesapla
computeElementLoad	Eleman kuvvetlerini hesapla

Tablo 3.2: HeatElement Sınıfı Sanal Fonksiyonları

Fonksiyon	Açıklama
computeKConductivity	Kondüksiyon matrisini hesapla
computeFlux	Yüzey akımını hesapla
computeHeatConvection	Konveksiyon matrisi ve yük vektörünü hesapla
computeHeatGeneration	Isı üretimini hesapla
computeEnergyStorageChange	Tutarlı ısı kapasitesi matrisini hesapla
computeLumpedEnergyStorageChange	Topaklanmış ısı kapasitesi matrisini hesapla
computeBht	Isı gradient matrisini hesapla

LoadingCondition Sınıfı

Panthalassa'da sonlu elemanlar modeline uygulanan yükler değişik şekillerde gruplandırılabilir ve bu gruplara istenen katsayılar verilerek yük kombinasyonları uygulanabilir. *LoadingCondition* Sınıfı sonlu elemanlar modeline uygulanan yük kombinasyonlarını simgeler (Şekil 3.9).



Şekil 3.9: Panthalassa'da Sonlu Elemanlar Modeline Uygulanan Yükleri Simgeleyen Sınıflar

Panthalassa'da iki yük tipi vardır. Sonlu elemanlar modelindeki düğüm noktalarına uygulanan yükler *NodeLoad* Sınıfı ile sonlu elemanlar modelindeki elemanlara uygulanan yükler *ElementLoad* Sınıfı ile simgelenir.

NodeLoad Sınıfı

Panthalassa'da düğüm noktalarına gelen dış etkiler *NodeLoad* Sınıfı ile simgelenir. *NodeLoad* Sınıfı ile simgelenebilen yükler *LoadType* isimli bir listede (enumeration) tutulmaktadır. Bu listede tutulan yük tipleri Tablo 3.3'de gösterilmiştir. *NodeLoad* sınıfı belirtilen etki tipinin büyüklüğünü zamana bağlı olarak tutabilir ve büyüklükler her yönde farklı olarak tanımlanabilir.

Tablo 3.3: *LoadType* Listesi Seçenekleri

Seçenek	**Açıklama**
Force	Kuvvet
Displacement	Deplasman
Velocity	Hız
Acceleration	İvme
Damper	Sönümleme
Restraint	Sınırlama

ElementLoad Sınıfı

Panthalassa'da sonlu elemanlara verilen yükler *ElementLoad* Sınıfı ile simgelenir. *ElementLoad* Sınıfı kuvvet, ısı, deplasman gibi yüklemeleri zamana göre simgeleyebilen genel bir sınıftır. Elemanlara uygulanan hacim ve yüzey yüklemeleri bu sınıf aracılığıyla tanımlanır.

LoadCombo Sınıfı

LoadCombo Sınıfı *LoadingCondition* nesneleri ile tanımlanan birden fazla yüklemenin birleşimini içerir. *LoadCombo* Sınıfı aracılığı ile yüklemeler kullanıcının belirleyeceği katsayılar kullanılarak doğrusal şekilde birleştirilip sonlu elemanlar modeline uygulanabilir.

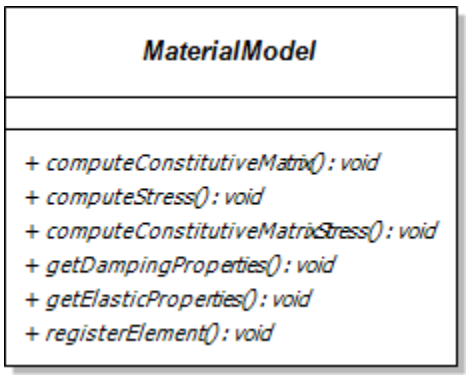
ChildStructure Sınıfı

Bölümleme paralel sonlu elemanlar analizinde analiz yükünü bilgisayarlar arasında paylaştırmak için kullanılan bir işlemdir. Bölümleme işlemi sonlu elemanlar modelini birden fazla alt-yapıya ayırır. Panthalassa'da bu bölümler *ChildStructure* Sınıfı ile simgelenir. *ChildStructure* Sınıfı simgelediği bölüme ait *Element*, *Node* ve yükleme sınıflarına sahip değildir sadece bu nesnelere ait işaretçileri (pointer) içerir. Bu şekilde sonlu elemanlar modeline ait nesneler kopyalama işlemene gerek olmadan bölümlere ayrılabilir.

MaterialModel Sınıfı

MaterialModel Sınıfı sonlu elemanların malzeme özelliklerinden sorumludur. Her eleman kullanıcının belirlediği bir *MaterialModel* nesnesine bağlanır ve kendisinden istenen matrisleri yaratırken bu *MaterialModel* nesnesinden aldığı malzeme özelliklerini kullanır. *MaterialModel* Sınıfı kendisine bağlı elemanlardan gerinim (strain) ve gerilim (stress) gibi bilgileri algoritmanın yürütülmesi sırasında alabilir ve bunlara bağlı kendisine özel işlemleri yapabilir. *Element* ve *MaterialModel* nesneleri arasındaki bu ilişki doğrusal veya doğrusal olmayan, örneğin akma modelleri, gibi malzeme özelliklerini anlatmakta kullanılır.

Şekil 3.10 Panthalassa'ya yeni malzeme modellerinin eklenmesi için yenilenmeleri gereken *MaterialModel* fonksiyonlarını gösterir. Bu fonksiyonlardan *computeConstitutiveMatrix* gerinim-gerilim ilişkisi (constitutive relationship) matrisinin hesaplanmasını, *computeStress* gerilim hesaplanması, *computeConstitutiveMatrixStress* ise tek bir operasyonda hem constitutive matrisinin hem de gerilimlerin hesaplanmasın sağlar. *getDampingProperties* ve *getElasticProperties* fonksiyonlarında malzemeye ait sönümleme ve elastik malzeme özellikleri geri dönülür. *registerElement* fonksiyonu ise analiz başlangıcında malzeme modelinin ilişkilendirildiği tüm elemanlar için *Analyzer* nesnesi tarafından çağrılır ve *MaterialModel* nesnesinin *Element* nesneleri ile bağ kurması sağlanır.



Şekil 3.10: *MaterialModel* Sınıf Diyagramı

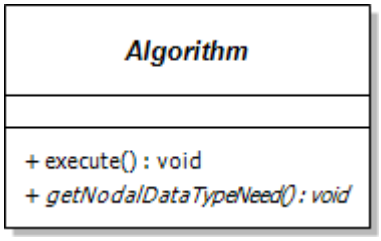
TimeTable Sınıfı

TimeTable Sınıfı bir veya birden fazla *TimeLine* nesnesi kullanarak çözümleme süresinin ve çözümleme sırasında kullanılan zaman adımlarının tanımlanmasını sağlar. *TimeLine* Sınıfı çözümleme sırasında çeşitli sebeplerle tutulan zaman bilgisini kontrol etmekle yükümlüdür. Örneğin dinamik çözümleme sırasında zaman aralığını bu sınıf kontrol eder. Bu şekilde çözümleme sırasında farklı süreler boyunca farklı zaman adımları kullanılabilir.

Algorithm Sınıfı

Algorithm Sınıfı sonlu elemanlar yönteminde kullanılan çözümlemelerdeki matematiksel işlemleri içeren kısmı simgeler. *Algorithm* Sınıfı sonlu elemanlar modelini simgeleyen nesneleri kullanarak analizi yürütür ve analiz sonuçlarını *NodeDataHolder* nesnesinde tutulan veri yapılarına yazar. Panthalassa algoritma yardımcıları adı altında değişik algoritmalarda kullanılabilecek olan işlemleri içeren bir dizi fonksiyon da içermektedir.

Panthalassa'ya yukarıda belirtilen işlemleri yeni algoritmalar için uygulayan algoritmalar eklenebilir. Bu işlem *Algorithm* Sınıfından (Şekil 3.11) türeyen yeni sınıflar ile gerçekleştirilir. Bu sınıflar analiz sırasında her zaman adımında çağrılan ve algoritmayı yürütmekten sorumlu *execute* ve algoritmanın kullandığı *Node* nesneleri ile ilişkilendirilen bilgi tiplerini *Analyzer* sınıfına veren *getNodalDataTypeNeed* fonksiyonlarını yenileyerek kendilerine özgü algoritmaları Panthalassa'ya eklerler.



Şekil 3.11: *Algoritm* Sınıf Diyagramı

Algoritma Yardımcıları

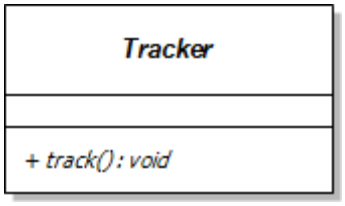
Algoritma yardımcıları değişik matris tipleriyle Panthalassa sınıflarının fonksiyonları arasında bağlantı kurmak ve algoritmaların yazımı sırasında ortaya sıkça çıkan matris birleştirme gibi işlemlerin kolaylaştırması için kullanılan C++11 standardının C++ diline getirdiği yeni bir özellik olan isimsiz fonksiyonları (lambda functions) kullanan bir grup fonksiyondur. Bu fonksiyonlar parametre olarak isimsiz birer fonksiyon almaktadır. Bu isimsiz fonksiyon algoritmada kullanılan özel matris tipi ile Panthalassa veri yapısı arasındaki bağlantıyı kurar. Liste 3.2 isimsiz fonksiyonların Panthalassa'da kullanımına bir örnek gösterir. Bu örnekte kullanılan *assembleStiffness* fonksiyonu bir *Structure* nesnesinde tutulan tüm elemanları algoritmanın sistem direngenlik matrisinde birleştirir. Algoritmanın kullandığı matris tipi değişebileceği için kodun o kısmı isimsiz bir fonksiyon içinde algoritma yazarı tarafından fonksiyona parametre olarak verilir. Derleyici algoritma kodunda *assembleStiffness* fonksiyonu ve isimsiz fonksiyondan gelen kodu birleştirir, böylece herhangi bir performans kaybı olmadan direngenlik matrisi birleştirme işlemi tamamlanır.

Liste 3.2: Panthalassa'da İsimsiz Fonksiyon Kullanım Örneği

```
ptl::assembleStiffness (structure->elements.begin(), structure->elements.end(), [&](int x, int y, double val)
{
        k(x,y) += val;
});
```

Tracker Sınıfı

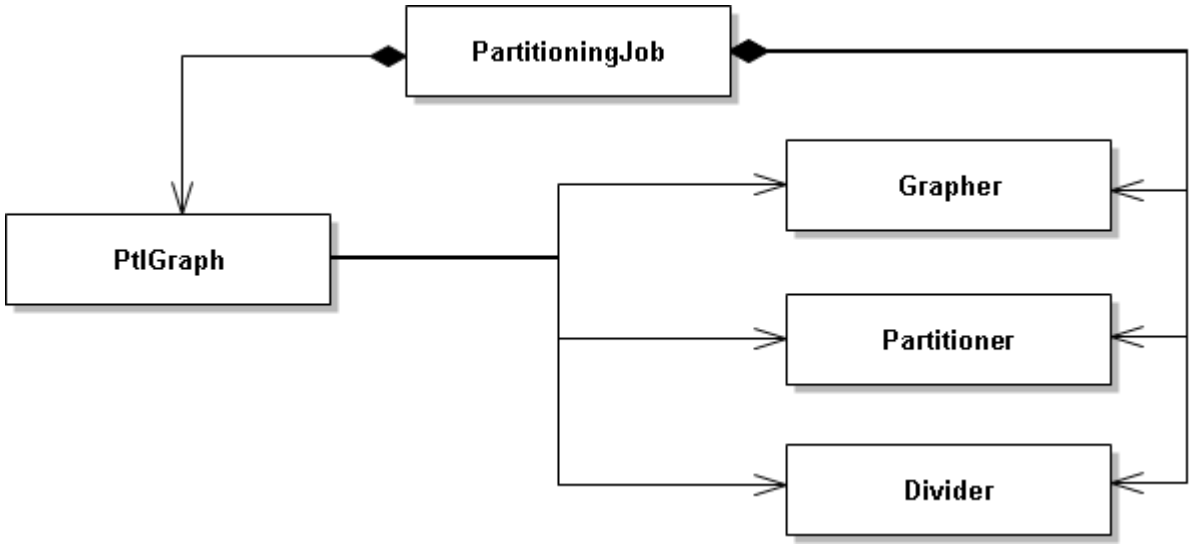
Tracker Sınıfı çözümleme sonuçlarının dosyalanarak saklanmasından sorumludur. *Tracker* Sınıfı *NodeDataHolder* Sınıfında toplanan analiz sonuçlarını dosyaya yazarak kullanıcıya bildirir. Sanal bir sınıf olan *Tracker* sınıfından (Şekil 3.11) türeyen sınıflar analiz sonuçlarını değişik dosya biçimlerinde saklanmasında kullanılırlar. *Analyzer* analizin her zaman adımında *Tracker* nesnelerinin *track* fonksiyonunu çağırır. *Tracker* Sınıfından türeyen sınıflar bu fonksiyonu yeniler ve analiz sonuçlarının değişik şekillerde kaydedilmesinin sağlarlar.



Şekil 3.12: *Tracker* Sınıf Diyagramı

PartitioningJob Sınıfı

PartitioningJob Sınıfı sonlu elemanlar modelinde uygulanacak olan bir bölümleme işlemini tanımlar. Bölümleme işlemi çözümlemenin herhangi bir anında uygulanabilecek şekilde tanımlanabilir. *PartitioningJob* Sınıfı ve bu sınıfın kapsadığı *Grapher*, *Partitioner* ve *Divider* nesnelerinin (Şekil 3.13) tüm özelliklerini kullanıcı belirlemektedir. *Algorithm* Sınıfı kullanıcının belirlediği *PartitioningJob* Sınıfını kullanarak bölümleme içini gerçekleştirir. Bu işlem analiz yükünün değiştiği durumlarda tekrarlanarak her bilgisayara eşit analiz yükü sağlanır.



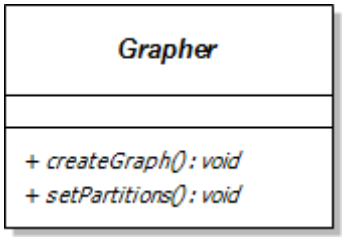
Şekil 3.13: *PartitioningJob* Sınıf Diyagramı

PtlGraph Sınıfı

PtlGraph Sınıfı genel bir grafiksel görünümü (graph represantation) simgeler. *PtlGraph* sınıfı grafiksel gösterim yapısı için Boost Graph (2012) kütüphanesinin kullanır. *PtlGraph* sınıfı Boost Graph kütüphanesinin sağladığı veri yapılarına ek olarak grafiksel gösterimin sonlu eleman modeli ile olan bağlantılarını tutan ve grafiksel gösterimin bölümlenmesi sırasında yardımcı olan ek veri yapıları da içerir.

Grapher Sınıfı

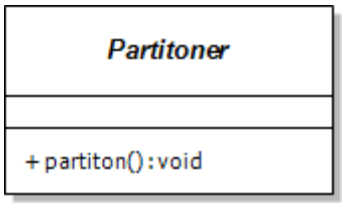
Grapher Sınıfı *Structure* nesnelerinden grafiksel gösterimler oluşturmaya yarayan bir sanal sınıftır. *Grapher* Sınıfı sanal bir sınıftır, kullanıcılar değişik grafiksel gösterim tipleri için alt sınıflar türeterek algoritmalarını uygulayabilirler. Bu algoritmalar *Grapher* Sınıfından (Şekil 3.14) türeyen yeni sınıflar ile uygulanır. Grapher sınıfının iki sanal fonksiyonu vardır: *createGraph* ve *setParititons*. *createGraph* Fonksiyonu *Structure* nesnesinden grafiksel gösterimi oluşturur. *setPartitions* Fonksiyonunda ise bölümleme işlemi sonrası oluşan parçalanmış grafiksel gösterime göre *Structure* nesnesinin *Element* ve Node nesnelerini ilişkilendirildikleri bölümleri atar.



Şekil 3.14: *Grapher* Sınıf Diyagramı

Partitioner Sınıfı

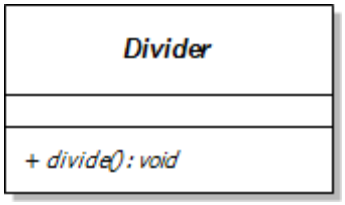
Partitioner sınıfı (Şekil 3.15) grafiksel gösterimleri bölümlemekle sorumlu bir sanal sınıftır. Bölümleme algoritmalarını uygulayan ve *Partitioner* Sınıfından türeyen yeni sınıflar *partition* isimli sanal fonksiyonu yenileyerek kendi bölümleme algoritmalarını uygulayabilirler. Bu fonksiyon grafiksel gösterimi bölümlemek ve grafiksel gösterimin köşe ve kenarlarına ait oldukları bölümü etiketlemekle sorumludur.



Şekil 3.15: *Partitioner* Sınıf Diyagramı

Divider Sınıfı

Grapher ve *Partitioner* sınıfları kullanılarak bölümlenmiş sonlu elemanlar modelinin her parçasından uygun *ChildStructure* nesnelerinin yaratılmasını sağlamak için *Divider* Sınıfı (Şekil 3.16) tanımlanmıştır. Panthalassa'ya genel durumlar için kullanılabilecek *Divider* sınıfından türeyen *PtlDivider* sınıfı da eklenmiştir.



Şekil 3.16: *Grapher* Sınıf Diyagramı

Plug-in Alt Yapısı

Panthalassa'ya Yeni Özelliklerin Eklenmesi

Nesne yönelimli mimarinin gereği üzere Panthalassa'ya ek olarak getirilecek olan yeni özellikler Panthalassa'da tanımlanan soyut sınıflardan türeyen yeni sınıflarla olur. Kullanıcılar soyut sınıflardan türeyen yeni sınıflar yazar ve bunları dinamik olarak yüklenen bilgisayar kütüphanelerinde toplarlar. Bu kütüphaneler Panthalassa için özel olarak yazılmış Pugg Kütüphanesi tarafından sisteme otomatik olarak yüklenir.

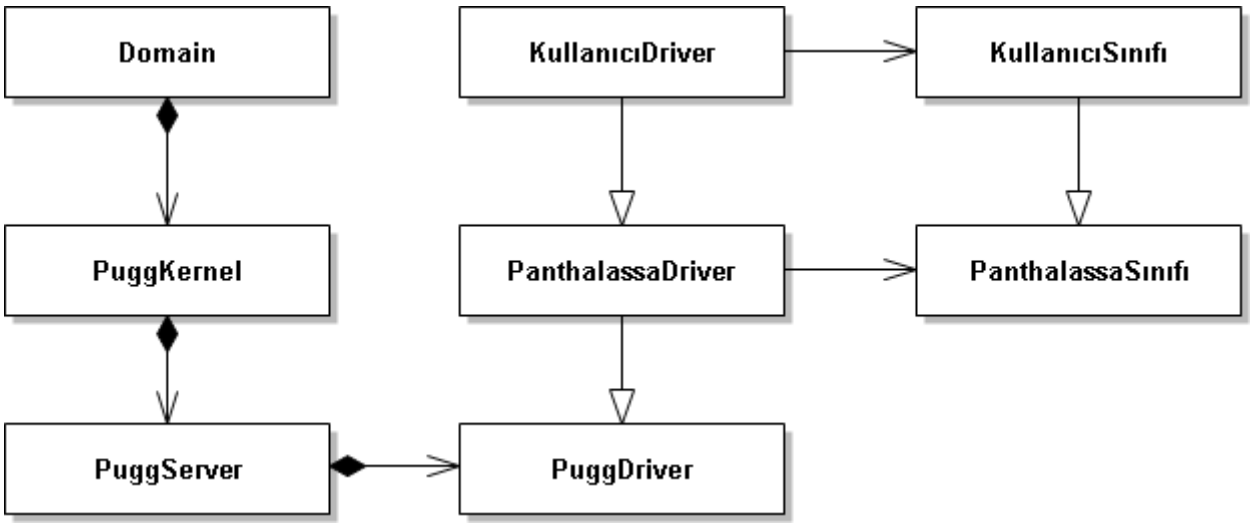
Pugg Kütüphanesi

Panthalassa'nın ekler kullanılarak geliştirilmesi için Pugg isimli kütüphane geliştirilmiştir (Şekil 3.17). Pugg kullanıcıların yazdığı eklerin standartlaştırılması ve bir sürüm sistemiyle, Panthalassa'nın güncel durumuna uymayan eklerin çalışmasını önler.

Kullanıcılar yazdıkları her yeni sınıf için *Driver* denilen bir ek sınıf daha yazarlar. *Driver* sınıfları bağlı oldukları sınıfın sürüm bilgisini tutar ve bağlı oldukları sınıftan yeni bir nesne oluşturmakla görevli bir fonksiyona sahiptirler. Panthalassa'da ekleme sistemi ile türetilebilecek her soyut sınıf için bir adet de soyut *Driver* sınıfı oluşturulmuştur. Kullanıcılar kendi ek sınıfları için yarattıkları *Driver* sınıfları ilgili soyut *Driver* Sınıfından türetirler. Panthalassa önce Pugg yardımı ile *Driver* nesnesine ulaşır ve *Driver* nesnesini kullanarak kullanıcının oluşturduğu ek sınıftan bir nesne yaratır. Her *Driver* sınıfı bağlı olduğu ek sınıfla ilgili bir de isim taşır. Kullanıcılar giriş dosyasında bu ismi kullanarak çözümleme sırasında hangi sınıfları kullanacaklarını belirler.

Pugg Kütüphanesinde *Driver* nesnelerinin kontrolü için üç ayrı sınıf yazılmıştır: *PuggServer*, *PuggKernel* ve *PuggPlugin*. *PuggServer* Sınıfı aynı soyut sınıflara bağlanan *Driver* nesnelerini bellekte tutan bir veri tabanıdır. Örnek olarak *Element* soyut sınıfından türeyen tüm sınıflara bağlı olan *Driver* nesneleri ayrı bir

PuggServer sınıfında tutulur. *PuggKernel* sınıfı değişik *PuggServer* nesnelerini kontrol eden bir veri tabanı olarak iş görür. *PuggPlugin* Sınıfı dinamik olarak yüklenen kütüphanelerden *Driver* sınıflarını otomatik olarak yükler ve bu sınıflardan yarattığı nesneleri uygun *PuggServer* nesnelerine kaydeder.



Şekil 3.17: Panthassa Ek (Plug-in) Yapısı

Panthalassa'da Plug-in Mantığıyla Yeni Sınıflar Türetilebilen Soyut Sınıflar

Panthalassa'da birçok soyut sınıftan Plug-in mantığıyla yeni sınıflar türetilebilir ve Panthalassa'ya dinamik kütüphaneler yardımı ile yüklenebilir. Tablo 3.4'de belirtilen sınıfların değişik uygulamaları bu bölümde anlatılan plug-in mantığı ile Panthalassa'ya eklenmiştir.

Tablo 3.4: Yeni Sınıflar Türetebilen Soyut Sınıflar

Soyut Sınıf
Element
HeatElement
MaterialModel
ModelBuilder
Algorithm
Tracker
Grapher
Partitioner
Divider

Element ve *HeatElement* sınıflarından türeyen ve değişik sonlu elemanlar elemanlarının uygulaması olup Panthalassa'ya eklenen sınıflar Tablo 3.5'de gösterilmiştir. Bu elemanların özellikleri ayrıntılı olarak Bölüm 2'de işlenmiştir.

Tablo 3.5: Panthalassa'ya eklenen sonlu eleman sınıfları

Sınıf	**Açıklama**
Truss	Makas Elemanı
Beam	Kiriş Elemanı
ShellQ	Dörtgen Kabuk Elemanı
ShellT	Üçgen Kabuk Elemanı
RecM4	Dörtgen Isı Elemanı
RecM8	Dörtgen Isı Elemanı
TriM3	Üçgen Isı Elemanı
TriM6	Üçgen Isı Elemanı
Brick8	Dörtgen Prizma Elemanı
Brick20	Dörtgen Prizma Elemanı
Wedge6	Üçgen Prizma Elemanı
Wedge15	Üçgen Prizma Elemanı
Tet4	Düzgün Dört Yüzlü Elemanı
Tet10	Düzgün Dört Yüzlü Elemanı

Yine Bölüm 2'de işlenen malzeme modeli kütüphanesinin parçaları olan doğrusal elastik malzeme modeli, elastoplastik malzeme modeli ve eşdeğer doğrusal malzeme modeli plug-in mantığı ile Panthalassa'ya eklenmiştir. Bu malzeme modellerinin uygulamaları olan ve *MaterialModel* Sınıfından türeyen sınıflar kullanılarak plug-inler oluşturulmuştur.

Panthalassa'nın kendi giriş dosyası biçimi olan *ptlx* dosyaları ModelBuilder sınıfından türetilen PtlxBuilder Sınıfı kullanılarak plug-in mantığı ile Panthalassa'ya eklenmiştir. Plug-in mantığı ile Panthalassa'ya eklenen *S2kReader* Sınıfı ise değişik sonlu elemanlar yazılımlarında kullanılan bir giriş dosyası biçimi olan S2k biçimli dosyaları okur.

Panthalassa'ya eklenen tüm sonlu elemanlar algoritmaları *Algorithm* Sınıfından türeyen sınıflar ile uygulanmışlardır. Bu sınıflar Tablo 3.6'da sıralanmıştır. Bu sınıfların ayrıntıları bu raporun ilgili bölümlerinde bulunabilir.

Tablo 3.6: Panthalassa'ya eklenen Plug-in Mantığıyla Eklenen Çözümleme Algoritmaları

Sınıf	**Açıklama**
SimpleStatic	Doğrusal Statik Çözümleme
NonlinearStatic	Doğrusal Olmayan Statik Çözümleme
ImplicitNewmark	Dinamik Belirtik Çözümleme
ExplicitNewmark	Dinamik Örtük Çözümleme
MumpsStatic	Paralel Toptan Statik Çözümleme
SubstructuredStatic	Paralel Alt-Yapı Tabanlı Statik Çözümleme
DistributedImplicitNewmark	Paralel Toptan Dinamik Belirtik Çözümleme
SubstructuredImplicitNewmark	Paralel Alt-Yapı Tabanlı Dinamik Belirtik Çözümleme
DistributedExplicitNewmark	Paralel Dinamik Örtük Çözümleme
Engpu	Grafik Kartlarında Çalışan Dinamik Örtük Çözümleme

Panthalassa'ya *Tracker* Sınıfından türeyen ve plug-in mantığıyla Panthalassa'ya eklenmiş üç sınıf vardır: *NodeTracker*, *ElementTracker* ve *VtkTracker*. Bu sınıflardan *NodeTracker* kullanıcı tarafından belirlenen Node nesnelerine ait bilgileri (deplasman, hız, ivme, sıcaklık vb.) yine kullanıcı tarafından belirlenen zaman adımları ile bir metin dosyasına yazar. *ElementTracker* ise aynı işlemleri *Element* nesneleri ve bu nesnelere ait bilgiler (gerilim, gerinim vb.) için yapar. *VtkTracker* ise sonlu elemanlar modelinin tümü ya da kullanıcı tarafından belirlenen bölüm için *Node* ve *Element* nesnelerine ait bilgileri Vtk dosya formatında kaydeder. Vtk dosya formatı Vtk Paraview programı (2012) ile görüntülenebilen ve paralel çözümlerin sonucu tutmakta yaygın olarak kullanılan bir dosya biçimidir.

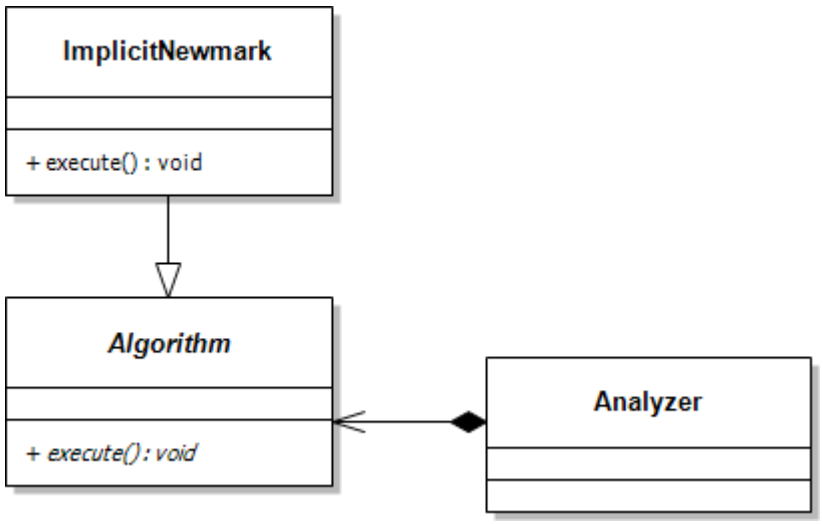
Panthalassa'ya bölümleme işleminde kullanılan *Grapher*, *Partitoner* ve *Divider* sınıflarından türeyen birçok algoritma da plug-in mantığı ile eklenmiştir. Eklenen paralel algoritmalardan *NodalGrapher* ve *ElementalGrapher* sınıfları *Grapher* sınıfından türer ve sırasıyla *Node* ve *Element* nesnelerini grafiksel gösterimin odağı olarak kullanarak sonlu elemanlar modelinden grafiksel gösterim oluştururlar. Metis kütüphanesini kullanarak grafiksel gösterimleri parçalayan *MetisPartitioner* ve bu uygulamanın paralel biçimi olan *ParMetisPartitioner Partitioner* Sınıfından türeyen ve Panthalassa'ya eklenmiş sınıflardır. *Divider* Sınıfından türeyen *PtlDivider* ise bölümlenen grafiksel gösterimlerden bölümlenmiş bir sonlu elemanlar modeli oluşturan bir sınıftır ve yine plug-in mantığı ile Panthalassa'ya eklenmiştir.

Örnek Uygulamalar

Tablo 3.6'da gösterildiği gibi Panthalassa'daki bütün çözümleme algoritmaları Algorithm sınıfından türetilerek uygulanmışlardır. Bu bölümde iki farklı çözümleme yönteminin uygulama detayları verilmiş ve bu sayede Panthalassa'nın veri yapısının ve çalışma mantığının daha iyi anlaşılması hedeflenmiştir.

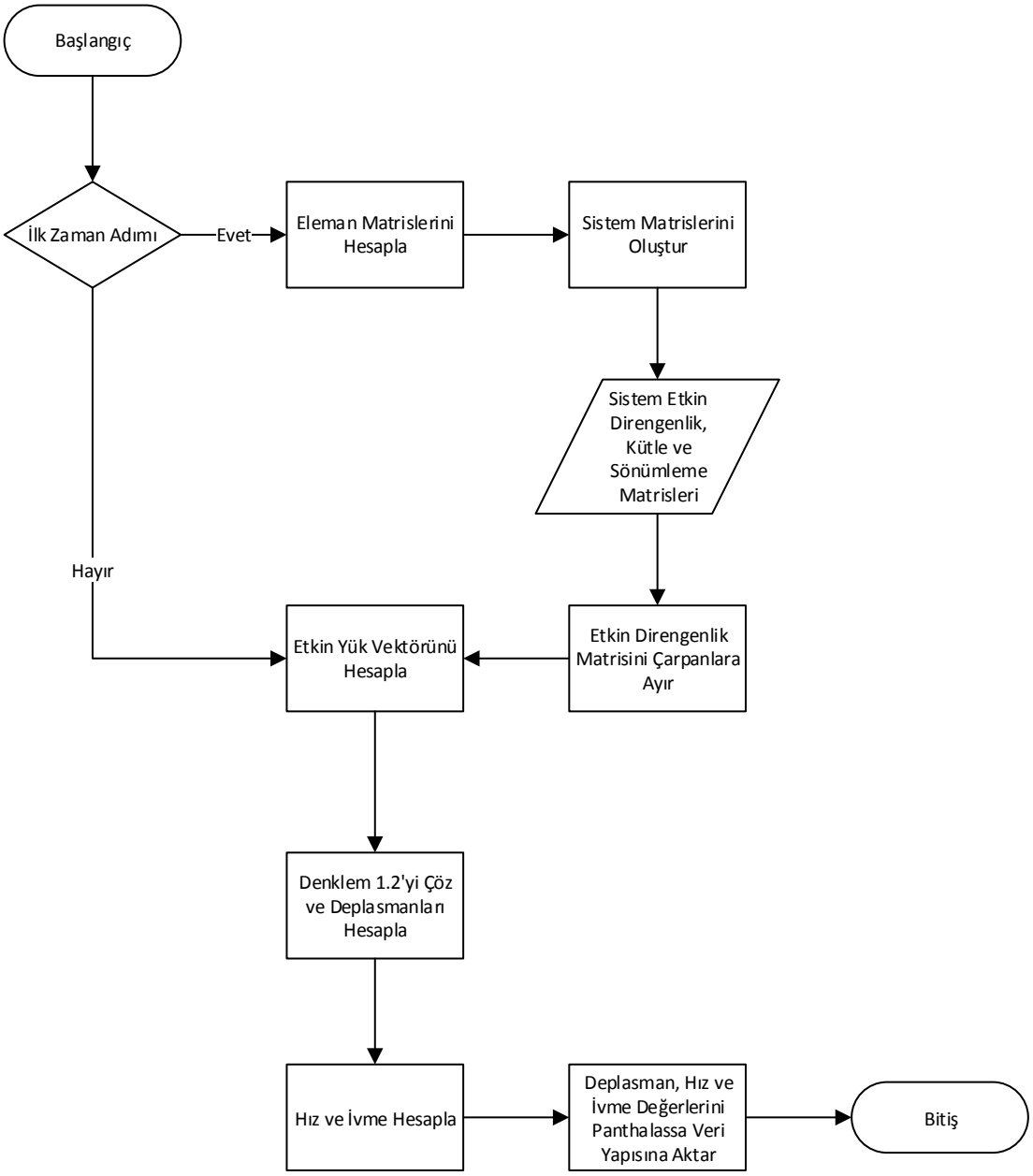
Zaman Alanında Doğrusal Örtük Çözümleme Yöntemi

Panthalassa'da Newmark Örtük Çözümleme Algoritmasının uygulaması *ImplicitNewmark* Sınıfı ile yapılmıştır (Şekil 3.18). *Algorithm* Sınıfından türeyen *ImplicitNewmark* Sınıfı, Newmark Örtük Çözümleme Algoritmasını *execute* Fonksiyonunda yürütür. Bu fonksiyon her zaman adımında *Analyzer* objesi tarafından çağrılır. *execute* Fonksiyonu her çağrılışında hareket denklemini çözerek denklemin çıktılarını Panthalassa veri yapısına yazar.



Şekil 3.18: ImplicitNewmark Sınıf Diyagramı

execute Fonksiyonunun işleyişi Şekil 3.19'da gösterilmiştir. *execute* Fonksiyonunda ilk zaman adımının işleyişi diğer zaman adımlarına göre farklıdır. İlk zaman adımında algoritma sonlu elemanlar modelindeki tüm elemanların direngenlik, kütle ve sönümleme matrislerini hesaplar ve bu matrisleri birleştirerek sistem etkin direngenlik matrisi, sistem kütle matrisi ve sistem sönümleme matrislerini oluşturur. Sönümleme matrisinin oluşturulması için Rayleigh yöntemi olarak bilinen ve sönüm matrisini direngenlik ve kütle matrisinin bir birleşimi olarak oluşturan yöntem kullanılmıştır. Daha sonra etkin direngenlik matrisi çarpanlarına ayrılır. Buradan sonraki işlemler tüm zaman adımları için aynıdır. Denklem 2.40'ın sağ tarafının toplamı olan etkin kuvvet vektörü hesaplanır. Bu vektör ve çarpanlarına ayrılmış etkin direngenlik matrisi kullanılarak deplasmanlar elde edilir. Deplasmanlardan hız ve ivme değerleri hesaplanır ve bu değerler Panthalassa *NodeDataHolder* objesine aktarılır.



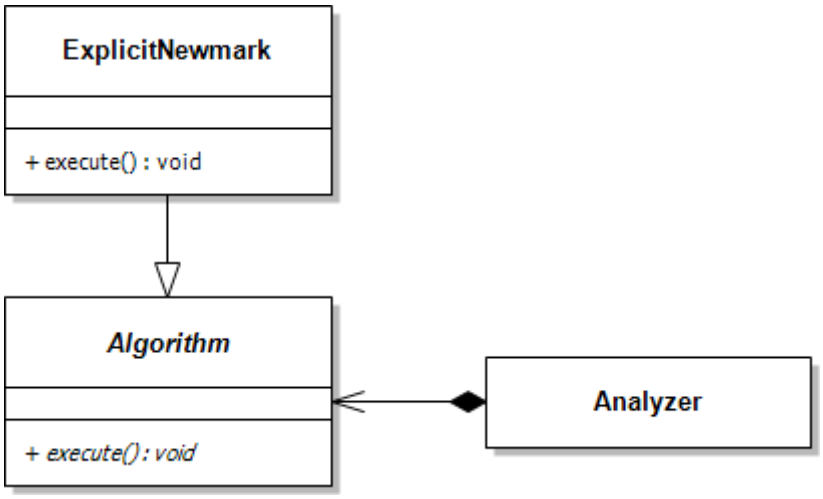
Şekil 3.19: *ImplicitNewmark execute* Fonksiyonunun İşleyiş Şekli

Doğrusal sistemlerde etkin direngenlik matrisi bir kere çarpanlarına ayrılır ve analiz zamanı boyunca kullanılır. *ImplicitNewmark* Sınıfı etkin direngenlik matrisinin her zaman adımında hesaplandığı bir versiyonuda içerir. Giriş dosyasında verilen bir opsiyonla çalıştıralabilecek bu versiyon eşdeğer doğrusal analiz gibi doğrusal ancak etkin direngenlik matrisinin sürekli değiştiği durumlarda kullanılabilir. Etkin direngenlik matrisinin çarpanlara ayrılması ve etkin kuvvet vektörü kullanılarak deplasmanlar bulunması

sırasında paralel statik çözümlemelerde de kullanılan MUMPS Kütüphanesi kullanılmıştır. Bu kütüphanenin detayları 4. Bölümde işlenmiştir.

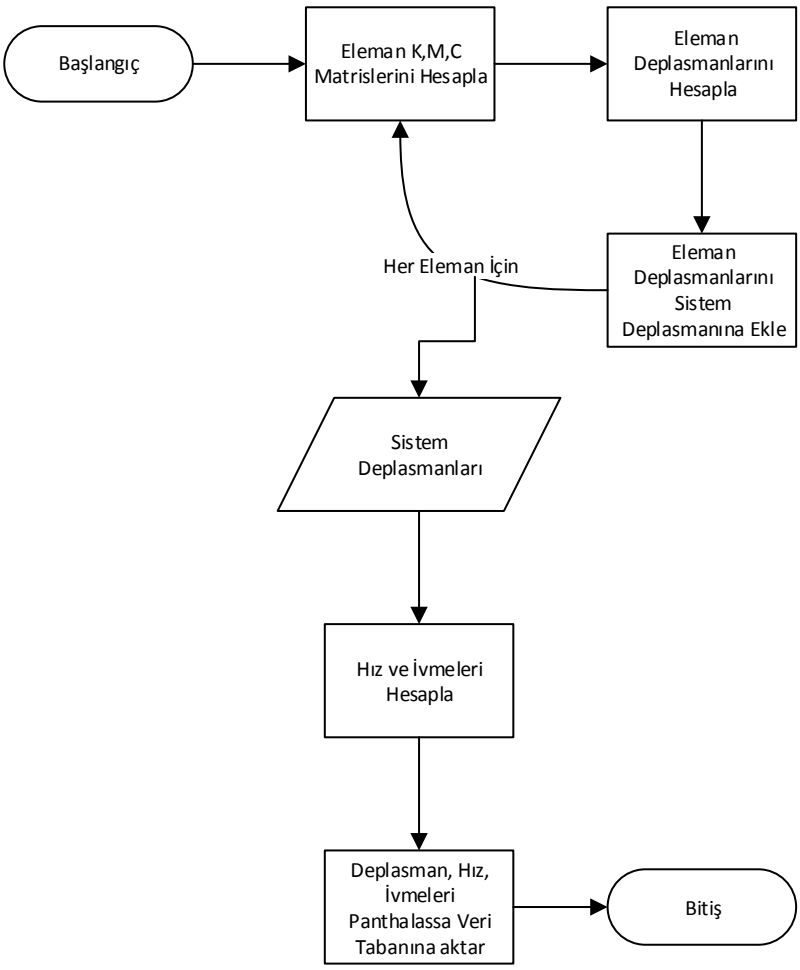
Zaman Alanında Doğrusal ve Doğrusal Olmayan Çözümleme Yöntemi

Panthalassa'da Newmark Belirtik Çözümleme Algoritması *ExplicitNewmark* sınıfı ile uygulanmıştır (Şekil 3.20). *Algorithm* Sınıfından türeyen *ExplicitNewmark* Sınıfı, Newmark Örtük Çözümleme Algoritmasını *execute* Fonksiyonunda yürütür. Bu fonksiyon her zaman adımında *Analyzer* objesi tarafından çağrılır. *execute* Fonksiyonu her çağrılışında Denklem 2.43'ü çözerek denklemin çıktılarını Panthalassa veri yapısına yazar.



Şekil 3.20: ExplicitNewmark Sınıf Diyagramı

execute Fonksiyonunun işleyişi Şekil 3.21'de sergilenmiştir. Bu fonksiyon ilk önce Denklem 5'i her eleman için çözer. Çözüm için her elemanın direngenlik, kütle ve sönümleme matrisi hesaplanır. Sönümleme matrisi hesaplanması için örtük algoritmadaki gibi Rayleigh yöntemi kullanılmıştır. Eleman matrisleri hesaplandıktan sonra Denklem 2.43 çözülür ve eleman deplasmanları hesaplanır. Hesaplanan eleman deplasmanları birleştirilerek sistem deplasmanı vektörü oluşturulur. Sistem deplasmanlarından sistem hızı ve ivmesi hesaplanır ve analiz sonuçları *NodeDataObject* objesine aktarılır.



Şekil 3.21: *ExplicitNewmark execute* Fonksiyonu İşleyiş Şekli

4. Bölüm

Paylaşık ve Dağıtık Sistemler için Paralel Çözümleme Algoritmalarının Geliştirilmesi

Giriş

Projenin belkemiğini oluşturan bu bölümde, yapı mekaniği çözümlemelerinde sıkça kullanılan algoritmaların paylaşık ve dağıtık sistemler için özelleştirilmesi ve büyük boyutlu modellerin çözümlenebilmesinin sağlanması üzerinde çalışmalar yapılmıştır. İlk olarak büyük doğrusal denklem sistemlerinin birden fazla durum için çözülmesini gerektirecek çözümleme algoritmaları üzerine çalışılmış, doğrusal statik çözümleme yönteminin, hem paralel toptan, hem de alt-yapı tabanlı çözücülerle uygulamaları yapılmıştır. Her iki paralel çözümleme algoritmasının başarımları en iyi hale getirildikten sonra, farklı hesaplama platformlarında, büyük boyutlu yapısal modellerin çözümlenmesiyle sınanmış ve karşılaştırılmıştır. Benzer şekilde, hem toptan hem de alt-yapı tabanlı çözücü kullanılarak, zaman alanında örtük çözümleme algoritmaları geliştirilmiş ve çeşitli denektaşı problemlerde her iki yöntemin de başarımı sınanmıştır.. Doğrusal olmayan çözümleme için yine toptan çözücü kullanan bir paralel algoritma geliştirilmiş ve algoritmanın başarımı çeşitli denektaşı problemlerle sınanmıştır. Doğrusal ve doğrusal olmayan dinamik çözümlemelerin gerçekleştirilmesi için ise paralel belirtik algoritmaların üzerinde durulmuştur. Belirtik algoritmaların paralel verimliliğinin yüksek olması, sistem matrislerinin oluşturulmasına gerek duyulmadığı için büyük boyutlu modellerin çözümüne olanak vermesi nedeniyle bu algoritmalar üzerinde durulmuş, dağıtık bellekli, paylaşımlı bellekli ve hem dağıtık hem de paylaşımlı bellekli sistemlerde çalışabilen, doğrusal ve doğrusal olmayan problemlerin zaman alanında çözümlenmesine olanak veren bir paralel algoritma geliştirilmiştir. Algoritmanın başarımı çeşitli boyutlardaki yapısal modellerle sınanmış ve milyon mertebesinde üç boyutlu sonlu elemanlara sahip modeller çözümlenebilmiştir.

Doğrusal Statik Çözümleme Algoritmaları

Büyük modellerin doğrusal çözümlemelerinin gerçekleştirilmesi için iki farklı yöntem kullanılmış ve bu yöntemlerin başarımları karşılaştırmalı olarak sınanmıştır. Yöntemlerden ilki toptan çözüm algoritması olarak adlandırılan algoritmada, sistem matrisleri paralel olarak oluşturulduktan sonra çekirdekler/işlemciler arası dağıtılarak sistem bir bütün olarak çözülür. Büyük matrislerin paralel olarak çözümü için MUMPS (MUltifrontal Massively Parallel Sparse Direct Solver: Çokyüzlü Paralel Seyrek Doğrudan Çözücü) (Amestoy et al. 2000) kütüphanesi kullanılmıştır. Alt-yapı tabanlı çözüm algoritması olarak adlandırılan ikinci yöntemde ise yapısal model ilk olarak çekirdek/işlemci sayısı kadar alt-yapılara bölümlenir. Alt-yapı sistem matrisleri paralel olarak alt-yapı sınır serbestlik derecelerine indirgendikten sonra sınır sistem matrisleri oluşturularak paralel olarak çözülür.

Toptan Çözüm Algoritması

Toptan çözüm algoritması iki ana basamaktan oluşmaktadır: Paralel sistem matrislerinin tümlenmesi (assembly)ve tümlenen seyrek denklem sisteminin bir seyrek matris çözücü yardımı ile çözülmesi.

Paralel Matris Tümlenmesi

Çözüm sürecinin tamamıyla paralel bir şekilde tamamlanması için direngenlik ve yük matrisleri bilgisayarlara paylaştırılmış bir şekilde tümlenmelidir. Bu şekilde hafızada büyük yer tutan direngenlik matrisi bilgisayarlar arasında paylaşılır ve hem tek bilgisayarda tutulabilecek direngenlik matrisinden daha büyük matrisler kullanılabilir ve tümleme işleme paralel bir şekilde yüksek performanslı olarak yapılabilir.

Paralel matris tümlemesinden faydalanmak için problem çözümünde kullanılan direngenlik matrisi seyrek koordinat matrisi tipinde bilgisayarlara dağıtılmış olarak tutulmuştur. Seyrek matris çözümü için kullanılan MUMPS kütüphanesi çözülecek seyrek matrislerin bilgisayarlar arasında değişik şekillerde dağıtılmasına olanak verir. Bu olanak sayesinde toptan çözüm algoritması uygulamasında, çözülecek olan sistemin direngenlik matrisi her çekirdek/işlemciye eşit sayıda eleman dağıtılarak hesaplanır. Tarafımızdan gerçekleştirilen çalışmalarda, sistem matrislerinin işlemciler arasında ne şekilde dağıtıldığının çözüm süresini değiştirmediği görülmüş bu sebepten dolayı da elemanlar işlemcilere sıralı bir şekilde dağıtılmıştır. Bu şekilde her bilgisayara eşit sayıda işlem ve bellek yüklemesi verilmiştir. Hafızada çok yer tutmayan ve hızlı olarak tümlenebilen yük matrisi ise tek bir bilgisayarda bir vektör halinde tutulmuştur.

Tümlenen Denklem Sisteminin MUMPS Yardımı ile Çözülmesi

MUMPS bilgisayarlar arasında bölünmüş olan direngenlik matrisini alır, ana bilgisayarda bir analiz sürecinden geçirir. Bu analiz sırasında kullanıcının seçtiği ya da MUMPS'ın otomatik olarak atadığı bir metotla sıralama yapılır, sembolik bir çarpanlara ayırmadan sonra matrisin grafiksel gösterimi oluşturulur. Analiz işleminden ortaya çıkan bilgiler ana bilgisayardan diğer bilgisayarlara gönderildikten sonra sayısal çarpanlara ayırma işlemi bir grup yoğun (dense) matris yardımıyla yapılır. Bu matrislere yüz (front) matrisleri denir. MUMPS önceden belirlenmiş hiyerarşik aşamalarda birden fazla sınır matrisini aynı anda ayrıştırdığı için bu metoda aşamalı çoklu sınır (multi-frontal) yöntemi adı verilmektedir. Kuvvet vektörü ana bilgisayardan diğer bilgisayarlara aktarılarak sınır matrislerinin çarpanlara ayrımından ortaya çıkan çarpanlar yardımıyla çözüm hesaplanır ve tekrardan ana bilgisayarda toplanır.

Panthalassa MUMPS kütüphanesini *Algorithm* sınıfından türeyen *MumpsSolver* sınıfıyla kullanır. *MumpsSolver* sınıfında yapılan uygulama hem tek hem de birden fazla işlemcili sistemlerde kodda değişiklik yapmadan kullanılabilecek şekildedir. *MumpsSolver* analiz işlemini gerçekleştirdikten sonra MUMPS'tan gelen çözüm vektörünü düğüm noktaları ile eşleştirilen bilgilerin tutlduğu *NodeDataHolder* Sınıfına aktarır ve kullanıcının istediği analiz sonuçları *Tracker* nesneleri yardımıyla dosyalanır.

Alt-yapı Tabanlı Çözümleme Algoritması

Alt-yapı tabanlı çözümleme yöntemi iki ana parçadan meydana gelmektedir: veri hazırlama ve paralel çözümleme. Veri hazırlama aşaması bilgisayarlar arası olabilecek en uygun veri dağıtımı şeklini belirlemek amacıyla geliştirilmiştir. Bu amaçla ilk olarak düğüm noktası koordinatları, eleman bilgileri, yükleme vb. bilgilerden oluşan yapısal veriler okunur, alt-yapılar çözüm için olabilecek en uygun şekilde oluşturulmaya çalışılır, direngenlik matrisi denklemlerini sıralar ve en son olarak paralel çözüm için gerekli verileri hazırlar. Veri hazırlama aşamasıyla ilgili detaylı bilgiler, tektürel sistemler için Kurc ve Will (2007) çoktürel sistemler için Kurc (2010)'da bulunabilir. Paralel çözümleme ise, eleman direngenlik matrislerinin

oluşturulması, bu matrislerin indirgenmesi, sınır denklemlerinin çözümü ve eleman kuvvetlerinin hesabı paralel bir şekilde gerçekleştirilir.

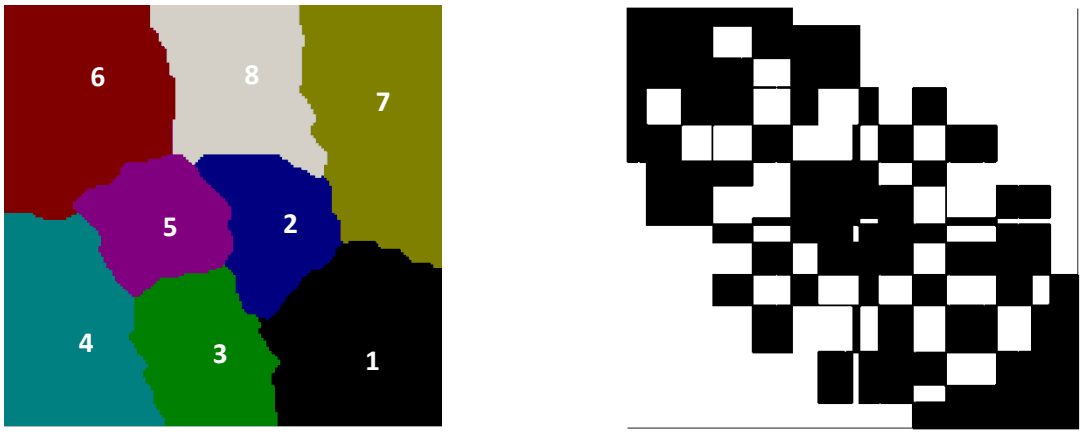
Paralel çözümleme her bilgisayarda alt-yapı ve ilgili nesnelerin oluşturulmasıyla başlar. Veri hazırlama aşamasında belirlenen alt-yapılara ait elemanlar kendilerinden sorumlu çekirdek/işlemcilerde yaratılır. Daha sonra yerel çözüm başlatılır. Her bir bilgisayar kendilerine ait alt-yapıların serbestlik derecelerini numaralandırır ve direngenlik matrisini ve yük vektörlerini oluşturur.

Bir sonraki adım ise iç serbestlik derecelerinin sınır serbestlik derecelerine indirgenmesidir. Bu aşamada kolon tabanlı seyrek matris çözüm yöntemi kullanılır. Bu yöntemde matrisin gerek alt gerekse üst üçgensel elemanlarının sadece sıfır olmayan elemanları bellekte tutulur. Seyrek matris indirgeme yöntemi için de çözüm süresi olarak en iyileştirilmiş çözüm kütüphanesi MUMPS seçilmiştir. Bu yöntemde MUMPS indirgeme işlemlerini gerçekleştirilmesi amacıyla kullanılmaktadır. İndirgeme işleminin sonuçlanmasından sonra sınır denklemlerinin çözümüne geçilir. Her bir alt-yapının sınır denklemleri takip eden bölümde detaylı bir şekilde anlatılan yöntemle toparlanarak sınır direngenlik matrisi ve yük vektörleri oluşturulur. Yoğun bir matris olan sınır direngenlik matrisi ScaLAPACK (Choi et al.) kütüphanesinde dağıtık sistemler için geliştirilmiş paralel yoğun matris çözücüleri kullanılarak ayrıştırılıp sınır deplasmanları hesaplanır.

Sınır deplasmanları hesaplandıktan sonra, bu deplasmanlar her bir bilgisayar kendi alt-yapılarının sınır düğüm noktalarına karşılık gelen deplasmanlara sahip olacak şekilde dağıtılır. En son aşamada da alt-yapıların iç düğüm noktalarının deplasmanları ve eleman kuvvetleri hesaplanır.

Sınır Denklemlerinin Paralel Çözülmesi

Sınır denklemlerinin hızlı bir şekilde paralel olarak çözülmesi amacıyla yoğun matrisler için geliştirilmiş olan ScaLAPACK kütüphanesi mercek altına alınmıştır. Şekil 4-1'de sekiz alt-yapıya bölünmüş olan kare bir plağın sınır denklem matrisinin dolu bölgeleri gösterilmiştir. Sınır denklemleri matrisi, Şekil 4-1'de görüldüğü üzere, kabaca yoğun matris özelikleri göstermektedir. Farklı yapısal modellerin farklı altyapı sayılarıyla oluşturulan sınır denklem matrisleri incelendiğinde, sınır denklemleri matrisi geniş bantlı bir matristen tamamı dolu bir matrise kadar çeşitlilik göstermiştir.



Şekil 4-1 Sekiz Altyapılı Kare Plak Modelinin Sınır Denklem Matris Yoğunluğu

Sınır denklemlerini çözecek olan algoritmanın serbestlik derecesi indirgeme süreci sonucunda her bir bilgisayarda parça parça oluşturulmuş olan sınır denklemleri matrisini ScaLAPACK fonksiyonlarının istediği biçimde tekrar bilgisayar kümesinde dağıtması, çözüm öncesi ve sonrasında minimum hafıza kullanarak en hızlı şekilde denklemeleri çözmesi gerekmektedir.

Sınır denklemlerinin çözülmesi aşaması sınır denklemi parçalarının bilgisayarlar arası nasıl ve hangi sırayla dağıtılacağının belirlenmesi, matris parçalarının dağıtılarak bilgisayarlarda matris parçalarının (yerel matrisler) oluşturulması, denklemlerin çözülmesi olmak üzere üç ana bölümden oluşmaktadır.

Her bir bilgisayarda serbestlik derecesi indirgeme süreci sonucunda oluşan aslında global düzeyde sınır denklemleri matrisini ifade eden veriler yer almaktadır. Bu veriler o alt-yapının sınır denklemleri matrisine ve o serbestlik noktalarındaki yüklemesine katkılarını içermektedir. Tabiî ki bu katkıların global düzende nerelere yerleştirilmesi gerektiğini içeren haritalama bilgileri de bu verilere dahildir. ScaLAPACK çözücülerinin çalıştırılması için bu katkıların uygun bir dağıtıma göre düzenlenmeleri gerekmektedir. Bu nedenle ilk olarak bilgisayarlarda indirgeme sonrasında sınır denklemlerine karşılık gelen veriler irdelenerek, hangi verilerin inceleyen bilgisayarda kalması, hangilerinin diğer bilgisayarlara gönderilmesi gerektiği belirlenir ve ilgili bilgisayarlara gönderim yapılır. Görüldüğü üzere bu bölümde cevaplanması gereken iki önemli sorudan ilki verilerin hangi bilgisayara ait olduğunun belirlenmesi, ikincisi ise veri aktarımının nasıl bir düzende gerçekleştirileceğidir.

Verilerin hangi bilgisayara ait olduğu yapay test aşamasında yazılmış olan döngüsel-blok dağıtımın çözümlenmesiyle elde edilmektedir (Dongarra et al.). Basit bir aritmetik işlem sonucunda ortaya çıkan verinin hangi bilgisayara ait olduğu bilgisi göz önünde bulundurularak her bilgisayar diğer bilgisayarlara göndereceği veri paketlerini oluşturur. Bu pakette o verilerin global düzendeki yerleri ve değerleri tutulmaktadır. Daha hızlı bir veri alışverişi gerçekleştirebilmek için veri paketlerine sıkıştırma işlemi uygulanabilir.

Hazırlanan paketler artık gönderime hazırdır ancak düşünülmeden yapılmış bir gönderim şeması bilgisayarların birbirlerini boşuna beklemelerine neden olacağı için gönderim başarımı çok düşürebilir. Bu nedenle, ek bir algoritmayla bilgisayarların meşgul olan bilgisayarlarla değil de boş olanlarla iletişimlerini kurup gönderimlerini yapabilecekleri bir şema oluşturulur. Bu işlem sırasında bilgisayarlar arasında bir iletişim olmamakta her biri ayrı ayrı aynı şemayı elde etmekte ve ona göre gönderim yapmaktadır. Bu algoritma programlamada çokça kullanılan basit bir işlem sırası algoritmasıdır. Gelecekte bu algoritmaya gönderilen verilerin ağırlıklarını göze alacak bir yapı eklenmesi gönderim başarımını daha da yukarılara çekecektir.

Veri değiş-tokuşu sonrasında her bilgisayar, sanki en baştan bir döngüsel-blok dağıtım metoduyla verileri dağıtılmış gibi kendilerinde olması gereken verilere sahiptir. Ancak, hala ScaLAPACK kütüphanesinin istediği yerel matrisler oluşturulmamıştır. Bu aşamada cevaplaması gereken iki sorudan ilki veri dağıtım yöntemine göre o bilgisayara ait olan yerel matrisin boyutlarının ne olduğu, ikincisi ise global matriste yeri bilinen bu verilerin yerel matriste nerelere yerleştirilecekleridir (Şekil 4-2). Ancak bu iki soru da ScaLAPACK kütüphanesinin hangi çözüm fonksiyonun kullanılacağı ile ilintilidir.

Sınır denklemleri matrisi yukarıda da belirtildiği üzere geniş bantlı bir matristen tam dolu bir matrise kadar değişkenlik gösterdiği için, geniş bantlı matris çözümü için ScalaPack kütüphanesinin PDPBSV, tam dolu matris çözümü için ise PSGESV fonksiyonları seçilmiştir. Bu iki algoritma da pozitif simetrik matrisler için özelleştirilmiş olan Cholesky metodunu kullanarak çözüm yapmaktadır. Ancak ScaLAPACK kütüphanesi hafıza kullanımını minimuma indirgemek amacıyla bu iki fonksiyon için istediği veri yapısı birbirinden farklıdır. Dağıtım olarak tam dolu matriste iki boyutlu döngüsel blok dağıtım beklenirken,

geniş bantlı çözüm için tek boyutlu döngüsüz blok dağıtım beklenmektedir. Her iki algoritmada da sadece matrisin herhangi bir üçgen yarısı depolanmalıdır.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & 0 & 0 \\ 0 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & 0 \\ 0 & 0 & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ 0 & 0 & 0 & a_{64} & a_{65} & a_{66} & a_{67} \\ 0 & 0 & 0 & 0 & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

Processes

0			1			2
*	*	a_{13}	a_{24}	a_{35}	a_{46}	a_{57}
*	a_{12}	a_{23}	a_{34}	a_{45}	a_{56}	a_{67}
a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}
a_{21}	a_{32}	a_{43}	a_{54}	a_{65}	a_{76}	*
a_{31}	a_{42}	a_{53}	a_{64}	a_{75}	*	*

Processes

0			1			2
*	*	a_{31}	a_{42}	a_{53}	a_{64}	a_{75}
*	a_{21}	a_{32}	a_{43}	a_{54}	a_{65}	a_{76}
a_{11}	a_{22}	a_{33}	a_{44}	a_{55}	a_{66}	a_{77}

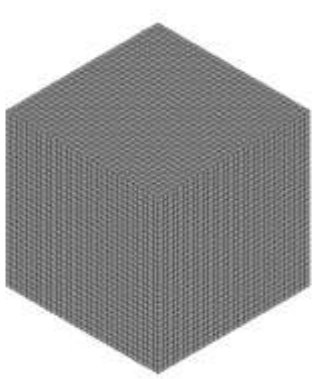
Şekil 4-2 Geniş bantlı matrisin simetrik olmayan ve olan paketler halinde tutulması (Dongarra et al.)

Yerel matrislerin hazırlanması aşamasından sonra her bilgisayar ScaLAPACK fonksiyonlarının ihtiyaç duyacağı hafızaları oluşturarak çözüm algoritmasına girer. Bu her iki algoritmada Cholesky ayrıştırması yaparak paralel bir şekilde çalışır ve bu şekilde sınır düğüm noktalarında deplasmanlar elde edilir.

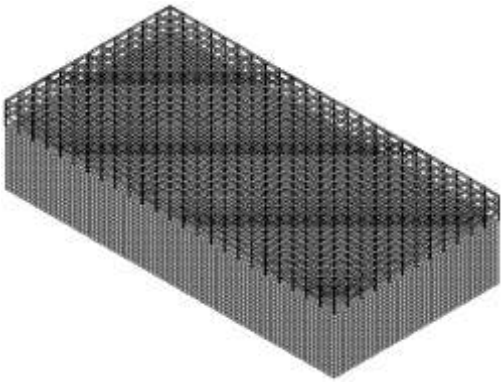
Sınır Denklemleri Çözüm Algoritmasının Test Edilmesi

Alt yapı tabanlı çözüm algoritmasında kullanılan sınır denklem çözücünün başarımı iki farklı bilgisayar kümesi kullanılarak test edilmiştir. HPCE kod adlı, ilk bilgisayar kümesi, her birisi Intel P4 3.2 GHz işlemcili, 1 GB geçici belleğe sahip sekiz adet bilgisayar oluşurken diğer bilgisayar kümesi, HPCE2, Intel Core2Quad Q9300@2.5 GHz işlemcili 3.23 GB geçici belleği olan sekiz adet bilgisayarlardan oluşmaktadır. Intel Core2Quad serisi işlemciler kuramsal olarak 2.5 GHz işlem gücüne sahip dört adet çekirdekten oluşmaktadır ve bu işlemciler aynı hafızayı paylaşımlı olarak kullanabilirler. Ancak bu testler esnasında işlemcinin bu özelliği kullanılmamıştır. Her iki bilgisayar kümesi de 1GBit yerel ağ bağdaştırıcısı kullanılarak oluşturulmuştur ve bütün bilgisayarlarda Windows XP işletim sistemi kullanılmaktadır.

Testler uygulamada sıkça karşılaşılan özelliklerdeki modelleri başarıyla örnekleyebilecek üç farklı model üzerinde uygulanmıştır. İlki 30x30x30 parçaya bölünmüş sekiz düğüm noktalı dörtgen prizma elemanlardan oluşan matematiksel bir küp (Cube) modelidir (Şekil 4-3a). İkinci model ise, bir boyutlu ve iki boyutlu sonlu elemanlardan oluşan bir nükleer atık arıtma tesisi (GroM) modelidir (Şekil 4-3b). Son model ise pratikte sıkça karşılaşılan özelliklere sahip, bir boyutlu ve iki boyutlu elemanlardan oluşan, oldukça düzensiz bir yapı olan otel (StRegis) modelidir (Şekil 4-3c).



(a) Küp Modeli

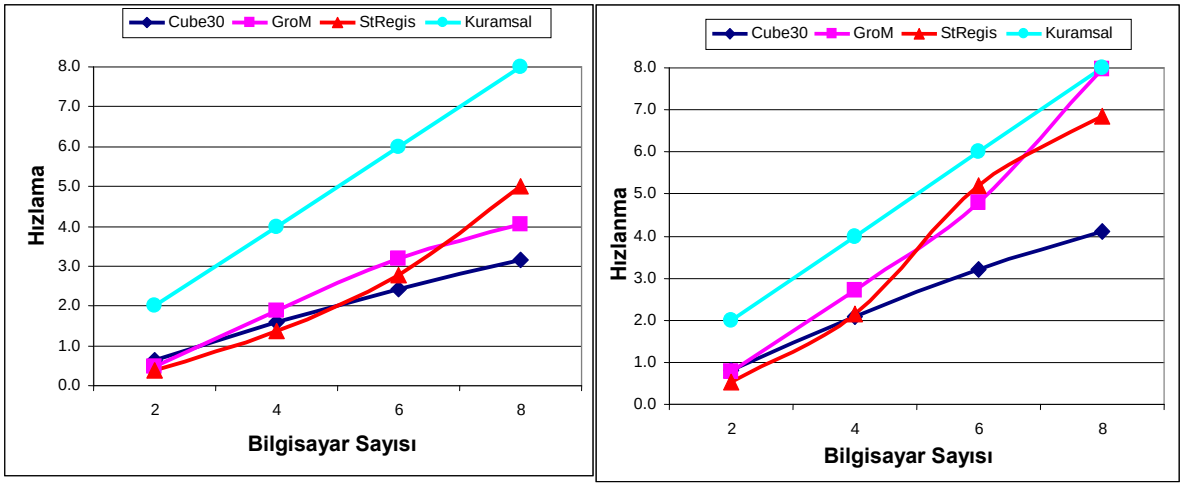


(b) Nükleer Atık Tesisi Modeli

(c) Otel Modeli

Şekil 4-3 Örnek Yapısal Modeller

Burada sınır denklem çözüm algoritmasının başarımının test edilmesi amaçlandığı için bu modellerin çözümleri sırasında oluşan sınır denklemlerinin 2, 4, 6 ve 8 bilgisayar kullanılarak yapılan çözümlerinin süreleri ölçülmüş ve bu süreler, bu çözümlerin tek bilgisayar kullanılarak yapılması için gerekecek olan muhtemel çözüm süreleriyle karşılaştırılmıştır. Sonuçlar Şekil 4-4'te verilmiştir.



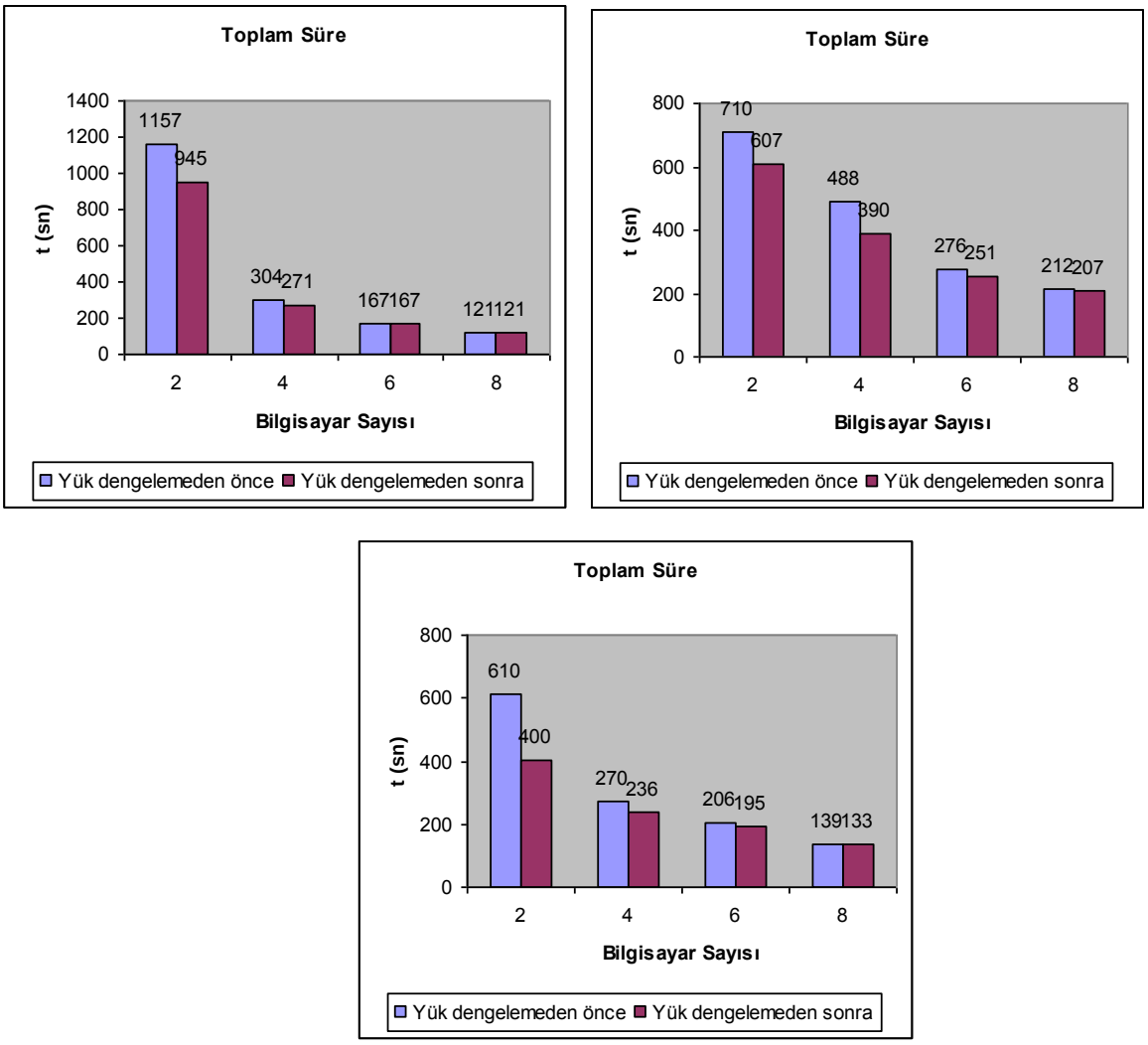
Şekil 4-4 Farklı Modellerin Sınır Direngenlik Matrislerinin, HPCE ve HPCE2 Bilgisayar Kümelerinde çözümünde elde edilen hızlanma grafikleri

Modele ve ayrıştırma metoduna göre farklılık gösterse de genel olarak 2 bilgisayarla yapılan çözümlerde sınır denklemlerinin sayısı 2000 civarındayken, 4 bilgisayarlı çözümlerde 5000, 6 bilgisayarlı çözümlerde 9000, 8 bilgisayarlı çözümlerde ise 12000 civarındadır. Şekil 4-4'de görüldüğü üzere iki bilgisayarlı çözümlerde başarım elde edilememiştir. Bunun nedeni, bu çözümde ortaya çıkan sınır denklemlerin paralel olarak çözümü için gereken veri aktarım zamanının bu denklemlerin tek bir bilgisayarda çözülmesinden daha uzun sürmektedir. Diğer çözümler incelendiğinde HPCE için 4 bilgisayarlı çözümde yaklaşık iki kat, 6 bilgisayarlı çözümde yaklaşık üç kat, 8 bilgisayarlı çözümde ise dört kat hızlanma elde edilmiştir. HPCE2 için ise daha yüksek bir başarım elde edilmiş hatta GroM modeli için kuramsal sınır değerine neredeyse ulaşılmıştır. Bunun nedeni yapının alt-yapılara bölünmesiyle çözüm süresini büyük oranda etkileyen bant genişliklerinin alt-yapı sayısıyla orantılı olarak azalmış olmasıdır.

Alt-yapı Tabanlı Çözümleme Platformunun Test Edilmesi

Alt-yapı tabanlı çözümleme yöntemi kullanılarak Şekil 4-3'de gösterilen modeller HPCE isimli bilgisayar kümesinde çözümlenmiştir. Bu üç modelle ilgili sonuçlar Şekil 4-5'te yer almaktadır. Grafiklerde yük dengeleme işlemi yapılmaksızın elde edilen sonuçların yanı sıra yük dengeleme yapıldıktan sonra elde edilen sonuçlar da verilmiş. Görüldüğü üzere küp modeli hariç yük dengeleme işlemi toplam çözüm süresinde iyileşmelere neden olmuştur. Küp modelinde bir iyileşme elde edilememesinin nedeni modelin karakteristik olarak her yönde simetrik ve tamamen düzenli olması sonucu ilk ayrıştırma işleminin gayet dengeli bir bölümleme yapmasından kaynaklanmaktadır.

Testlerde ortaya çıkan önemli sonuçlardan bir diğeriyse alt-yapı sayısı arttıkça yük dengeleme işleminin etkinliğinin azalıyor olmasıdır. Alt-yapı sayısının artması sınır denklemlerinin sayısını ciddi derecede arttırması nedeniyle yük dengeleme sonucunda elde edilen iyileşme bu sınır denklemlerinin çözümü için fazladan gereken süreden daha düşük kalmaktadır. Bu testler sonucunda elde edilen en önemli bulgu ise alt-yapı tabanlı çözüm metodu ile yapılan çözümlerin kullanılan bilgisayar sayısıyla genelde orantılı olarak daha az zamana ihtiyaç duyduğunun gözlenmiş olmasıdır.



Şekil 4-5 Sırasıyla Küp, Nükleer Atık Arıtma Tesisi ve Otel Modellerinin HPCE Bilgisayar Kümesi kullanılarak yapılan çözümlerin toplam süreleri

Birden Fazla Çekirdek Kullanarak Altyapı Tabanlı Çözüm

Bir önceki bölümde sunulan sonuçlar, her bir bilgisayarda tek bir işlem (process) kullanılarak gerçekleştirmiştir. Oysa günümüz modern bilgisayarlarında artık işlemciler birden fazla çekirdeğe sahip olduklarından dolayı, bu çekirdeklerin de hesaplama sırasında, özellikle indirgeme aşamasında, kullanımı çözümleme süresini azaltacaktır. Bu amaçla, her bilgisayarda birden fazla sayıda yaratılan işlemler (process), bilgisayar isimleri kullanılarak bir *MPI_Communicator* altında gruplanabilirler. Böylece her bilgisayar sorumluluğu altındaki altyapının denklemlerini birden fazla çekirdeği kullanarak oluşturur. Mesela, bu grupta yer alan ilk işlem altyapı veritabanında yer alan ilk sonlu elemanın denklemlerini altyapı direngenlik matrisine yerleştirirken, bu gruptaki ikinci işlem ikinci sonlu elemanın denklemlerini yerleştirir. Bu grupta başka işlem olmadığını farz edersek birinci işlem üçüncü sonlu elemanın, ikinci işlem ise dördüncü sonlu elemanın direngenlik matrislerini, altyapı direngenlik matrisine yerleştirecektir. Dağıtım bu şekilde dönüşümlü (cyclic) olarak devam edecektir. MPI, işletim sisteminin izin verdiği kadar

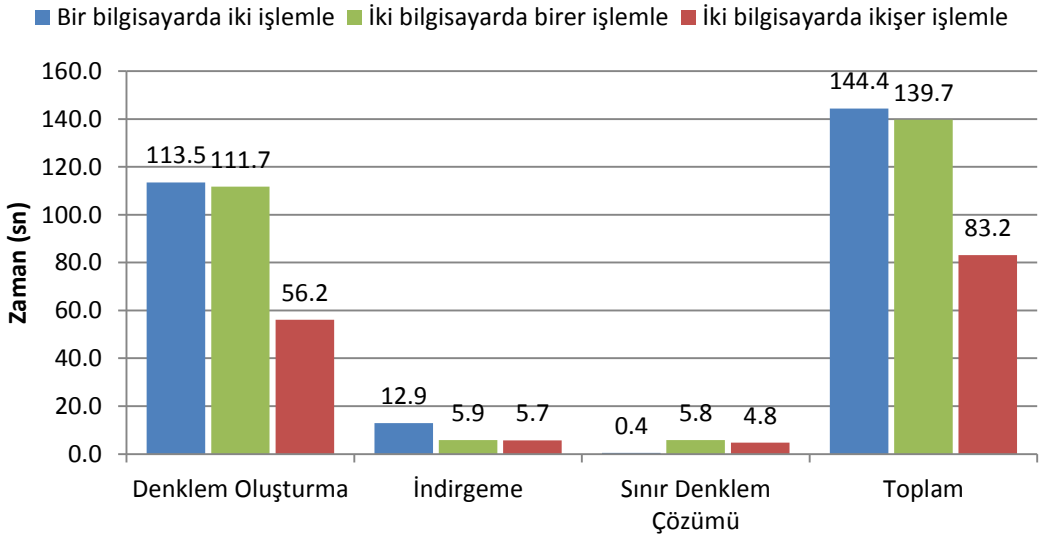
işlem çalıştırmaya olanak sağlasa da işlem sayısının çekirdek sayısını aşması veri alışverişi ve işletim sisteminin diğer gereksinimleri için çoğunlukla iyi sonuç vermeyeceği öngörülebilir.

Altyapı denklemlerinin sınır düğüm noktalarına indirgenmesi işlemi de bu işlem grubu tarafından yapılabilir. *MPI_Communicator* sınıfı aracılığıyla işlemlerin bu şekilde gruplanması, bu hesaplamalar süresince herhangi bir veri alışveriş hatasını tamamen önlemektedir. Benzer şekilde sınır denklemlerin çözümü için kullanılan yoğun veya bant matris çözücü de her bir bilgisayardan seçilen sözcü işlemlerden oluşturulan yeni bir *MPI_Communicator* grubunu kullanarak sınır denklem çözümünü yapmaktadır. Aynı bilgisayarda yer alan birden fazla işlemin sınır denklem çözümüne dâhil edilmesi veri alış-verişi ihtiyacını arttıracağından bant genişliğinin kullanışlılığını azaltacağı öngörülebilir. O nedenle her bir bilgisayardan bir adet işlem sınır denklem çözümüne dâhil olmaktadır. Sınır denklem çözümü sonrası yine aynı bilgisayar içerisindeki işlemlerden oluşan grup, sınır denklem çözümlerini kullanarak altyapının geri kalan yer değiştirmelerini hesaplar. Bu şekilde bütün yapının çözümü elde edilmiş olunur.

Birden Fazla Çekirdek Kullanarak Altyapı Tabanlı Çözücülerin Sınanması

Altyapı tabanlı çözücülerin birden fazla çekirdek kullanımının sınanması amacıyla (400x400) 160.000 adet kabuk elemanın her düğüm noktasında 3'er adet serbestlik derecesi kullanılarak elde edilen, yaklaşık 500.000 denkleme sahip olan bir matematiksel model, HPCE2 isimli homojen bilgisayar kümesi kullanılarak çözümlenmiştir.

İlk olarak, farklı sayıda çekirdek kullanımının çözüm süresi üzerindeki etkisinin sınanması amacıyla, çözüm yönteminin ana işlemleri, üç farklı durum için detaylı olarak incelenmiştir. Bu testte seçilmiş olan matematiksel model iki adet altyapıya bölünmüş olup ilk durumda bu model bir bilgisayarda işletilen iki adet işlemin (process) her birisine bir adet altyapı yükleyerek çözümlenmiştir. İkinci durumda ise bu iyileştirme yapılmadan önce kullanılan yöntem olan, her bir bilgisayarda birer işlem yaratarak çözüm elde edilmiştir. Son durumda ise yapılan yenilik sayesinde iki bilgisayarda ikişer adet işlem yaratarak bu bilgisayarların ikişer adet işlemcisinin kullanılması sağlanmıştır.



Şekil 4-6 İki adet altyapıya bölümlenmiş olan yapının farklı durumlar altında çözümü

Şekil 4-6'da göze çarpan ilk detay denklem oluşturma işleminin toplam işlem süresindeki baskınlığıdır. Bu durum artan altyapı sayısıyla azalacak ve sınır denklem çözümü daha baskın hale gelecektir. Bu durum bir sonraki test içerisinde detaylı olarak tartışılacaktır.

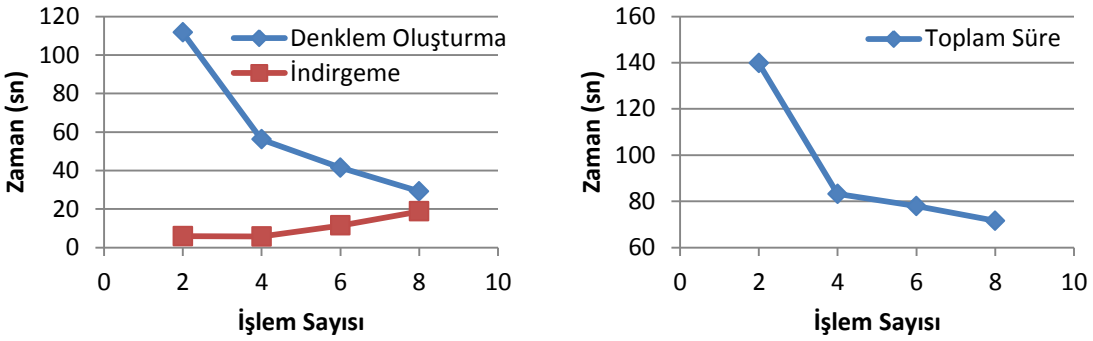
Görüldüğü üzere işlemlerin farklı bilgisayarlara yerleştirilmesiyle denklem oluşturma süresinde ciddi bir değişiklik olmaz iken indirgeme süresinde bir azalma, sınır denklem çözümü süresindeyse bir artış meydana gelmiştir. İndirgeme hiçbir veri alışverişine ihtiyaç duyulmayan bir işlemdir. Ancak bir bilgisayarda iki adet işlem kullanılması, kullanılan hafıza miktarının sınıra dayanmasına ve bunun yanı sıra veri yolunda sıkışıklığa neden olarak yavaşlamaya neden olmaktadır. Sınır denklem çözümündeki 0.4 saniyeden 5.8 saniyeye artış ise bilgisayarlar arası veri alışverişi için harcanan süreden kaynaklanmaktadır. İşlemler (process) aynı bilgisayarda iken bu veri akışı 0.4 saniye gibi çok kısa bir sürede gerçekleşirken proseslerin farklı bilgisayarlara yerleştirilmesiyle bu işlem ağ bağdaştırıcısı ve bağlantısı aracılığıyla gerçekleştirilmektedir.

Her bir bilgisayarda ikişer adet işlemin (process) kullanıldığı üçüncü durumda ise her bir altyapıdan iki adet işlem sorumludur. Bu şekilde iki çekirdeğin kullanımı sağlanmıştır. Bu nedenle denklem oluşturma sırasında harcanan süre neredeyse yarı yarıya azalmıştır. İndirgeme süresinde ve sınır denklem çözümünde bir miktar azalma gözlenmiştir. Sonuç olarak denklem oluşturma işlemindeki bu kazanım toplam sürelere de yansımış ve 112 saniyede tamamlanan çözüm 83 saniyede tamamlanabilmiştir. Bu %35'lik bir hızlanmaya denk gelmektedir.

Bu ön incelemenin ardından yapılan değişikliklerin farklı altyapı sayılarında nasıl bir davranış sergilediğini incelemek amacıyla farklı testler yapılmıştır. Bu yeniliğin temelde etki ettiği hesaplar denklem oluşturma ve indirgeme hesaplamaları olduğu için özellikle bu kısımlar incelenmiştir.

İkinci aşama testlerde daha önceden ikiye, dörde, altıya ve sekize bölümlenmiş olan yapısal modeller sırasıyla iki, dört, altı ve sekiz bilgisayarın her birinde birer, ikişer, üçer ve dörder işlem çalıştırılarak çözümlenmiştir. Örneğin Şekil 4-7'de sonuçları verilen iki adet altyapıya bölünmüş yapı iki bilgisayar kullanılarak çözümlenmiştir. Her bir bilgisayarda sırasıyla bir, iki, üç ve dört işlem (process) çalıştırılarak sınamalar gerçekleştirilmiştir.

Şekil 4-7'de görüldüğü üzere her bir bilgisayarda çalıştırılan işlem sayısı arttıkça altyapı başına düşen işlem sayısının artması nedeniyle denklem oluşturma işi artan işlem (process) sayısıyla orantılı olarak azalmaktadır. Örneğin önceden bilgisayar başına bir işlem çalıştırılırken 112 saniyede tamamlanabilen denklem kümesi oluşturma işi, bilgisayar başına iki adet işlem çalıştırılmasıyla 56 saniyede tamamlanabilmektedir. Bilgisayar başına üç ve dört işlem çalıştırılmasıyla da bu zamanlama sırasıyla 41 saniyeye ve 29 saniyeye düşmektedir.

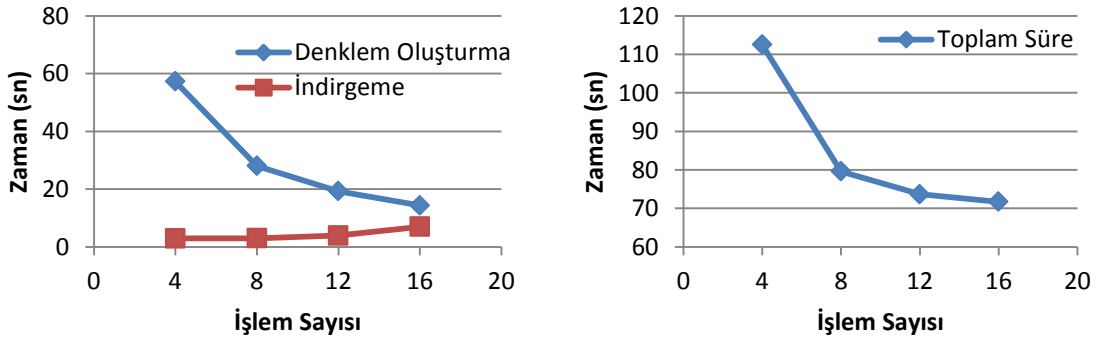


Şekil 4-7 İki adet altyapıya bölümlenmiş olan yapının iki bilgisayar kullanılarak çözümlenmesinin (a) detaylı süreleri, (b) toplam süreleri

Ancak denklem oluşturma sonrası indirgeme hesabında bir miktar yavaşlama meydana gelmektedir. İndirgeme işlemiyle çarpanlama işleminin benzerliği düşünüldüğünde indirgeme sırasında meydana gelen yavaşlamanın temel sebebi indirgeme için kullanılan MUMPS çözücüsünün kullanım kılavuzunda açıklanmıştır. Kılavuza göre indirgeme işinin bütün modelin direngenlik matrisinin kısmi çarpanlanması olması nedeniyle hesaplama hızını ve kalitesini arttıran bazı yöntemler hesaplama sırasında kullanılmamaktadır (Amestoy et al. 2000). Buna rağmen, detaylı sürelerde de görülebileceği üzere indirgeme süresindeki yavaşlama denklem oluşturma işleminin paralelleştirilmesi sonucu elde edilen kazanım düşünüldüğünde göz ardı edilebilecek düzeydedir.

Toplam çözüm süreleri incelendiğinde denklem oluşturma süresinin etkisinin orantılı olarak oraya da yansıdığı görülebilmektedir. Örneğin, önceden her bir bilgisayarda birer adet işlem (process) kullanımıyla çözüm 140 saniyede tamamlanırken, bilgisayar başına ikişer adet işlem çalıştırılarak çözüm 83 saniyede tamamlanabilmektedir. Bu sayede, bilgisayar başına ikişer adet işlem kullanarak elde edilen başarım 1.68 kat hızlanmaktadır. Bilgisayar başına üçer ve dörder işlem kullanılması halinde ise başarım sırasıyla 1.80 ve 1.96 kat hızlanma olarak gerçekleşmiştir.

Şekil 4-8'de dörde parçalanmış olan yapının dört bilgisayarda birer, ikişer, üçer ve dörder işlem (process) işleterek elde edilmiş sonuçları görülmektedir.

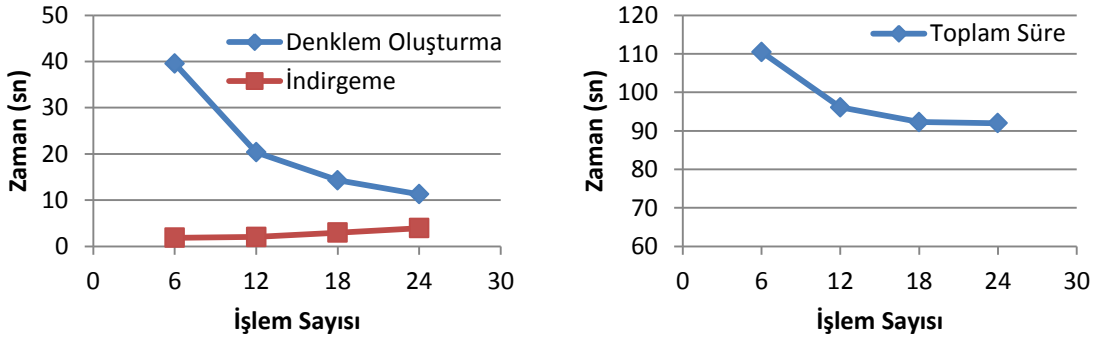


Şekil 4-8 Dört adet altyapıya bölümlenmiş olan yapının dört bilgisayar kullanılarak çözümlenmesinin (a) detaylı süreleri, (b) toplam süreleri

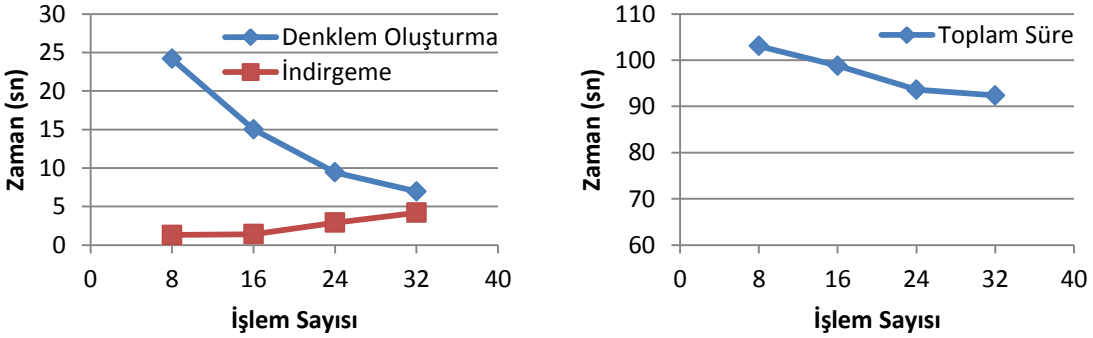
Bir önceki teste benzer şekilde bu test sonuçlarında da her bir bilgisayarda çalıştırılan bir altyapıdan sorumlu olan işlem (process) sayısı arttıkça bununla orantılı olarak denklem oluşturma işleminde hızlanma gözlenmektedir. Örneğin, her bir bilgisayarda birer işlem (process) kullanarak 57 saniyede gerçekleştirilen denklem oluşturma işlemi, bilgisayar başına dörder işlem kullanılmasıyla 14 saniyede tamamlanabilmektedir. Ancak, bilgisayar başına dörder işlem kullanımıyla indirgeme işlem süresi 3 saniyeden 7 saniyeye yükselmiştir. Yukarıda bahsedilen nedenlerin yanı sıra bilgisayardaki bütün işlemcilerin kullanılması nedeniyle veri alış-verişi ve işletim sistemi ihtiyaçları bu sürenin artışına neden olmaktadır.

Şekil 4-8'deki toplam süreler incelendiğindeyse daha önceden 113 saniyede gerçekleştirilen çözüm bilgisayar başına dörder işlem (process) kullanılmasıyla 71 saniyede tamamlanmaktadır. Diğer bir deyişle çözümün başarımı bilgisayar başına ikişer, üçer ve dörder işlem kullanılmasıyla sırasıyla 1.39, 1.51 ve 1.62 kat hızlanma olarak ölçülmüştür.

Şekil 4-9'da altıya parçalanmış olan yapının altı bilgisayarda birer, ikişer, üçer ve dörder işlem işletilerek elde edilmiş sonuçları, Şekil 4-10'da ise sekize parçalanmış olan yapının sekiz bilgisayarda birer, ikişer, üçer ve dörder işlem işletilerek elde edilmiş sonuçları yer almaktadır.



Şekil 4-9 Altı adet altyapıya bölümlenmiş olan yapının altı bilgisayar kullanılarak çözümlenmesinin (a) detaylı süreleri, (b) toplam süreleri



Şekil 4-10 Sekiz adet altyapıya bölümlenmiş olan yapının sekiz bilgisayar kullanılarak çözümlenmesinin (a) detaylı süreleri, (b) toplam süreleri

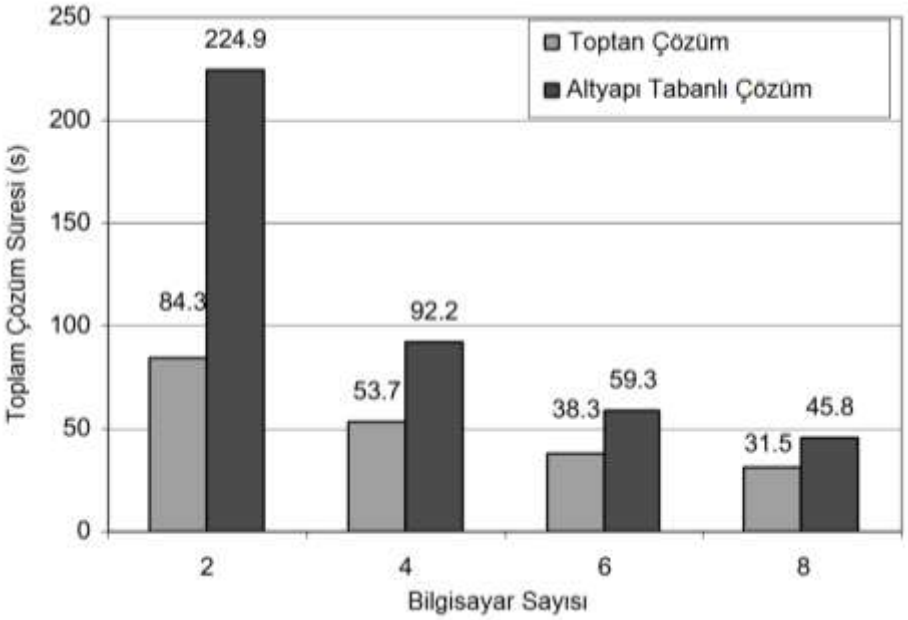
Şekil 4-9 ve Şekil 4-10'da önceki testlerde gözlemlenen davranış kendini tekrar etmektedir. Şekil 4-9'da toplam çözümdeki başarım her bir bilgisayarda ikişer, üçer ve dörder işlem kullanılması halinde sırasıyla 1.14, 1.20 ve 1.24 kat hızlanma olarak gözlenmiştir. Şekil 4-10'da ise bu başarım biraz daha azalarak 1.07, 1.10 ve 1.14 kat olarak gerçekleşmiştir. Bunun nedeni altyapı sayısı arttıkça daha da büyüyen sınır denklem çözüm süresinin toplam çözüm süresi içerisindeki oranın artmasıdır. Örneğin iki adet altyapı kullanıldığında yaklaşık 3000 denklem içeren sınır problemi, sekiz altyapı kullanıldığında yaklaşık 13000 denkleme sahip bir sisteme dönüşmektedir. Bu artışla sınır denklem çözümünün toplam çözüm süresi içindeki oranı %4'den %55'e çıkmıştır. Bu nedenle denklem oluşturma ve indirgeme işlemi sırasında hızlanmayla elde edilen avantaj, altyapı sayısı arttıkça sınır denklemlerinin çözümünde harcanan vakit nedeniyle azalmaktadır.

Toptan ve Altyapı Tabanlı Çözüm Algoritmalarının Karşılaştırılması

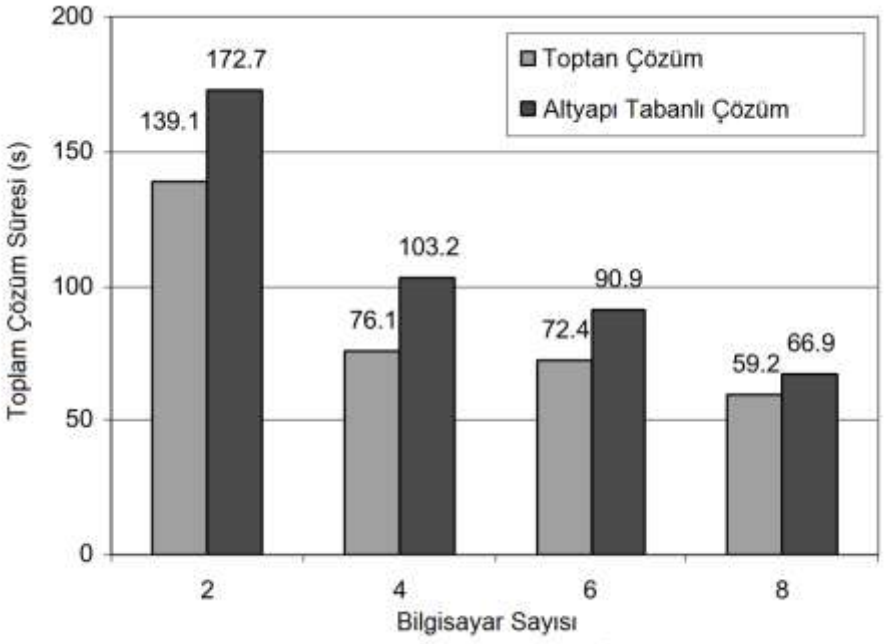
Önceki bölümlerde anlatılan toptan ve altyapı tabanlı paralel çözüm algoritmaları değişik modeller kullanılarak HPCE2 isimli homojen bir bilgisayar kümesi üzerinde karşılaştırılmıştır. Algoritmaların karşılaştırmasında altyapı tabanlı algoritmanın performans testlerinde kullanılan üç model, küp, nükleer atık arıtma tesisi ve otel modelleri (Şekil 4-3), kullanılmıştır. Modeller homojen bilgisayar kümesinde iki, dört ve sekiz bilgisayar kullanılarak çözülmüştür. Çözümler sırasında her bilgisayardan sadece tek çekirdek kullanılmıştır. Bunun sebebi algoritmaların bilgisayar kümesi üzerinde dağıtık paralel performanslarının ölçülmek istenmesidir. Karşılaştırılmalarda model bilgisi hazırlama, denklem sıralama, direngenlik matrisi hazırlama, model deplasmanlarının hesaplanması ve hesaplanan deplasmanların Panthalassa veri yapısına aktarılmasını içeren toplam çözüm süreleri kullanılmıştır. Giriş dosyasından Panthalassa veri yapısının oluşturulması ve bu veri yapısının bilgisayarlar arasındaki paylaşımında harcanan süreler karşılaştırılmaların dışında tutulmuştur. Bu şekilde sadece iki algoritmanın analiz performanslarına odaklanılmıştır.

Şekil 4-11'de küp modeli için toplam çözüm süreleri sergilenmiştir. Küp simetrik ve tekdüze bir model örneğidir. Bu modelde toptan çözüm algoritması altyapı tabanlı çözüm algoritmasını göre önemli ölçüde hız farkı yaratmıştır. İki bilgisayar kullanılan testde toptan çözüm algoritması altyapı tabanlı algoritmadan 2.67 kat daha hızlıdır. Bu fark bilgisayar sayısı arttıkça azalmakta, sekiz bilgisayarlı testde 1.45 kata düşmektedir. Direngenlik matrisi hesaplama süresi iki algoritmada da toplam çözüm süresinin %15 - %20 civarında bir kısmını almıştır. Toptan çözüm algoritmasında sekiz bilgisayarlı çözüm iki bilgisayarlı çözüme göre 2.7 kat daha hızlıdır.

Nükleer atık arıtma tesisi modeli simetrik olsa da küp modelinden farklı olarak değişik eleman tiplerini barındırmaktadır. Şekil 4-12 bu model için toplam çözüm sürelerini göstermektedir. Bu modelde direngenlik matrisinin seyrekliği önceki modeldeki gibi tekdüze değildir ve toptan çözüm algoritması ile altyapı tabanlı çözüm algoritması arasındaki hız farkı küp modelindekine göre azdır. İki bilgisayar kullanılan testte toptan çözüm algoritması altyapı tabanlı çözüm algoritmasından 1.24 kat daha hızlıdır. Bu hız farkı sekiz bilgisayarlı testde 1.13 kata kadar azalmıştır. Altı bilgisayar kullanılan testde toptan çözüm algoritması direngenlik matrisinin çarpanlara ayrılması ve çözüm adımlarında dört bilgisayarlı teste göre daha yavaş çalışmıştır. Bu gözlem algoritmanın ikinin katı bilgisayar sayısına göre optimize edildiğini göstermektedir. Altı bilgisayarlı testde direngenlik matrisi hesaplama süresi dört bilgisayarlı teste göre çok daha az olduğu için toplam çözüm süresinde altı bilgisayarlı test dört bilgisayarlı teste göre daha hızlı olmuştur. Direngenlik matrisi hesaplama süresi toplam çözüm süresinin dörtte biri olarak gerçekleşmiştir. Toptan çözüm algoritması sekiz bilgisayarın kullanıldığı testde iki bilgisayarın kullanıldığı teste göre 2.35 kat hızlıdır.



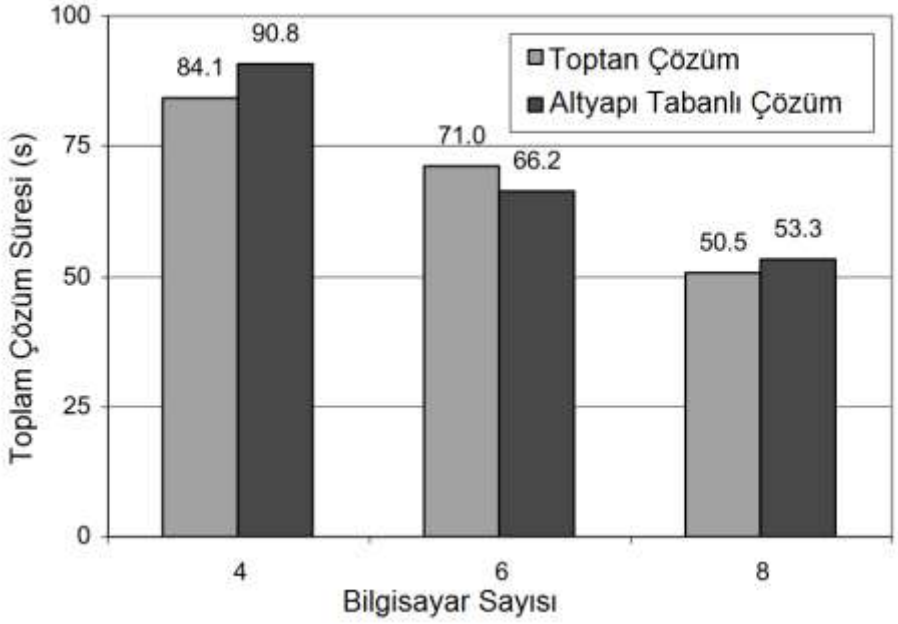
Şekil 4-11 Küp Modeli Toplam Çözüm Süreleri



Şekil 4-12 Nükleer Atık Arıtma Tesisi Modeli Toplam Çözüm Süreleri

Şekil 4-13'te otel modeli için toplam çözüm süreleri gösterilmiştir. Bu modelde iki çözüm yöntemi için de iki bilgisayarla çözüm RAM bellek yetmezliği nedeniyle gerçekleştirilememiştir. Daha fazla bilgisayar kullanılan testlerde iki çözüm algoritması arasındaki hız farkı gözardı edilebilir seviyededir. Dört bilgisayar kullanıldığında toptan çözüm yöntemi altyapı tabanlı çözüm yöntemine göre sadece 1.08 kat hızlıdır. Modellerde artan düzensizlik altyapı tabanlı çözücünün performans olarak toptan çözücüye yaklaşmasını

sağlamıştır. Bu modelde altyapı tabanlı çözücünün performansını toptan çözücününkine yaklaştıran bir diğer sebep de direngenlik matrisi hesaplama süresinin toplam çözüm süresinin %30 - %35'i kadar olmasıdır.

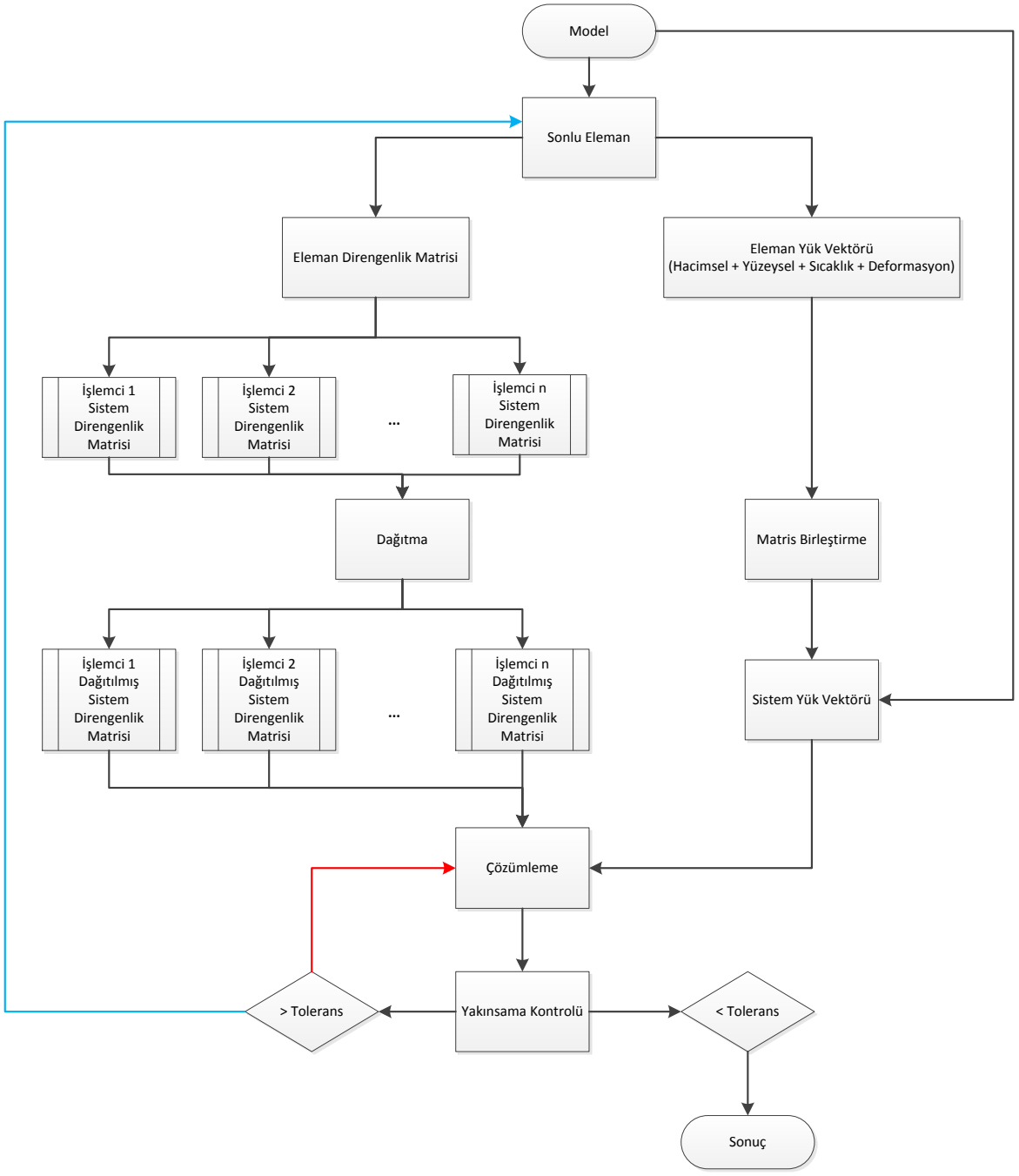


Şekil 4-13 Otel Modeli Toplam Çözüm Süreleri

Sonuç olarak toptan çözüm yöntemi özellikle simetrik ve tektip yapı modellerinde altyapı tabanlı çözüm yöntemine göre daha fazla performans sağlamıştır. Simetrinin ve tektipliğin azaldığı modellerde altyapı tabanlı çözücü toptan çözücüyle aradaki çözüm süresi farkı azalmaktadır.

Doğrusal Olmayan Statik Çözümleme Algoritması

Doğrusal olmayan statik çözümleme algoritmasınının paralelleştirilmesi işleminde de toptan çözümleme yöntemi kullanılmıştır. Paralelleştirilen bu algoritmada doğrusal olmayan malzeme ve doğrusal olmayan geometri problemleri Standart Newton Raphson ve Geliştirilmiş Newton Raphson metotları kullanılarak çözülmektedir. Her iki çözümleme metodu için de yük kontrollü çözümleme mümkündür. Toptan çözümleme yöntemi ile geliştirilmiş doğrusal olmayan statik çözümleme algoritmasının başarımı ilerleyen bölümlerde incelenmiştir.



Şekil 4-14 Doğrusal Olmayan Statik Çözümleme Algoritması

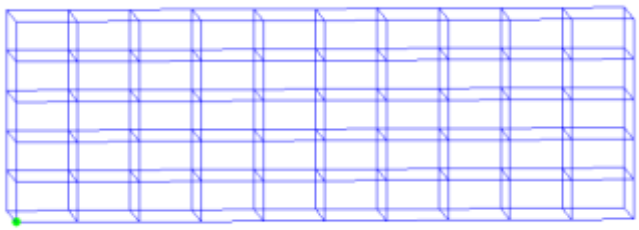
Şekil 4-14'te doğrusal olmayan statik çözümleme algoritmasının paralelleştirilmiş halinin akış diyagramı gösterilmiştir. Doğrusal olmayan statik çözümleme algoritması paralelleştirilirken MUMPS kütüphanesindeki algoritmalardan yararlanılmıştır. Seri doğrusal olmayan statik çözümleme algoritmasında olduğu gibi sonlu eleman kütüphanesindeki elemandan eleman direngenlik matrisi ve eleman yük vektörü alınır. Doğrusal olmayan çözümleme algoritmasında paralel matris yöntemi ile

sistem direngenlik matrisi seyrek matris olarak tüm bilgisayarlarda tutulması ve bu seyrek matrisin MUMPS kütüphanesindeki algoritmalar yardımı ile tüm işlemcilere eşit yük gelecek şekilde dağıtılması sağlanmıştır. Sistem yük vektörü ise tek bir işlemcide matris birleştirme yöntemi kullanılarak oluşturulmuş ve tutulmuştur. Oluşturulan sistem direngenlik matrisi ve sistem yük vektörü MUMPS kütüphanesi yardımı ile çözümlenir.

Seri doğrusal olmayan statik algoritmanın yapısında bulunan yakınsama kontrolü ve yeniden çözümleme işlemleri bu algoritma için de aynen kullanılmıştır.

Doğrusal Olmayan Statik Çözümleme Algoritmasının Test Edilmesi

Doğrusal olmayan statik çözümleme algoritmasının çok bilgisayarlı sistemlerdeki performansını incelemek amacı ile Şekil 4-15'te gösterilen düz konsol kiriş problemi doğrusal olmayan malzeme modeli ve farklı eleman ağ sayısında Brick8 elemanı kullanılarak çözülmüştür.



Şekil 4-15 Düz Konsol Kiriş Problemi

Model özellikleri aşağıdaki gibidir.

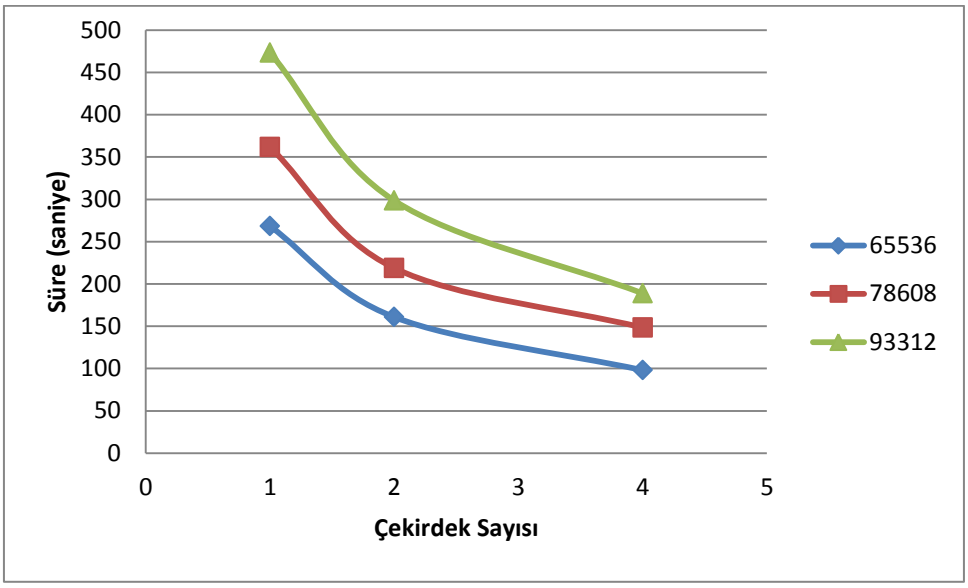
Geometrik Özellikler:	Malzeme Özellikleri:	Kesit Özellikleri:
Uzunluk: 8	E: 1728	Yükseklik: 2
Genişlik: 1	v: 0.2	
	Sy: 14400	
	Et = 1000	

Yükleme Özellikleri:

Yükleme: 40 (kirişin serbest ucunda -z yönünde)

Yukarıda özellikleri verilen kiriş problemi çözümünde Intel i7 2.67GHz işlemcili ve 6GB RAM içeren bilgisayar kullanılmıştır. Bu bilgisayarda Windows 7 işletim sistemi bulunmaktadır.

Şekil 4-15'te gösterilen düz konsol kiriş problemi sırasıyla 65536, 78608 ve 93312 adet Brick8 elemanı ve 1, 2 ve 4 çekirdek kullanılarak çözümlenmiştir. Algortimanın eleman sayısı - çekirdek performansı Şekil 4-16'da gösterilmiştir.



Şekil 4-16 Doğrusal Olmayan Statik Çözümleme Algoritmasının Eleman Sayısı Süre İlişkisi

Şekil 4-16'daki değerler incelendiğinde doğrusal olmayan çözümleme algoritmasını paralel sistemlerde kullanmanın avantajlı olduğu görülmektedir. Kullanılan çekirdek sayısı arttıça çözümleme süresi azalmaktadır. Farklı eleman ağı ile oluşturulmuş konsol kiriş modelinin iki ve dört çekirdek kullanımıyla çözümlemenin sırasıyla ortalama 1.64 ve 2.56 kat hızlandığı görülmektedir.

Zaman Alanında Dinamik Çözümleme - Örtük Tümleştirme Yöntemleri

Alt-yapı Tabanlı Çözücülerle Paralel Örtük Doğrusal Dinamik Çözümleyicisi

Örtük doğrusal dinamik çözümleme temelde sistemin farklı zaman adımlarında maruz kaldığı etkiler altında çözümlenmesinden oluşmaktadır. Bu nedenle aslında her bir zaman adımında daha önceden geliştirilmiş olan altyapı tabanlı paralel doğrusal statik çözümleyici kullanılabilir.

Çözüm işlemi her bir bilgisayarın girdi dosyasını okuduktan sonra serbestlik derece denklemlerini numaralandırmasıyla başlamaktadır. Bu numaralandırmayı kullanarak her bir bilgisayar sorumlu olduğu alt yapı sisteminin direngenlik matrisini ve yük vektörünü oluşturur. Bunlara ek olarak, çözüme başlamadan önce kütle ve sönümleme matrislerinin de oluşturulması gerekmektedir. Bu matrisler hafıza kullanımını asgari düzeye indirme amacıyla eleman seviyesinde oluşturulmaktadır. Eleman kütle matrisleri yapının ihtiyaçlarına göre elemanların kütlesinin düğüm noktalarına eşit olarak dağıtılmasıyla (lumped) veya yayılmış (consistent) olarak oluşturulabilinir. Eleman sönümleme matrisi ise Rayleigh tarafından önerildiği şekilde eleman direngenlik matrisiyle, eleman kütle matrisinin doğrusal birleştirilmesiyle oluşturulmaktadır (Cook 2001).

Yukarıda bahsedildiği üzere dinamik çözüm için yapının dinamik direngenliği matrisinin, $[\overline{K}]$, bir kereye mahsus ayrıştırılması yeterlidir. Bu işlem, altyapı tabanlı doğrusal çözümleyici tarafından iki aşamada gerçekleştirilmektedir. İlk asama altyapının içerisinde kalan noktalardaki etkilerin statik indirgeme metodu kullanılarak sınır noktalarına aktarılmasından oluşmaktadır. İndirgeme işlemi LDL^T metodu kullanan bir paralel doğrudan çözücü olan MUMPS tarafından yapılmaktadır (Amestoy et al. 2000). Çözümde bu aşamanın sonuna kadar bilgisayarlar arasında herhangi bir veri alışverişine veya senkronizasyona ihtiyaç yoktur.

İkinci aşamada, her bilgisayarın kendi sorumluluğunda olan alt yapı matrisini sınır noktalarına indirgenmesinden sonra sınır noktaları denklem sistemini oluşturmak amacıyla bilgisayarlar veri alışverişinde bulunur. Ardından da oluşturulan sınır denklemler sistemi ScaLAPACK kütüphanesinin LDL^T yöntemi kullanan fonksiyonları aracılığıyla paralel olarak ayrıştırılır.

Benzer şekilde o zaman adımına ait yük vektörü de her bir zaman adımında alt yapı seviyesinde hesaplanır, sınır noktalara indirgenir ve diğer alt yapılardan gelen etkilerle birleştirilerek sınır nokta yüklemeleri elde edilir. Sonrasında bu sistemin çözümü gerçekleştirilir ve sınır noktalar seviyesinde yer değiştirmeler elde edilir. Bu yer değiştirmeler kullanılarak her bilgisayar kendi sorumluluğundaki altyapıya ait diğer yer değiştirmeleri hesaplar. Nokta yer değiştirmelerine ek olarak her bir zaman adımında eleman yükleri ve gerilmeleri de hesaplanabilir. Böylelikle bir sonraki zaman adımına geçmek için gereken bütün değerler elde edilmiş olur.

Toptan Çözücü Kullanarak Paralel Örtük Doğrusal Dinamik Çözümleyicisi

Yukarıda da anlatıldığı üzere örtük doğrusal dinamik çözümleme esas olarak sistemin farklı zaman adımlarında maruz kaldığı etkiler altında çözümlenmesinden oluşmaktadır. Bu nedenle aslında her bir zaman adımında daha önceden geliştirilmiş olan toptan doğrusal statik çözümleyici de bu algoritma için kullanılabilir.

Bu yaklaşımda ise yapı altyapılara ayrıştırılmadan toptan olarak işleme tabi tutulmaktadır. Her bilgisayar girdi dosyasını yükledikten sonra, kendi aralarında sırayla dönüşümlü olarak dağıtılan elemanların direngenlik, kütle ve sönümleme matrislerini oluşturur ve bunları bütün yapıya ait olan matris içerisine uygun şekilde yerleştirir. Sonrasında bütün bilgisayarlar paralel olarak bütün yapıya ait olan matrisi ayrıştırır. Her bir zaman adımında da 3 numaralı denkliğin sağ tarafında kalan yapının o zaman adımındaki yüklemesini içeren vektörü hesaplayıp yine paralel olarak çözümü gerçekleştirirler. Bu yaklaşımda bahsi geçen bütün bu ayrıştırma ve çözüm işlemleri MUMPS çözücüsü tarafından gerçekleştirilmektedir.

Karşılaştırma Sonuçları

Yukarıda bahsi gecen, örtük dinamik yapı çözümlemesi için geliştirilmiş olan çözücülerin başarımının sınanması amacıyla örnek bir model HPCE2 isimli homojen bir bilgisayar kümesinde çözümlenmiştir.

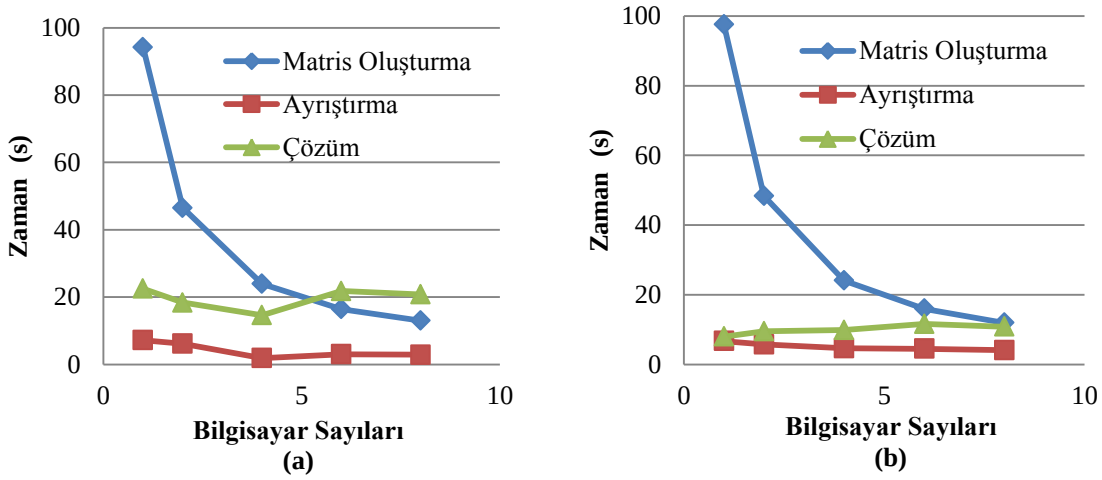
Örnek olarak hazırlanan model (Şekil 4-17), 40000 adet dörtgen plaka elemanından oluşmaktadır. Toplamda 40,401 noktası ve yaklaşık 130,000 denklem içermektedir. Yapıya farklı zaman adımlarında yüklemeler uygulanmış ve yapının bu yükleme altında davranışı 20 saniye boyunca 0.02 saniye aralıklarla hesaplanmıştır.

Şekil 4-17 Örnek Bölümleme Sonrası Kare Plak Modeli

Şekil 32, örnek modelin alt-yapı tabanlı doğrusal dinamik çözücü ve toptan doğrusal dinamik çözücünün 1, 2, 4, 6 ve 8 işlemci kullanılarak çözümlenmesiyle elde edilmiş zamanlamaları göstermektedir. Bu şekillerde, doğrusal dinamik çözüm metodunun üç ana işlevi olan matris oluşturma, ayrıştırma ve çözüm safhaları ayrı ayrı incelenmiştir.

Her iki çözücü için de matris oluşturma sürelerinin artan işlemci sayısıyla doğru orantılı olarak azaldığı görülmektedir. Örneğin, alt-yapı tabanlı çözücü ile tek işlemcide 94,6 saniyede gerçekleştirilen matris oluşturma iki ve dört işlemcide 46.1 ve 24.3 saniyede tamamlanmıştır.

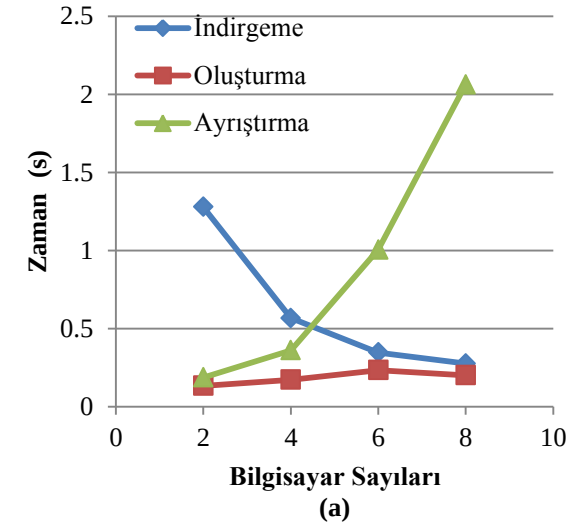
Statik çözümlerde çözüm süresinin büyük bir kısmını ayrıştırma işlemi oluşturmaktadır. Ancak bu örnek dinamik çözümdeki gibi bin adet zaman adımında yapılan çözümle karşılaştırılınca, bir kerelik yapılan ayrıştırma işleminin toplam çözüm süresi içerisindeki payı daha azdır. Ayrıca, Şekil 4-18'de de görülebileceği üzere ayrıştırma işlemi artan bilgisayar sayısıyla birlikte daha hızlı gerçekleştirilebilmektedir.

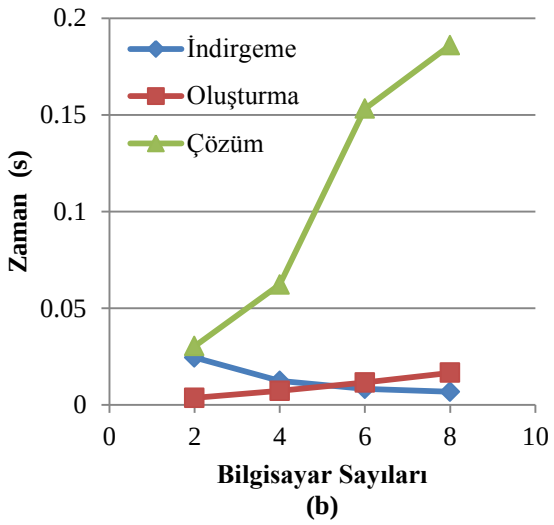


Şekil 4-18 Alt-yapı tabanlı (a) ve toptan (b) çözücülerin üç ana işlem için ihtiyaç duyduğu süreler

Şekil 4-19'da ise alt-yapı tabanlı çözücünün alt-yapı matris ve vektörlerini sınır düğüm noktalarına indirgemesinin, bu matris ve vektörleri oluşturmasının ve sonrasında da ayrıştırmasının veya çözmesinin zamanlamaları verilmiştir. Görüldüğü üzere indirgeme işlemi artan bilgisayar sayısıyla doğru orantılı olarak hızlanmaktadır. Ancak, matris oluşturma ve ayrıştırma işlemiyle vektör oluşturma ve sınır denklem çözümü işlemleri tam tersine daha uzun sürede tamamlanabilmektedir.

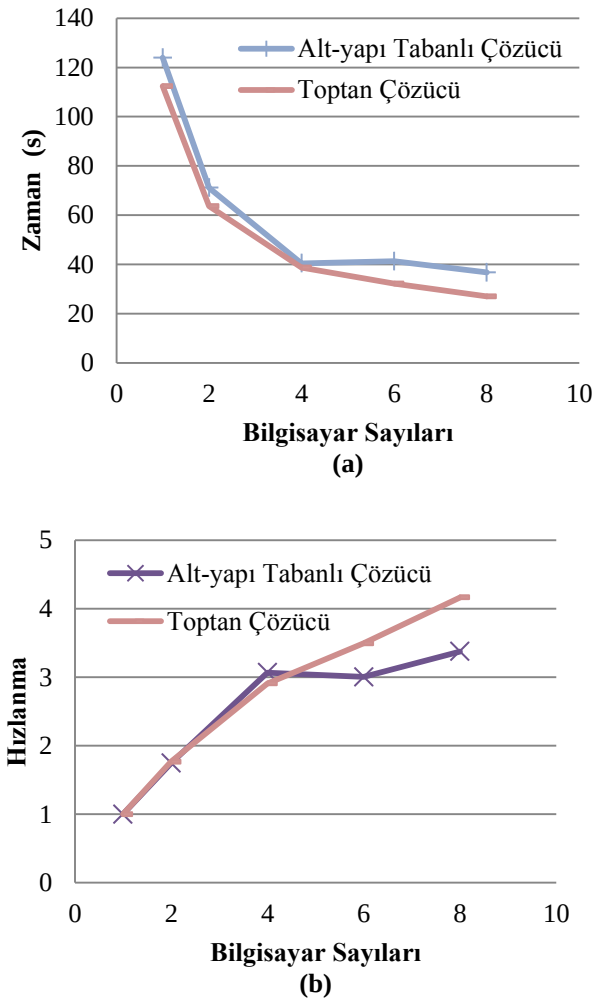
Artan alt-yapı sayısıyla artan sınır düğüm sayısı düşünüldüğünde bu doğal bir gözlemdir. Çünkü daha büyük bir sınır denklem sistemini oluşturmak ve çözümlemek için bilgisayarlar arasında daha fazla iletişime ve daha fazla işleme ihtiyaç duyulmaktadır. Sonuç olarak alt-yapılara bölümlemeyle indirgeme süresinden kazanılırken sınır denklem süresinden kaybedilmektedir. Ancak indirgeme çoğu zaman toplam çözüm süresinin daha büyük bir yüzdesini oluşturduğu için toplamda çoğu zaman daha hızlı bir çözüm süresi elde edilmektedir.





Şekil 4-19 Alt-yapı tabanlı çözücünün sınır denklem matrisi(a) ve vektörleri(b) işlem süreleri

Son olarak alt-yapı tabanlı çözücü ile toptan çözücünün örnek modeli farklı sayıda işlemci kullanarak dinamik çözümlemeleri sonucu elde edilen toplam süreleri Şekil 4-20 (a)'da verilmiştir. Alt-yapı tabanlı ve toptan çözücünün tek işlemci kullanarak yaptıkları dinamik çözümler sırasıyla 123.1 ve 112.8 saniyede gerçekleştirilmiştir. Alt-yapı tabanlı çözücünün iki, dört ve sekiz işlemci kullanmasıyla toplam çözüm süresi 71.0, 43.6 ve 31.9 saniyeye düşmüştür.



Şekil 4-20 Alt-yapı tabanlı ve toptan çözücülerin toplam sürelerinin (a) ve hızlanmalarının (b) karşılaştırması

Paralel çözüm sürelerinin tek işlemci kullanılarak gerçekleştirilen çözüm süresine bölünmesiyle hesaplanan hızlanma değerleri ise Şekil 4-20 (b)'de yer almaktadır. Alt-yapı tabanlı çözücü için iki, dört, altı ve sekiz işlemci ile çözümler için 1.74, 3.06, 2.90 ve 3.48 oranında bir hızlanma elde edilirken, toptan çözücü aynı işlemci sayılarında 1.76, 2.91, 3.49 ve 4.16 oranında hızlanma sergilemiştir.

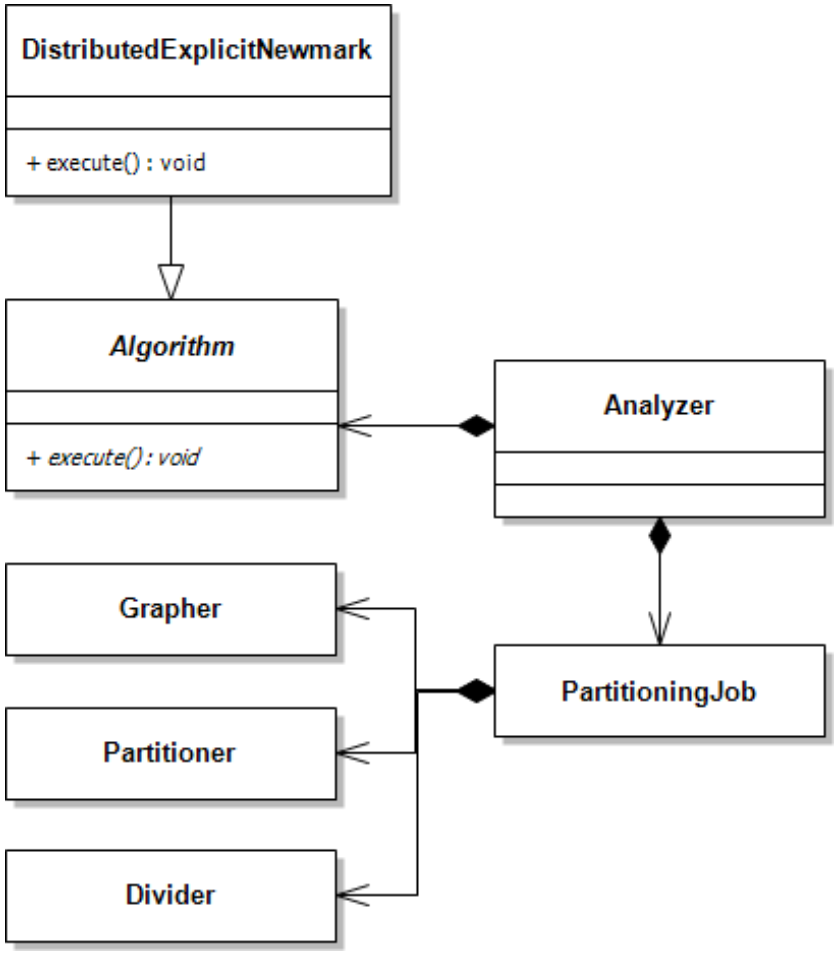
Zaman Alanında Dinamik Çözümleme – Belirtik Tümleştirme Yöntemleri

Üçüncü bölümde detayları anlatılan ve tek bilgilsayarda için geliştirilmiş olan, doğrusal ve doğrusal olmayan problemlerin çözümünde kullanılabilen, Newmark Belirtik Çözümleme Algoritmasının paralel bilgisayarlar üzerinde çalışacak şekilde yeni bir uygulaması yapılmıştır. Bu uygulamada kullanılan algoritma esasen üçüncü bölümde kullanılan algoritmanın aynısıdır. Seri algoritmanın paralel hale getirilmesi sonlu elemanlar modelinin bölümlenmesi esasına dayanır. Paralel çalışan her işlemci için sonlu elemanlar modelinden bir bölüm ayrılır. Her işlemci kendi bölümünde üçüncü bölümde ayrıntılanan seri

algoritmayı çalıştırır. Her zaman adımın sonunda tüm işlemcilerin sonuçları birleştirilir ve analize bir sonraki zaman adımıyla devam edilir.

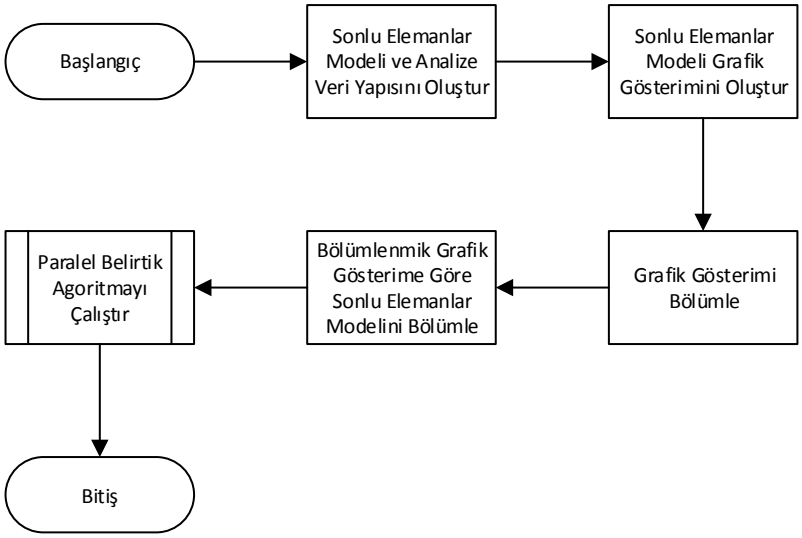
Paralel Belirtik Dinamik Çözümleme Algoritması Uygulanışı

Belirtik algoritmanın paralel uygulaması *DistributedExplicitNewmark* Sınıfı ile yapılmıştır. Bu sınıf Panthalassa plug-in mantığı ile Panthalassa sistemine eklenmiştir. Şekil 4-21, *DistributedExplicitNewmark* sınıf diyagramını sergiler. *DistributedExplicitNewmark* Sınıfı *Algorithm* Sınıfında türetilmiştir. Sonlu elemanlar modeli bölümleme işlemi için Panthalassa'nın *Grapher*, *Partitioner* ve *Divider* sınıfları kullanılmıştır. Giriş dosyasında kullanıcı modelin bölümlenmesi için hangi bölümleme algoritmalarını dolayısıyla hangi *Grapher*, *Partitioner* ve *Divider* uygulamalarını kullanacağını belirtir. Panthalassa bu uygulamaları kullanarak bölümleme işlemini yapar. Paralel Belirtik Dinamik Algoritma bölünmüş sonlu elemanlar modeli üzerinde paralel olarak çalışır ve çözümlemeyi gerçekleştirir.



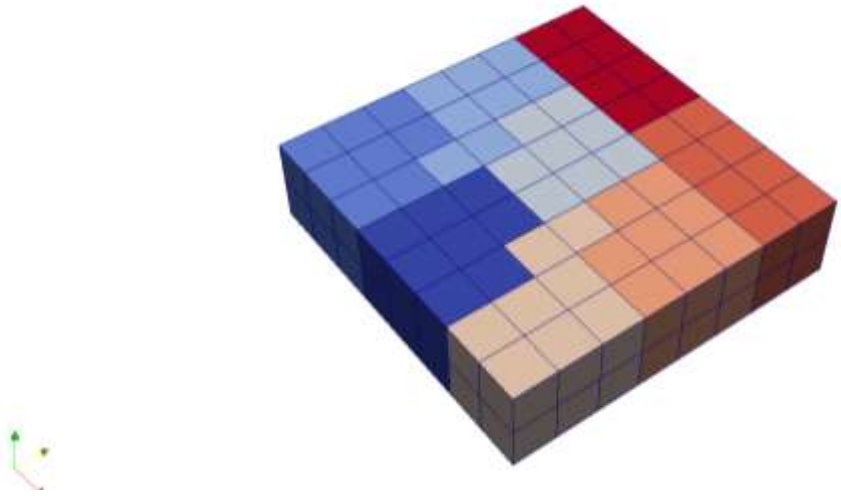
Şekil 4-21 *DistributedExplicitNewmark* Sınıf Diyagramı

Şekil 4-22'de paralel belirtik analizin basamakları gösterilmiştir. Analizin ilk basamağı giriş dosyasında sonlu elemanlar modeli ve analizi veri tabanının oluşturulmasıdır. *ModelBuilder* Sınıfı ile yapılan bu işten sonra *Grapher* Sınıfı kullanılarak sonlu elemanlar modelinin grafiksel gösterimi oluşturulur. *Partitioner* Sınıfı ile grafiksel gösterim bölümlenir ve *Divider* Sınıfı ile bölümlenmiş grafiksel gösterimden bölümlenmiş sonlu elemanlar modeli oluturulur. Son olarak Paralel Belirtik Çözümleme Algoritması bölümlenmiş sonlu elemanlar modeli üzerinde çalıştırılır.



Şekil 4-22 Paralel Beliritk Analiz Basamakları

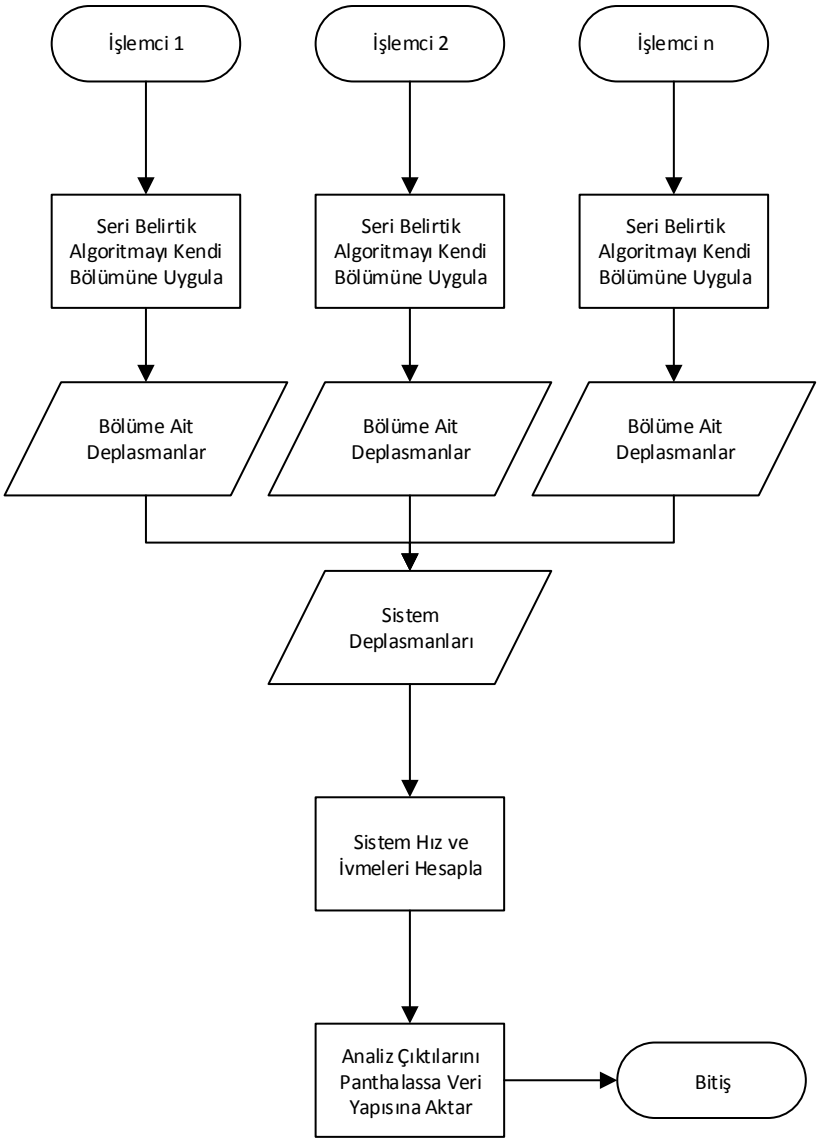
Paralel Belirtik Çözümleme Algoritmasının testlerinde grafiksel gösterimin bölümlenme işleminde ParMETIS kütüphanesi kullanılmıştır. ParMETIS kütüphanesi grafiksel gösterimleri paralel olarak parçalara ayıran bir bilgisayar kütüphanesidir. Şekil 4-23, ParMETIS Kütüphanesi kullanılarak 8 parçaya ayrılmış bir sonlu elemanlar modelini gösterir.



Şekil 4-23 ParMETIS Kütüphanesi Kullanılarak Parçalanmış Sonlu Elemanlar Modeli

Şekil 4-24'te Paralel Belirtik Çözümleme Algoritmanın bölümlenmiş sonlu elemanlar modeli üzerinde çalışma şekli gösterilmiştir. Paralel algoritmada her işlemci kendisine düşen bölüm üzerinde seri

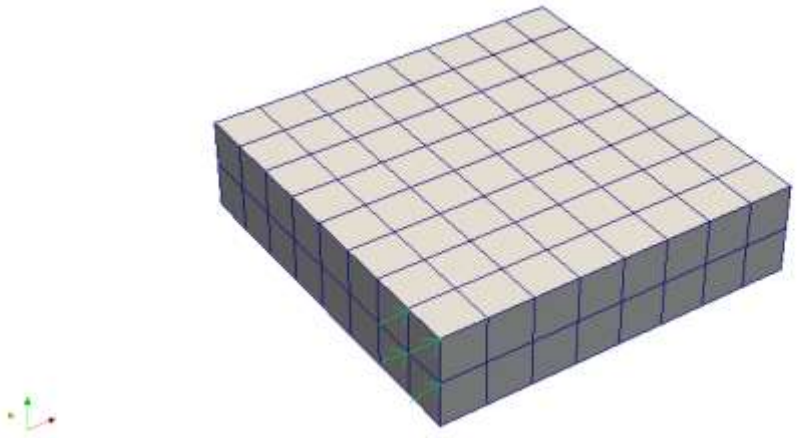
algoritmayı uygular. Bu işlem sonucunda ortaya çıkan bölüm deplasmanları birleştirilerek sistem deplasmanları oluşturulur. Deplasmanlardan hız ve ivmeler hesaplanır ve son olarak sonuçlar Panthalassa veri yapısına aktarılır.



Şekil 4-24 Paralel Belirtik Çözümleme Algoritması İşleyiş Şekli

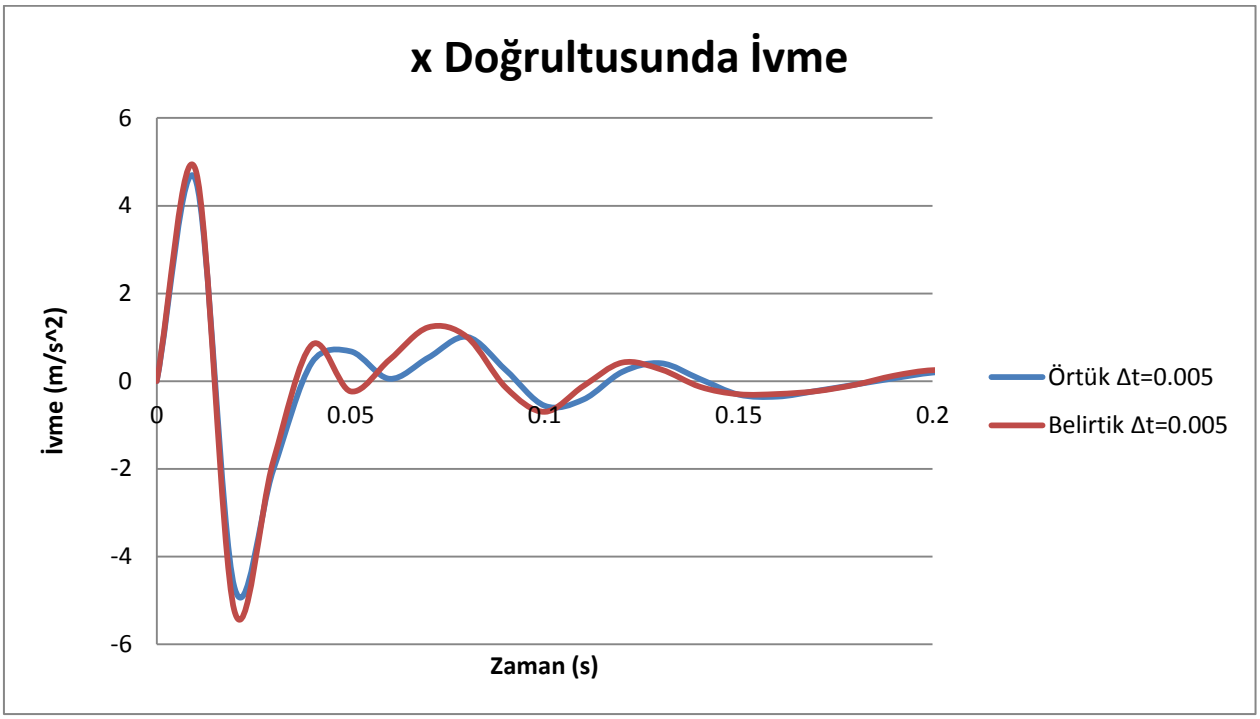
Zaman Alanında Paralel Dinamik Çözümleme Algoritmalarının Doğrulaması

Paralel Belirtik Dinamik Çözümleme Algoritmasının doğrulanması için örnek bir problem seçilip örtük dinamik çözümleme algoritmasıyla karşılaştırılmış, daha sonra çözüm paralel çözümleme sistemin doğrulanması için paralel olarak tekrarlanmıştır.

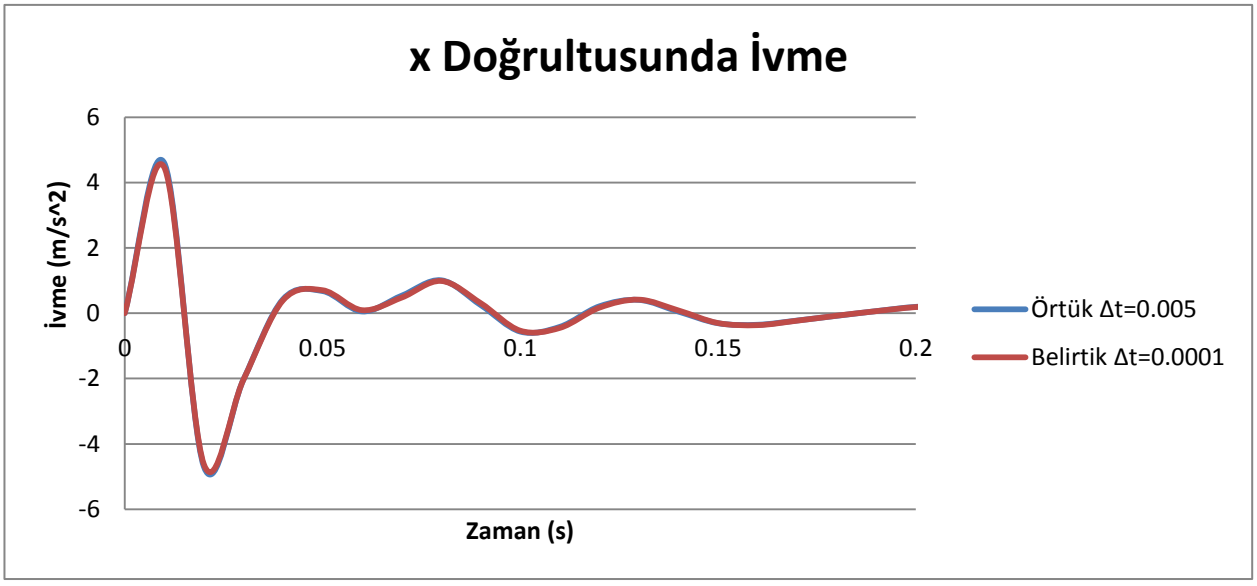


Şekil 4-25 Doğrulamada Kullanılan Sonlu Elemanlar Modeli

Şekil 4-25'te gösterilen sonlu elemanlar modeli tabandan sabitlenmiş ve üst köşesindeki elemanın düğüm noktalarından x yönünde anlık olarak yüklenmiştir. Şekil 4-26 ve Şekil 4-27 modelin köşe noktasının örtük ve belirtik algoritmalarla hesaplanmış x yönündeki ivme değişimini gösterir. Belirtik algoritmada kullanılan zaman adımı düşürüldükçe belirtik algoritma sonuçları örtük algoritmaya yaklaşmaktadır. Belirtik algoritmanın zaman adımı 0.0001 saniyeye indiğinde iki algoritma hemen hemen aynı sonuçları vermektedir. Sonuç olarak örtük algoritmalarda sistem matrislerinin oluşturulması ve ayrıştırılması gerekmekte ve bu durum paralel hesaplama açısından ciddi bir veri bağımlılığı yaratmakta ancak daha büyük zaman adımları kullanılarak gerçek çözüme yakın sonuçlar elde edilebilmektedir. Belirtik algoritmada ise bütün hesaplar eleman düzeyinde gerçekleştirildiği dolayısıyla sistem matrislerinin oluşturulması ve ayrıştırılması gerekmediği için veri bağımlılığı dolayısıyla bilgisayarlar arası veri alışveriş gerekliliği sadece alt-yapı sınırlarında mevcuttur. Ancak aşağıda gösterilen sonuçlardan da görülebileceği gibi gerçek sonuca yakınsamak için örtük sonuca nazaran çok daha küçük zaman adımları kullanılmasına ihtiyaç vardır.



Şekil 4-26 Örtük ve Belirtik Dinamik Algoritmalarının Karşılaştırması (Δt=0.005)



Şekil 4-27 Örtük ve Belirtik Dinamik Algoritmalarının Karşılaştırması

Belirtik algoritmanın paralel olarak da doğru çalıştığını göstermek için aynı test 1,2 ve 4 bilgisayarda tekrarlanmıştır. Tablo 4-1'de görüldüğü gibi algoritma paralel olarak aynı sonuçları vermektedir.

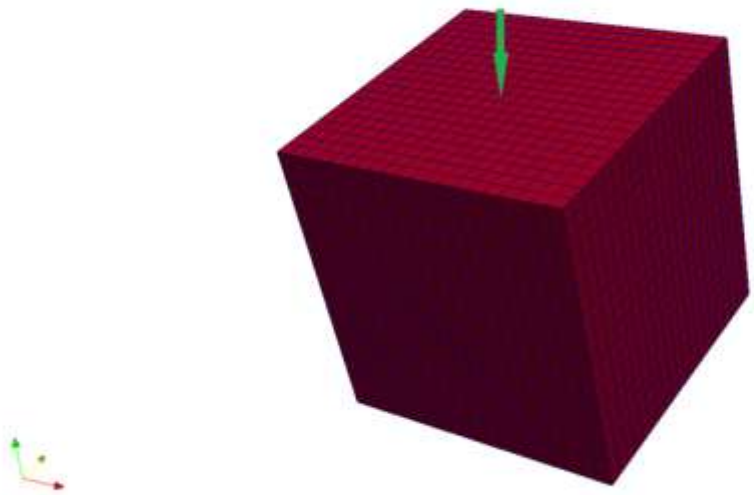
Tablo 4-1 Paralel Belirtik Çözümleme İvme Değerleri

	Bilgisayar Sayısı		
Zaman (s)	***1***	***2***	***4***
0.01	4.59489	4.59489	4.59489
0.2	0.195726	0.195726	0.195726
0.4	-0.02326	-0.02326	-0.02326
0.6	-0.03763	-0.03763	-0.03763
0.8	0.015178	0.015178	0.015178
1	-0.00526	-0.00526	-0.00526

Paralel Belirtik ve Toptan Örtük Dinamik Algoritmaların Başarım Testleri

Dinamik Doğrusal Çözümleme

Bu bölümde Belirtik ve örtük algoritmalarla yapılan analizlerin değişik sayıda işlemci kullanılarak ulaşılan çözüm zamanlamaları iki farklı bilgisayar kümesinden elde edilen sonuçlarla karşılaştırılmıştır. HPCE2 isimli ilk bilgisayar kümesi sekiz Intel Core2 Quad işlemcili ve 3GB RAM içeren bilgisayardan oluşmuştur (Eski bilgisayar kümesi). Bilgisayarlarda Windows XP işletim sistemi kurulu olup, 1Gbit hızında bir ağ anahtarı ile birbirlerine bağlanmışlardır. Yeni bilgisayar kümesi ise altı bilgisayardan kurulu olup, her bilgisayarda iki tane dört çekirdekli Intel Xeon işlemci ve 24GB RAM vardır. Bilgisayarlar Infiniband teknolojisiyle birbirlerine bağlanmıştır.



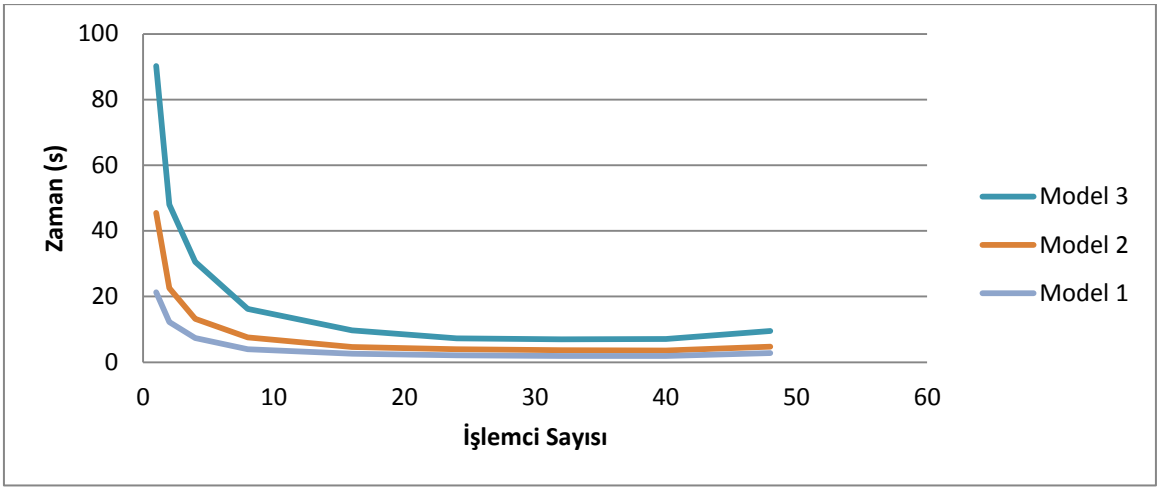
Şekil 4-28 Örtük ve Belirtik Algoritmaların Karşılaştırılmasında Kullanılan Model

Şekil 4-28'de görülen küp şeklindeki toprak modeli orta noktasından verilen anlık bir kuvvet ile yüklenmiş ve 0.01 saniyelik zaman aralıkları ile 1 saniye boyunca analiz edilmiştir. Analizlerde değişik eleman boyutları kullanılarak geliştirilmiş yedi farklı model kullanılmıştır (Tablo 4-2).

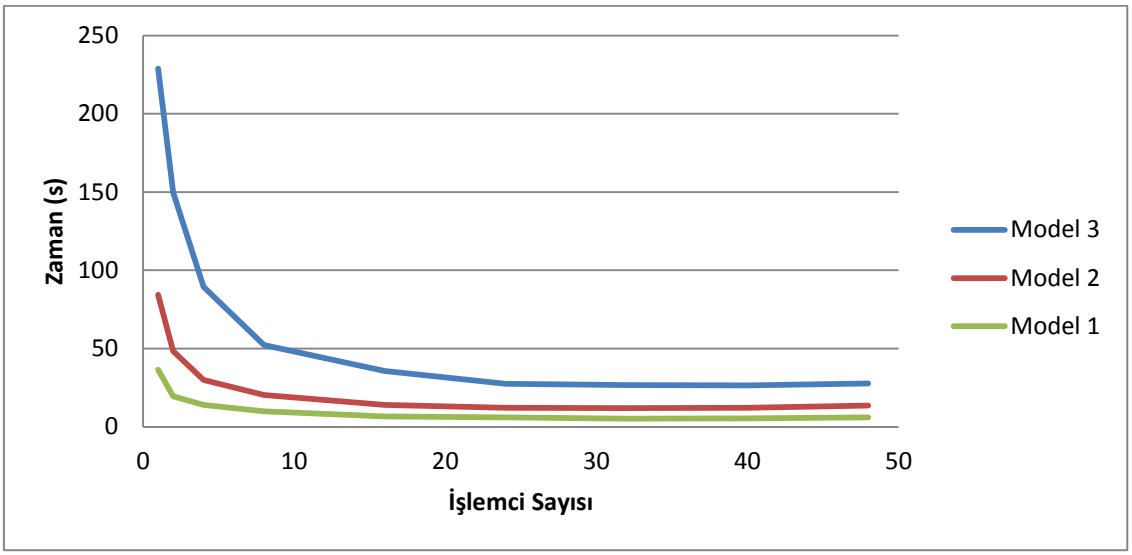
Tablo 4-2 Karşılaştırmada Kullanılan Modeller

Model	Eleman Sayısı	Denklem Sayısı
Model 1 (25x25x25)	15625	52728
Model 2 (50x25x25)	31250	103428
Model 3 (40x40x40)	64000	206763
Model 4 (50x50x50)	125000	397953
Model 5 (100x40x40)	160000	509343
Model 6 (200x40x40)	320000	1013643
Model 7 (100x100x100)	1000000	3090903

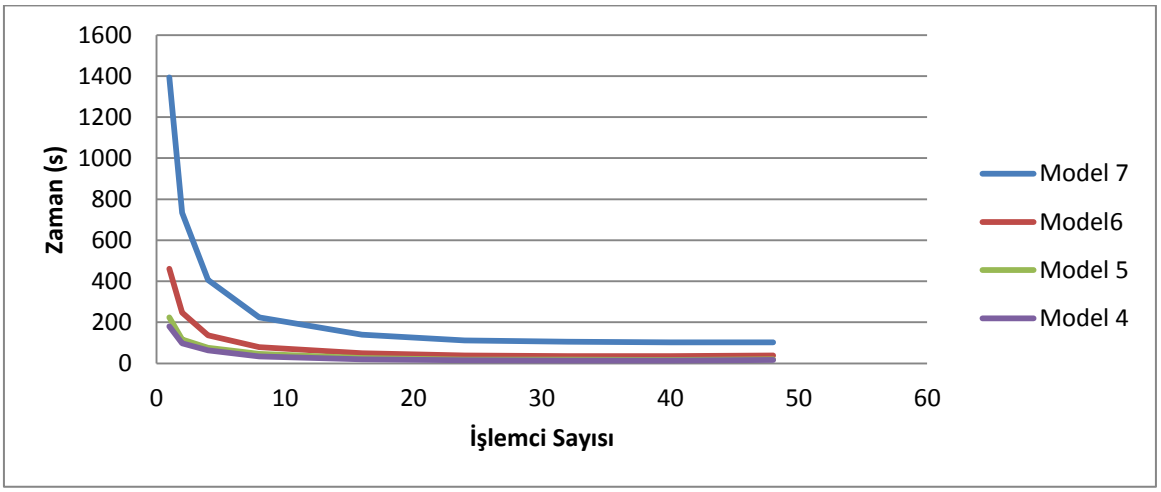
Şekil 4-29, Şekil 4-30, Şekil 4-31 ve Şekil 4-32 örtük ve belirtik algoritmaların yedi model için analiz sürelerinin kullanılan işlemci sayısına göre değişimini sergiler. Belirtik algoritma, tüm modelleri çözmeyi başarmıştır, ancak örtük algoritma yedi nolu en büyük modeli bellek gereksinimleri nedeniyle çözememiştir.



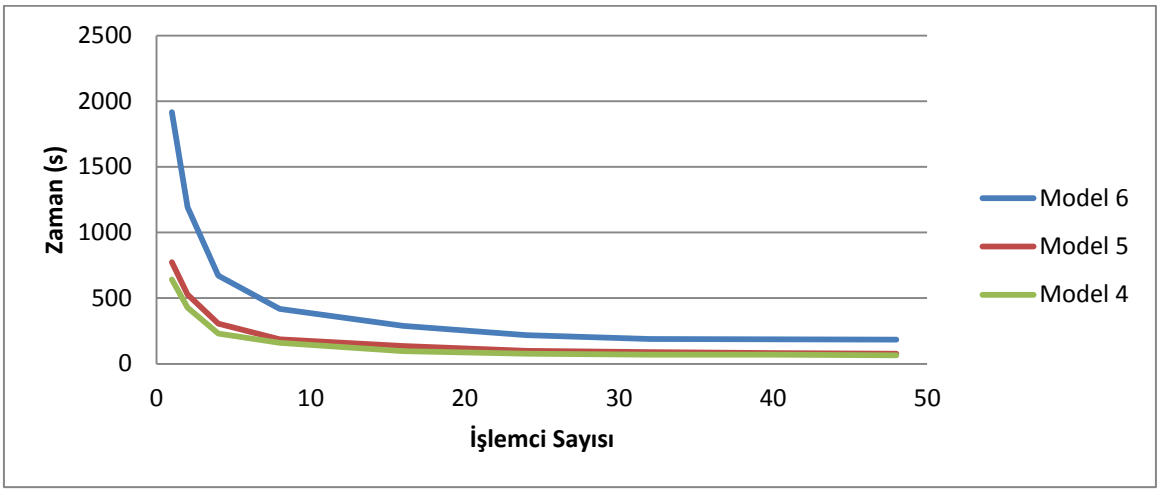
Şekil 4-29 Belirtik Algoritma Analiz Süreleri Modeller: 1, 2, 3



Şekil 4-30 Örtük Algoritma Analiz Süreleri, Modeller: 1, 2, 3

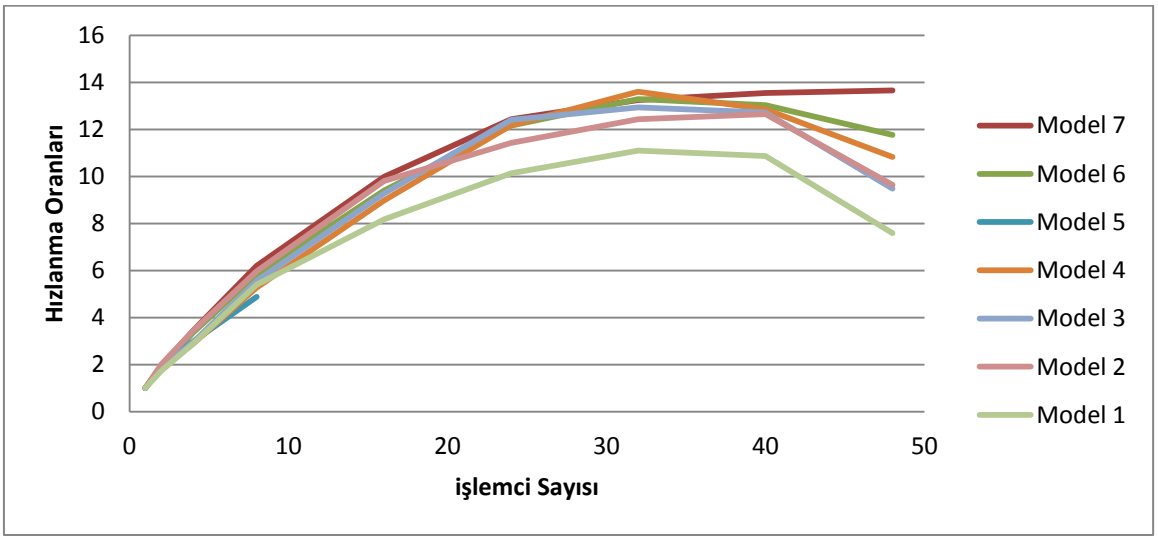


Şekil 4-31 Belirtik Algoritma Analiz Süreleri, Modeller: 4, 5, 6, 7

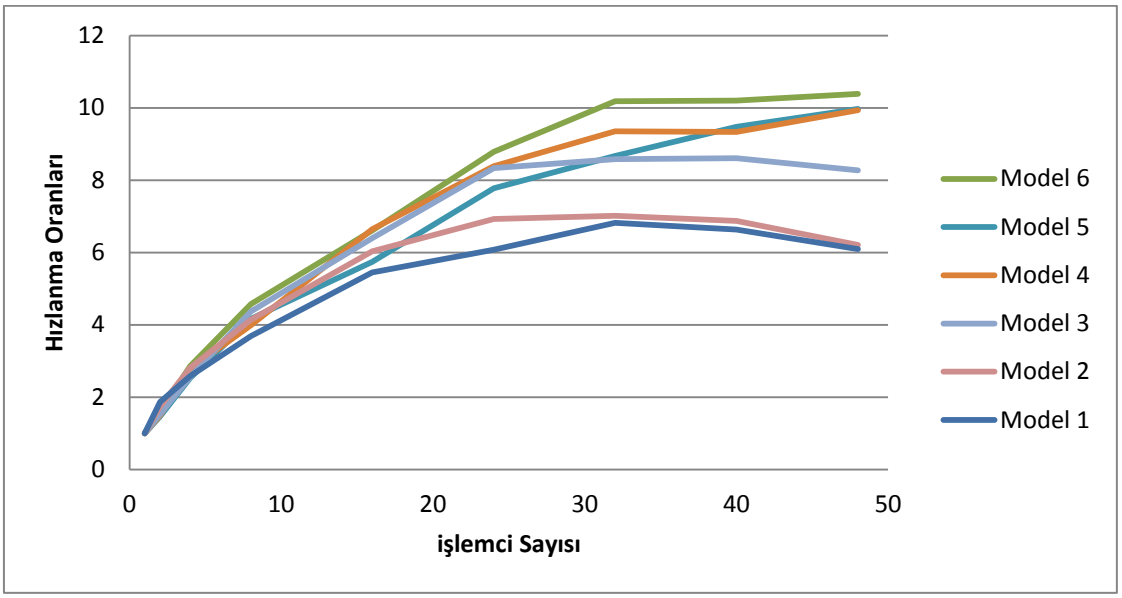


Şekil 4-32 Örtük Algoritma Analiz Süreleri, Modeller: 4, 5, 6

Şekil 4-33 ve Şekil 4-34 örtük ve belirtik algoritmaların yeni bilgisayar kümesi üzerinde ulaştıkları hızlanma oranlarının kullanılan işlemci sayısına göre değişimini gösterir. Modeller büyüdükçe iki algoritma da daha büyük hızlanma oranlarına ulaşmıştır. Belirtik algoritma en yüksek hızlanma oranına en büyük model olan model yedide 13.65'lik hızlanma oranıyla ulaşmıştır. Örtük algoritma en yüksek hızlanma oranına çözebildiği en büyük model olan model altıda 10.39'luk hızlanma oranıyla ulaşmıştır. İki algoritma da en büyük hızlanma oranlarına 48 işlemci kullanarak ulaşmış olsalar da küçük modellerde hızlanma oranı en yüksek değerine daha düşük işlemci sayılarında ulaşmıştır. Belirtik algoritmada 24, örtük algoritmada ise 32 işlemci kullanımı küçük modellerde en yüksek hızlanma oranlarına ulaşımı sağlamıştır.



Şekil 4-33 Belirtik Algoritma Hızlanma Oranları

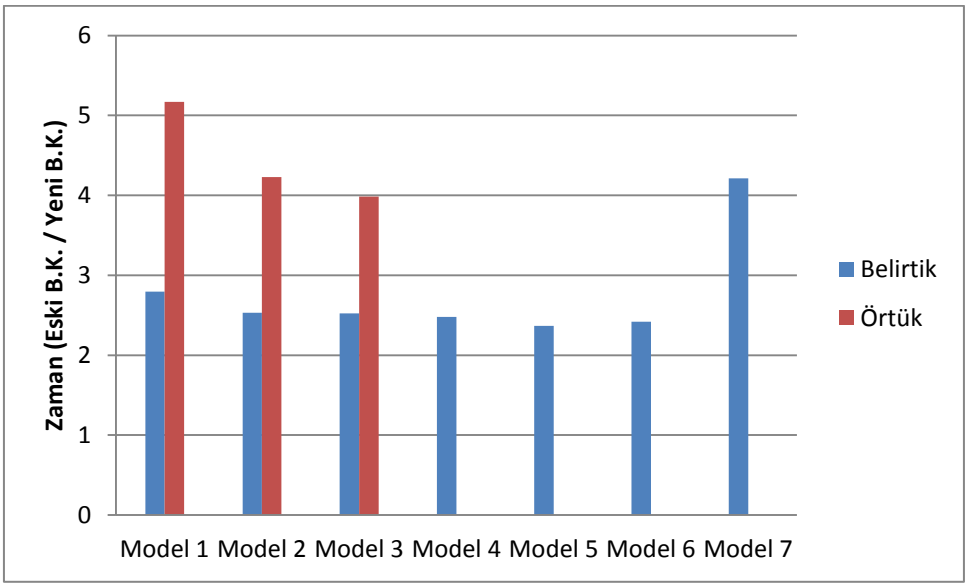


Şekil 4-34 Örtük Algoritma Hızlanma Oranları

Tablo 4-3 tüm modeller için HPCE2 ve yeni bilgisayar kümelerinde ulaşılmış en hızlı çözüm sürelerini gösterir. Örtük algoritma kullanılarak HPCE2 bilgisayar kümesinde ancak en küçük üç model çözülebilirken, yeni bilgisayar kümesinde en büyük model haricindeki altı model de çözülebilmiştir. Belirtik algoritma ilk altı modelde yeni bilgisayar kümesine göre ortalama 2.5 kat daha hızlı çalışmıştır (Şekil 4-35). Yedinci model de bu oran 4.2 kata çıkmıştır. Bu orandaki yükseliş bu modelin HPCE2 bilgisayar kümesi ile çözümlenmesi sırasında ortaya çıkan bellek yetersizliği problemlerinden kaynaklanmaktadır.

Tablo 4-3 Eski ve Yeni Bilgisayar Kümelerinde Belirtik ve Örtük Algoritmalarla Değişik Modeller İçin Ulaşılmış En Hızlı Çözüm Süreleri (saniye)

Model	Eleman Sayısı	Yeni B.K Belirtik	Yeni B.K Örtük	HPCE2 Belirtik	HPCE2 Örtük
Model 1	15625	1.92	5.36	5.36	27.70
Model 2	31250	3.60	12.02	9.10	50.80
Model 3	64000	6.98	26.60	17.59	106.00
Model 4	125000	13.31	64.70	33.00	x
Model 5	160000	17.29	77.72	40.90	x
Model 6	320000	34.75	184.46	84.00	x
Model 7	1000000	102.03	X	430.00	x



Şekil 4-35 Belirtik ve Örtük algoritma Kullanılarak HPCE2 ve Yeni Bilgisayar Kümelerinde Ulaşılan En Hızlı Çözüm Süreleri Oranları

Doğrusal Olmayan Dinamik Çözümleme

Bir önceki modelde doğrusal dinamik olarak çözülen zemin modeli bu bölümde örtük eşdeğer doğrusal ve belirtik eşdeğer doğrusal olarak yeni bilgisayar kümesinde çözülmüştür. Zemin modelinde yük olarak Şekil 4-28'de gösterilen anlık yük yerine eşdeğer doğrusal malzeme modelinin doğrulanmasında kullanılan Loma-Prieta deprem kaydı kullanılmıştır. Belirtik algoritmanın zaman aralığı belirlenmesinde doğrulama analizlerinde hesaplanan 0.0005 saniye kullanılmıştır. Örtük algoritmada ise deprem kaydının zaman aralığı olan 0.02 saniye analiz zaman aralığı olarak alınmıştır. Eşdeğer doğrusal analiz her zaman aralığında modelin sistem sönümleme ve direngenlik matrislerinin hesaplanması gerektirdiğinden doğrusal analize göre daha uzun zaman alır. Çözümlerin aldığı uzun zamanlar nedeniyle doğrusal dinamik olarak test edilen modellerden en küçük dördü (

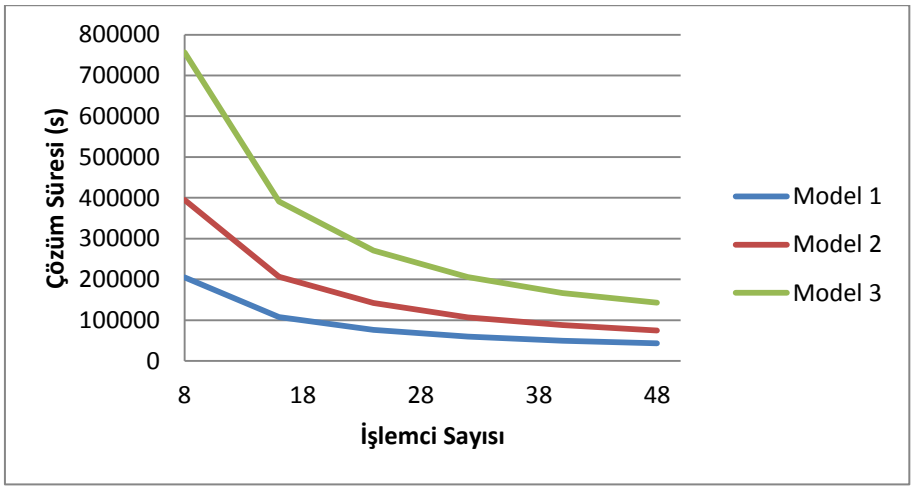
Tablo 4-4) değerlendirmeye alınmış ve testlerde en az sekiz işlemci kullanılmıştır.

Tablo 4-4 Eşdeğer Doğrusal Analizlerinde Kullanılan Modeller

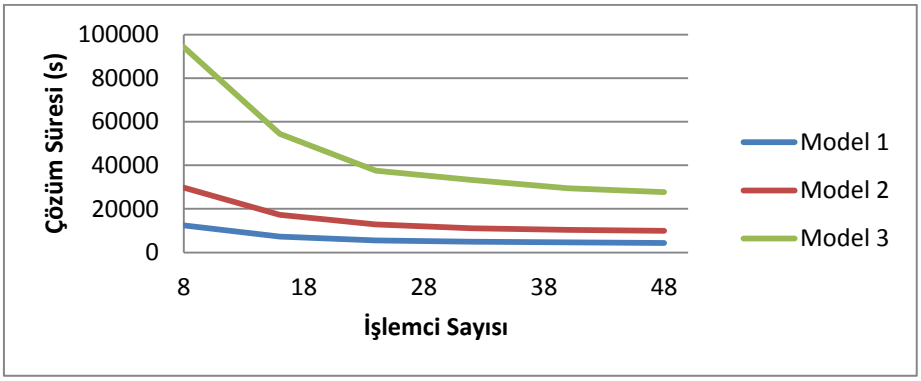
Model	Eleman Sayısı	Denklem Sayısı
Model 1 (25x25x25)	15625	52728
Model 2 (50x25x25)	31250	103428
Model 3 (40x40x40)	64000	206763
Model 4 (50x50x50)	125000	397953

Şekil 4-36 ve Şekil 4-37 değişik sayıda işlemciler kullanılarak yapılan belirtik ve örtük eşdeğer doğrusal dinamik analizlerine ait çözüm sürelerini gösterir. Doğrusal dinamik analizin aksine eşdeğer doğrusal dinamik analizde örtük algoritma daha başarılı performans sağlamıştır (Tablo 4-5). Bunun sebebi bu

algoritmanın zaman aralığından bağımsız kararlılık göstermesidir. Bu nedenle analizlerde örtük algoritmaya göre çok daha büyük ($\Delta t_{\text{örtük}} = 0.02s, \Delta t_{\text{belirtik}} = 0.0005s$) zaman aralıkları kullanılabilmiştir.



Şekil 4-36 Belirtik Eşdeğer Doğrusal Dinamik Algoritma Çözüm Süreleri



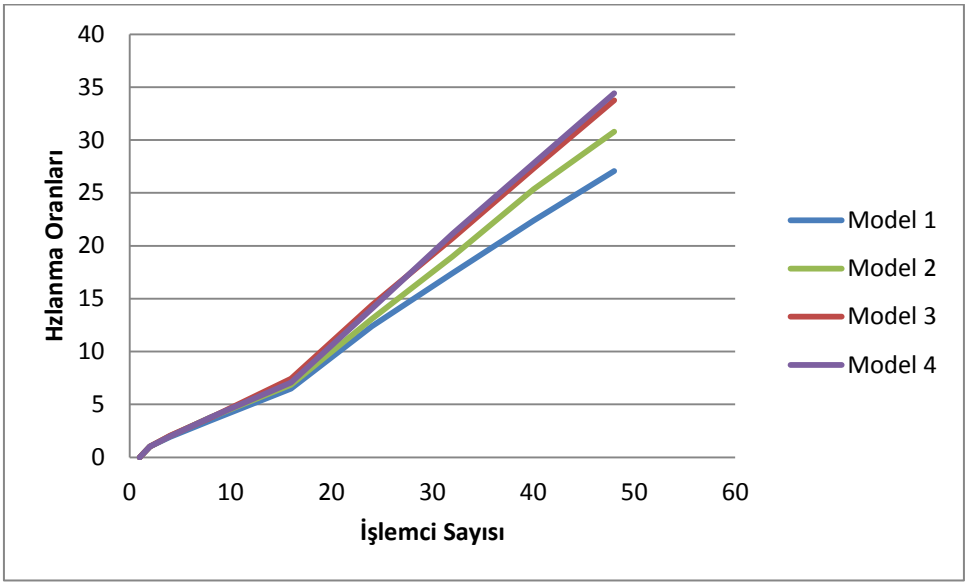
Şekil 4-37 Örtük Eşdeğer Doğrusal Dinamik Algoritma Çözüm Süreleri

Tablo 4-5 Belirtik Ve Örtük Eşdeğer Doğrusal Dinamik Algoritmalarla Değişik Modeller İçin Ulaşılmış En Hızlı Çözüm Süreleri (saniye)

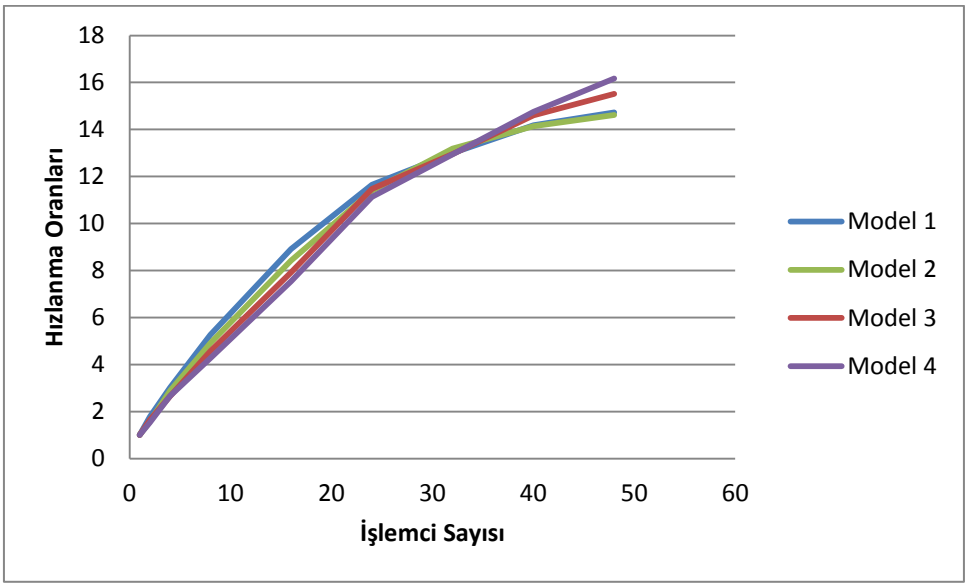
Model	Eleman Sayısı	Belirtik	Örtük
Model 1	15625	43170	4428
Model 2	31250	74794	9989
Model 3	64000	142715	27733
Model 4	125000	264540	79825

Error! Reference source not found.4-38 ve **Error! Reference source not found.**4-39 değişik sayıda işlemciler kullanılarak yapılan belirtik ve örtük eşdeğer doğrusal dinamik analizlerine ait hızlanma

oranlarını gösterir. Sekiz çekirdekten daha az çözümlerde problem sınırlı sayıda zaman adımı kullanılarak çözülmüş ve toplam çözüm süresi herbir zaman adımı için harcanan zaman kullanılarak tahmin edilmiş ve hızlanma oranları bu tahminlere göre hesaplanmıştır. Tahminlere göre en büyük modelin tek işlemci kullanılarak çözülmesi için yaklaşık 138 güne ihtiyaç vardır. Eşdeğer doğrusal analiz doğrusal analizle aynı iletişim gereksinimlerine sahiptir ancak daha fazla hesaplama içerir. Bu özellik hem belirtik hem de örtük algoritmanın doğrusal analizlere göre daha yüksek hızlanma oranlarına ulaşmalarına sebep olmuştur. Belirtik algoritma örtük algoritmaya göre daha az işlemciler arası iletişime gereksinim duyduğundan hızlanma oranlarının büyümesi bu algoritmanın kullanıldığı analizlerde daha belirgin olmuştur. Belirtik algoritma model dörtte 48 işlemci kullanarak 40.3'lük hızlanma oranına ulaşırken örtük algoritma bu değer 16.2 olmuştur. Belirtik ve örtük algoritma modeller büyüdükçe ve işlemci sayısı arttıkça daha büyük hızlanma oranlarına ulaşmışlardır.



Şekil 4-38 Belirtik Eşdeğer Doğrusal Dinamik Algoritma Hızlanma Oranları

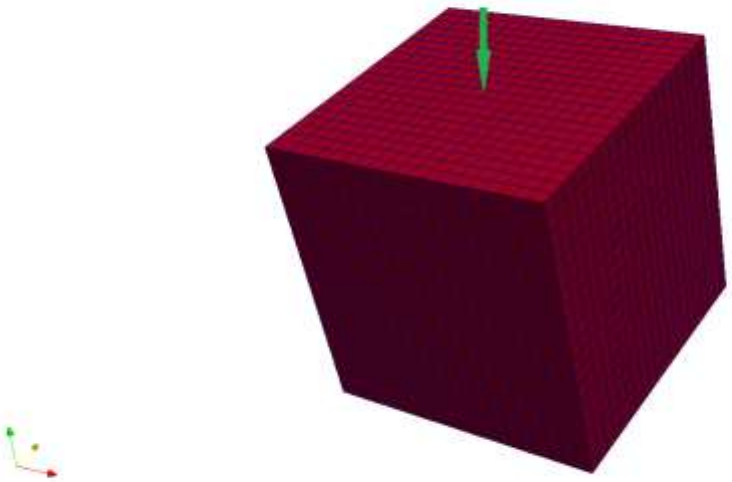


Şekil 4-39 Örtük Eşdeğer Doğrusal Dinamik Algoritma Hızlanma Oranları

Paralel Belirtik Algoritma İle Çok Büyük Boyutlu Modellerin Çözülmesi

Belirtik algoritma ile çözülebilecek en büyük modelin bulunması amacıyla beş farklı test modeli dinamik doğrusal olmayan belirtik algoritmayla analiz edilmişlerdir.

Testler içinŞekil 4-40'ta gösterilen küp şeklindeki toprak modeli, anlık bir kuvvetle yüklenmiş ve modelin 1 saniye boyunca yaptığı deplasmanlar hesaplanmıştır. Çözümleme süresince 0.05 saniyelik zaman aralıkları kullanılıp toplam 20 zaman adımı için çözümleme gerçekleştirilmiştir. Tablo 4-6'da testlerde kullanılan modeller, bu modellerin eleman ve modellerin içerdiği denklem sayıları verilmiştir. Testlere bir 1,000,000 elemana sahip modelle (Model 1) başlanmış ve model büyüklüğü arttırılarak devam edilmiştir (Tablo 4-6). Testlerde kullanılan bilgisayar sistemine daha önce sadece GİB testlerinde kullanılan GPUBlade bilgisayarı da dahil edilip testler toplam yedi bilgisayar üzerinde 56 işlemci kullanılarak yapılmıştır. Testlere dört işlemci kullanılarak başlanmış sırasıyla 7, 8, 16, 32, 56 işlemci kullanılarak tekrar edilmiştir.

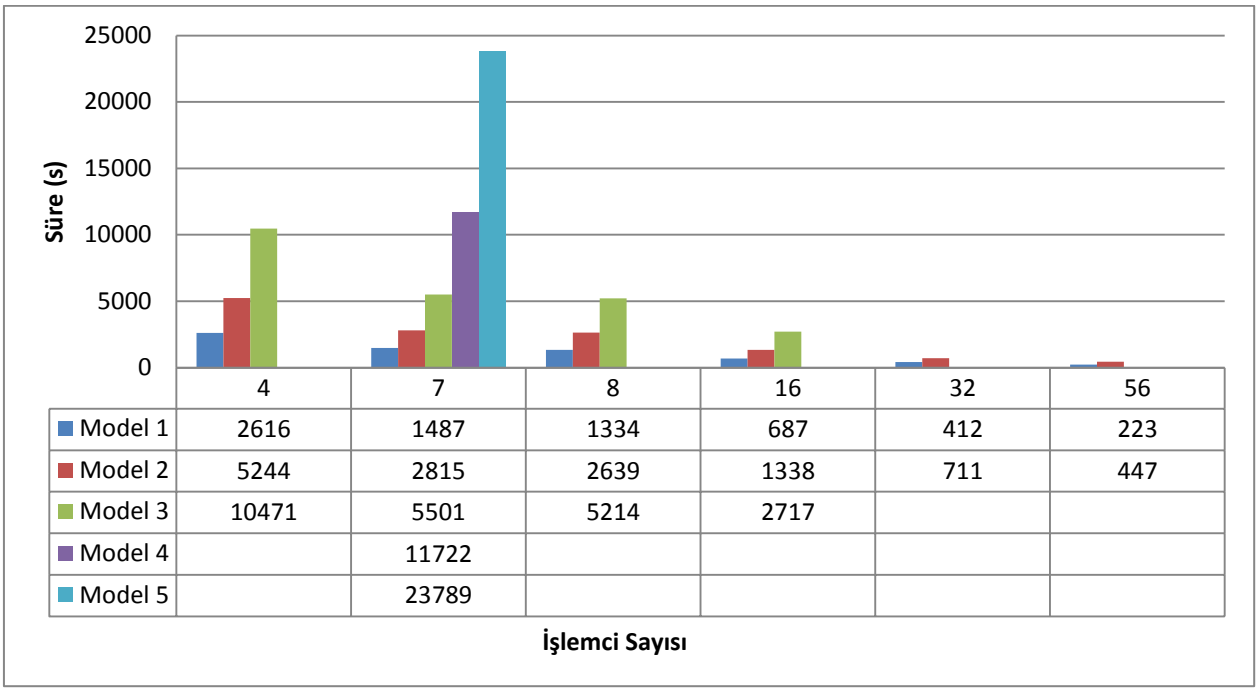


Şekil 4-40 Belirtik Algoritma Testlerinde Kullanılan Model

Tablo 4-6 Belirtik Algoritma Testlerinde Kullanılan Modeller

Model	Eleman Sayısı	Denklem Sayısı (Yaklaşık)
Model 1 (100x100x100)	1,000,000	3,000,000
Model 2 (100x100x200)	2,000,000	6,000,000
Model 3 (100x200x200)	4,000,000	12,000,000
Model 4 (200x200x200)	8,000,000	24,000,000
Model 5 (200x200x400)	16,000,000	48,000,000

Belirtik algoritma kullanılarak yapılan testlerin sonuçları Şekil 4-41'de gösterilmiştir. Çözüm süreleri kullanılan işlemci sayısıyla azalmış, modellerin içerdiği elemanların sayısıyla artmıştır. Model 1 ve Model 2 tüm testlerde başarıyla çözülmüşlerdir. Bellek sorunları Model 3'ün 32 ve 56 işlemcili testlerde, Model 4 ve Model 5'in ise 7 işlemcili testler haricindeki tüm testlerde çözülememesine yol açmıştır. Testler sonucunda 16,000,000 elemanlık Model 5 sadece bilgisayar başına 1 işlemci kullanılarak yapılan 7 işlemcilik testte çözülebilse de şu an için yeni bilgisayar kümesinde çözülebilen en büyük model olmuştur. Bu modelin çözümlenmesi sırasında bilgisayarların bellek kapasitesinin 95% kullanıldığı gözlenmiş, bu yüzden daha büyük modellerle testlere devam edilme gerekliliği duyulmamıştır. Analizler sırasında ortaya çıkan bellek sorunları giriş dosyası okuma işleminden dolayı değil, sonlu elemanlar modelini oluşturan nesnelerin kapladığı ve algoritmanın işleyişi sırasında kullanılan bellekle alakalıdır.



	4	7	8	16	32	56
Model 1	2616	1487	1334	687	412	223
Model 2	5244	2815	2639	1338	711	447
Model 3	10471	5501	5214	2717		
Model 4		11722				
Model 5		23789				

Şekil 3-41 Belirtik Algoritma Test Sonuçları

5. Bölüm

GA-GİB'ler için Çözümleme Algoritmalarının Geliştirilmesi

Giriş

Genel amaçlı grafik işlem birimleri, GA-GİB, çok sayıda düşük kapasiteli işlemci içermesi sebebiyle son yıllarda bilimsel hesaplama için kullanılabilecek bir seçenek haline gelmiştir. Proje kapsamında ise GA-GİB'ler için çeşitli çözücüler geliştirilmiş ve bu çözücülerin yapısal mekanik problemlerinde kullanılmasıyla ilgili çalışmalar yapılmıştır. Sonlu eleman yöntemi kullanıldığında elde edilen matrislerin seyrek matris olması nedeniyle ilk olarak seyrek matris çözücüler üzerine yoğunlaşılmış ve çoklu-sınır ve aşamalı çoklu sınır yöntemi kullanan iki tip seyrek matris çözücüzü geliştirilmiştir. Bu çözücülerin başarımları farklı GİB'lerde sınanmıştır. Çoklu sınır ve aşamalı çoklu sınır yöntemlerinin çözüm mantığı, seyrek matrislerden küçük boyutlu yoğun matris oluşturmak üzerine kuruludur. Dolayısıyla yoğun matris işlemlerindeki hızlanmalar direk olarak seyrek matris çözümünün başarımını etkilemektedir. Bu sebepten dolayı yüksek başarımla çalışan yoğun matris çözücüsünün geliştirilmesi üzerine de çalışılmıştır. Yapı mekaniği problemlerinde hızlanmaya en ihtiyaç duyulan alanlardan bir tanesi de zaman alanında doğrusal olmayan dinamik çözümlemedir. Proje kapsamında GA-GİB'ler için belirtik algoritma kullanılarak bir doğrusal olmayan dinamik algoritma geliştirilmiş ve algoritmanın başarımı hem farklı GİB'lerde sınanmış hem de elde edilen sonuçlar paralel MİB çözümleriyle karşılaştırılmıştır.

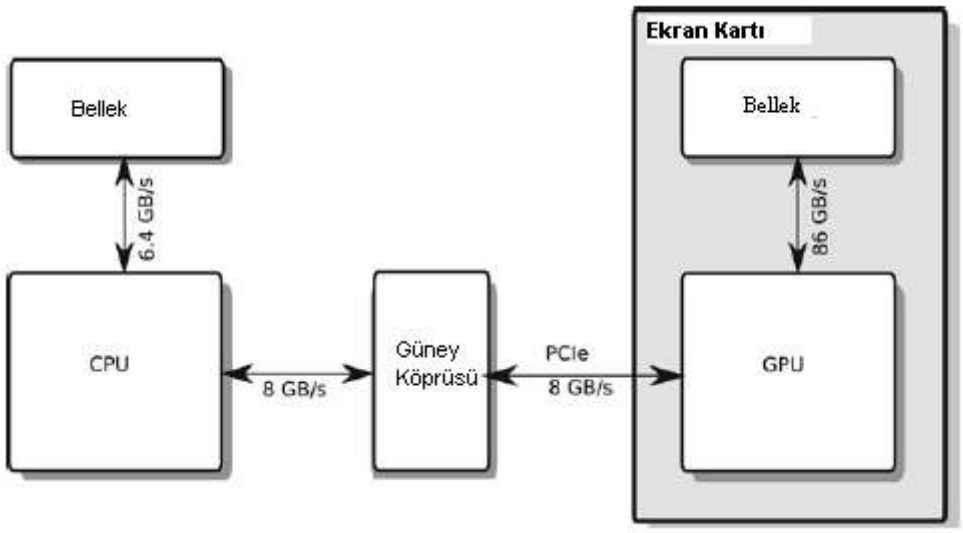
Yeni Nesil Ekran Kartlarının Donanım Yapısı

Grafik işlemcili platformlar çok değişik alanlarda gerekli olan işlem gücünü rahatlıkla sağlayabilmektedirler. Bu genel amaçlı hesaplamaları, çok yüksek kapasitede sayısal hesaplama gücüne ve çok geniş bellek gücüne borçludurlar. Ayrıca çok ucuz oldukları için kolaylıkla kurulabilecek bir sistemdir (Schenk et al. 2008).

Grafik işlemciler başlangıçta sadece 2 ve 3 boyutlu grafik işlemlerini hızlandırmak için tasarlanmış ve uzun bir süre, grafik işlemciler sadece Direct-X veya OpenGL gibi grafik kütüphaneleri kullanılarak programlanmaktaydılar. Ancak günümüzde AMD'nin "stream computing" ve Nvidia'nın "Cuda" teknolojileri sayesinde ekran kartlarını, üzerlerindeki yüksek başarımlı grafik işlemcileri kullanarak çok çeşitli alanlarda hesaplama yapmak için kullanmak mümkün hale gelmiştir. Oyun piyasasının çok hızlı gelişmesi ve ekran kartlarında duyulan hızlı hesaplama ihtiyacından dolayı grafik işlemciler, bilgisayar ana işlemcilerine (MİB) göre çok daha hızlı bir şekilde gelişme göstermişlerdir. Bu sebepten dolayı da ekran kartları yakın maliyete sahip işlemcilere göre daha fazla işlem yapabilme gücünü barındırmaktadırlar. Örneğin GeForce 8800 GİB (Grafic Process Unit, grafik işlem birimi) 315 GFlop/s işlem gücüne sahipken çift çekirdekli Intel-Xeon 5160 işlemcisi 24 GFlop/s hızına ulaşabilmektedir (Schenk et al. 2008). Diğer taraftan grafik işlemcilerinin (GİB) programlanması ana işlemcilere (MİB) göre daha zor ve bu işlemcilerin

sunduğu programlama olanakları ve esneklikleri ana işlemcilere göre çok daha azdır. Bu sebepten dolayı da grafik işlemcilerinin sadece hesaplama yükü ağır, özel işlemler için kullanılmaları önerilmektedir (Schenk et al. 2008).

Şekil 5.1, ekran kartı, örnek bir bilgisayarda ana işlemci ve ana kartın birbirlerine nasıl bağlandığını göstermektedir (Schenk et al. 2008). Ana işlemciyle grafik işlemcisi "South Bridge" diye adlandırılan bir ara bağlantıyla birbirlerine bağlanmaktadır. PCI express ara yüzünü kullanan bu bağlantının veri aktarım hızı 8 GB/s civarındadır. Bir diğer önemli nokta ise, ana işlemci bellek veri aktarım hızı 6.4 GB/s hızında kalırken grafik işlemcisi ekran kartı üzerindeki bellekle 86 GB/s hızıyla veri alışverişi yapabilmektedir. Şekilde gösterilen hız değerleri farklı donanımlarda değişim gösterse de genel olarak işlemci-ekran kartı bağlantısı Şekil 5.1'deki gibidir. Ekran kartı ve grafik işlemci üreticisi NVidia'nın ürettiği grafik işlemcileri son yıllarda sırasıyla 8, 9 ve 10 serileri olmak piyasa sürülmüştür. Takip eden bölümde 8 serisi grafik işlemcilerinin donanım yapıları detayları verilmiştir.



Şekil 5.1: İşlemci ve Ekran kartı Bağlantı Şeması (Schenk et al. 2008)

8 Serisi Grafik İşlemcileri

Grafik işlemcileri genel olarak "Stream İşlemcilerinden" (SP – Stream Processor)'den oluşmaktadır. Örneğin 8800GTX ekran kartları G80 grafik işlemcisine sahiptir. G80 grafik işlemcisinde 128 adet stream işlemcisi bulunmaktadır. Her bir stream işlemcisi üzerinde bir iş parçası (thread) çalıştırılarak herhangi bir matematiksel hesap paralel olarak yapabilmektedir. Her bir stream işlemci üzerinde kendi lokal verilerini ve yapacağı işlem bilgilerini tutabildiği küçük boyutlu bellekler (register) bulunmaktadır. 8 tane Stream işlemcisi araya gelerek Streaming çoklu işlemci (Streaming Multiprocessor) ünitesini oluşturmaktadır (Şekil 5.2). Bu çoklu işlemciler kendilerine ait özel paylaşımlı belleğe ve yapılacak işlemleri saklayan ek belleklere sahiptir. I-Cache adı verilen bellek stream işlemcilerde gerçekleştirilecek işlemlerle ilgili bilgiyi saklar. C-Cache sadece okunabilir bellektir ve stream işlemcilerinde gerçekleştirilecek matematiksel işlemlerle ilgili veriyi saklar. Stream işlemciler buradaki bilgiyi ihtiyaç duyduklarında kullanabilirler ama değiştiremezler (Shimpi and Wilson 2008). Streaming Çoklu İşlemcisinde ayrıca iki adet Özel Fonksiyon Birimi (SFU – Special Function Unit) bulunmaktadır. Bu birimler grafik hesaplamaları sırasında sıkça

kullanılan fonksiyonların yüksek verimlilikle gerçekleştirilmesini sağlayan özel donanımlardır. İş dağıtıcının (MT Issue) görevi ise stream işlemcilerine ve özel fonksiyon birimlerine gelen iş yüklerini dengeli bir şekilde aralarında dağıtmaktır.

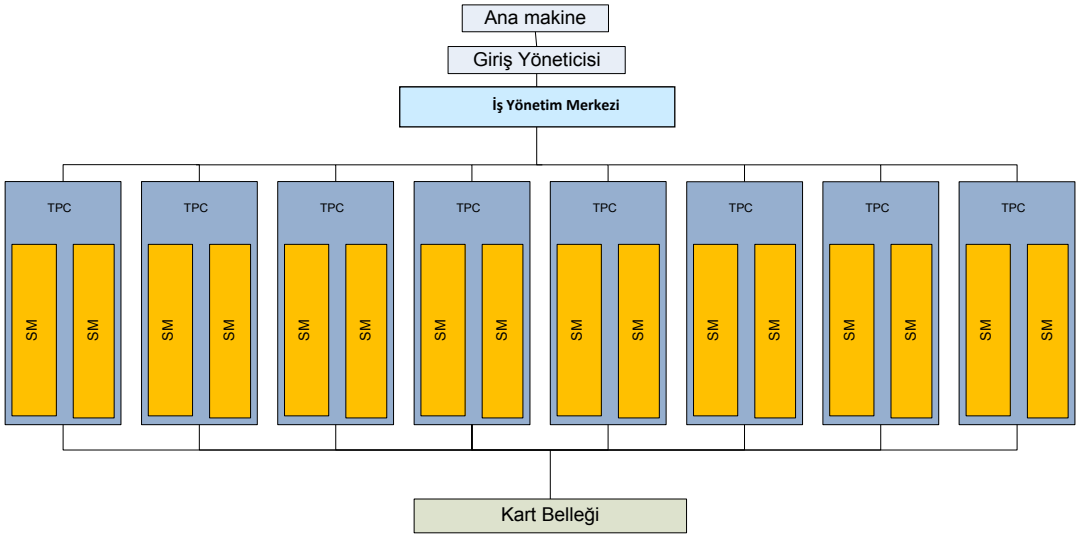
Stream işlemcilerde gerçekleştirilecek işlemler için kullanılacak veriler paylaşımlı bellekte tutulur. Bu paylaşımlı belleğin büyüklüğü çok fazla olmadığı (8 serisi işlemcilerde 16 kByte), her bir stream işlemcinin kendi lokal verileri için cok sınırlı bir belleğe sahip olduğu da düşünülürse, stream işlemcilerde ancak düşük düzey basit matematiksel işlem gerçekleştirmek mümkündür. Streaming çoklu işlemciler ekran kartlarında tek komutla çoklu data işleme, SIMD, (single instruction multiple data) görevini yerine getirmektedirler. Bu işlemciler 32 bitlik işlem hassasiyetine sahip olup sadece 8 basamak hassasiyetli sayılarla (single precision) işlem yapabilmektedir.

Şekil 5.2: Streaming Çoklu İşlemci Yapısı (Oster 2008)

Streaming çoklu işlemcilerin bir üst seviyesinde Texture İşlemci Kümesi (Texture Processor Cluster, TPC) yer alır. G80 işlemcilerinde iki tane Streaming çoklu işlemci bir araya gelerek Texture İşlemci Kümesini oluşturmaktadır (Şekil 5.3). Texture İşlemci Kümesi yapısı özel olarak grafiklerde geometri ve üç boyut hesaplamaları için geliştirilmiş olup bu kümeler direk olarak işlem yönetim merkezine bağlıdır (Thread Control Unit) (Oster 2008). İşlem yönetim merkezi, giriş yöneticisinden gelen işlemleri, texture işlemci kümesi dolayısıyla streaming çoklu işlemcilerde iş parçalarının yaratılmasını sağlayarak gerçekleştirir. Giriş yöneticisi ana makineye, yani ana işlemciye bağlı olup, işleteceği komutları ana işlemciden alır. Gerçekleştirilecek işlemin grafik hesaplaması ya da sayısal hesaplama olup olmadığına göre işlem yönetim merkezini idare etmek giriş yöneticisinin temel görevidir.

G80 grafik işlemcisinde toplam 8 adet texture işlemci kümesi, 16 adet shared çoklu işlemci, 128 tane stream işlemcisi bulunmaktadır. Stream işlemcilerinin her biri 1.35 GHz hızında çalışmaktadır. G80 grafik işlemcisi ekran kartı üzerindeki kart belleğine 6 yollu 64 bitlik veriyoluyla bağlanmakta olup yaklaşık 86.4 GByte/s'lık bir hızla veri aktarabilmektedir. Bellekler ise 1.8 GHz hızında çalışmakta olup 768 MByte büyüklüğe sahiptir. Her Streaming çoklu işlemcisinde bir adet SIMT (single instruction multiple thread –

tek komut birden fazla iş) yöneticisi bulunmaktadır. SIMT yöneticisi çözgü (warp) adı verilen paralel çalışabilen iş parçalarını (thread) başlatabilir, yönetebilir, sıralayabilir ve yürütebilir. Her bir çözgü (warp) 32 adet paralel iş parçasından oluşmaktadır. SIMT yöneticisi tek bir streaming çoklu işlemcide 24 adet çözgü yaratabilmektedir. Diğer bir deyişle, 8 serisi işlemcilerde, toplamda tek bir SIMT tek komutla 24x32=768 adet iş parçacığını yaratıp yönetebilmektedir (Nvidia 2008).



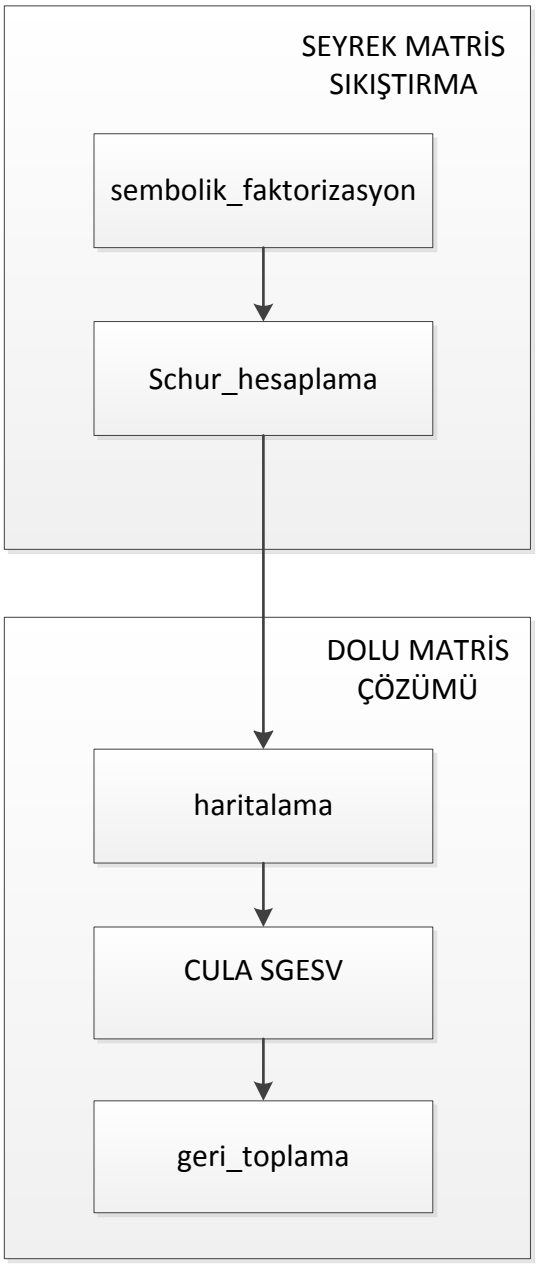
Şekil 5.3: G80 işlemcisinin yapısı

Çoklu-Sınır (Multiple Front) Çözüm Algoritması

GA-GİB (Genel amaçlı grafik işlem birimi) sistemlerin, "Tek Komut Çoklu Veri" (SIMD) olarak çalışması, aritmetik işlemlerin ekran kartı içerisindeki veri aktarımından daha kısa sürede yapılması, bunların dışında hâlihazırda GA-GİB sistemleri kullanarak dolu matrisleri çözen kütüphanelerin bulunması sebebiyle ilk olarak seyrek matrisler için "Çoklu Sınır (Multiple-Front)" çözüm algoritmasının geliştirilmesine karar verilmiştir. Bu kararda önemli rol oynayan başka bir etmen ise çoklu sınır algoritmasının bazı değişikliklerle daha kolay bir biçimde diğer algoritmalara dönüştürülebiliyor olmasıdır.

Algoritmanın temel mantığı altyapı sistemleri arasındaki ortak serbestlik derecelerine ait denklemlerin toplandığı sınır denklem sistemini oluşturduktan sonra bu sistemi çözmek ve bulunan sonuçları kullanarak her altyapı sisteminde geriye kalan deplasman değerlerini çözmektir. Seyrek matris çözücü algoritmasının akış diyagramı Şekil 5.4'te verilmiştir. Şekilde görüldüğü gibi algoritma temel olarak iki bölümden oluşmaktadır. İlk bölüm olan Seyrek Matris Sıkıştırma (İndirgeme) işleminin amacı, daha önceki aşamalarda oluşturulmuş alt-yapı matrislerinin, seyrek matrisler için geliştirilmiş algoritmalar ile sınır matrisine etkisi olacak Schur Birleşenlerinin hesaplanması ve seyrek matrislerin sıkıştırılma işlemlerinin yapılmasıdır. Bu bölümde kullanılan *sembolik_faktorizasyon* fonksiyonu ile seyrek matrislerde yapılacak olan faktorizasyon ve sıkıştırma işlemleri sırasında gerekli olan üst üçgensel matristeki elemanların yerleri ve bu elemanların kendi aralarındaki ilişkisi belirlenir. *Sembolik_faktorizasyon* fonksiyonunun çıktılarının kullanıldığı *Schur_hesaplama* fonksiyonuyla da direngenlik matrislerinin üst üçgensel matrisi ve Schur Birleşenleri, kuvvet vektörlerinin de sıkıştırılmış halleri elde edilerek, seyrek matris sıkıştırılması tamamlanmış olur. İkinci bölümde ise "Seyrek Matris

Sıkıştırma" bölümünde elde edilen bütün Schur Bileşenleri ve sıkıştırılmış kuvvet vektörleri haritalama fonksiyonu ile birleştirilerek bir adet sınır direngenlik matrisi ve bir adet sınır kuvvet vektörü elde edilir. Arayüz direngenlik matrisinin dolu matris olması sebebiyle bu aşamada GİB ile çalışan bir dolu matris çözücüye ihtiyaç duyulmaktadır. Bu ihtiyaç ticari bir ürün olan CULA kütüphanesinin (GİB ile hızlandırılmış lineer cebir kütüphanesi) deneme sürümü kullanılarak giderilmiştir. Arayüz denklemlerinin çözümü ile elde edilen arayüz düğümlerine ait deplasman değerleri geri_toplama fonksiyonu ile ilk bölümde üst üçgensel matris haline getirilmiş altyapı matrisleriyle birlikte kullanılarak sistemin bütün deplasman değerleri çözülmüş olur.



Şekil 5.4: Seyrek Matris Çözücü Algoritması Blok Diyagramı

Seyrek Matris Sıkıştırma İşlemi

Bu bölümde alt-yapı matrislerinin faktorizasyonu ve sınır direngenlik matrisine etki edecek Schur bileşenlerinin hesaplanması tamamlanır. Ancak Schur bileşenlerinin hesaplanabilmesi için alt-yapı sistemlerindeki ortak serbestlik derecelerine ait denklemlerin denklem sisteminde sonda olması gerekmektedir. Bu düzenleme, sıkıştırma işleminden önceki basamaklarda tamamlanır ve altyapı direngenlik matrisleri ile kuvvet vektörleri sıkıştırma işlemine bu düzenlemeden sonra girer. Sıkıştırma işlemi sembolik_faktorizasyon ve Schur_hesaplama olmak üzere iki fonksiyondan oluşmaktadır.

sembolik_faktorizasyon: Verilen bir seyrek matrisin faktorizasyonu sırasında terimler arasında eleme işleminin hangi sırayla yapılacağı, faktorizasyon sonucunda oluşan üst üçgensel matrisin kolonlarında kaç adet sıfır olmayan terimin bulunduğu ve bu terimlerin yerleri bu fonksiyonla belirlenir. Fonksiyonun çıktıları Schur_hesaplama fonksiyonunda kullanılır.

Schur_hesaplama: sembolik_faktorizasyon fonksiyonun çıktıları kullanılarak ayrıştırma işlemi başlatılır. Alt yapı matrisinde sonda bulunan ortak serbestlik dereceleri hariç, diyagonalde bulunan bütün terimlere göre ayrıştırma işlemi yapılır. Bu işlem tamamlandığında serbestlik derecelerinin oluşturduğu Schur bileşeni elde edilmiş olur. Ayrıştırma işlemi sırasında kuvvet vektörü de değiştirilerek arayüz vektörüne etki edecek sıkıştırılmış kuvvet vektörü elde edilir. Ayrıca diğer serbestlik derecelerinin bulunduğu bölüm de üst üçgensel matris haline dönüştürülmüş olur.

Şekil 5.5'da Seyrek Matris Sıkıştırma İşlemi'nin iki ayrı alt yapıya ait olan direngenlik matrisleri için uygulanışı gösterilmiştir. Şeklin, en üst bölümünde yer alan sıkıştırma işleminin yapılacağı altyapı direngenlik matrislerinde sıfır olmayan terimler içi dolu kare (■), matrislerin sağ alt bölümüne toplanmış halde bulunan çarpılar (X) ise ortak serbestlik derecelerine ait terimleri göstermektedir. Seklin altında bulunan matrisler ise sembolik_faktorizasyon ve Schur_hesaplama fonksiyonları uygulandıktan sonra matrislerin elde edilen son halleridir. sembolik_faktorizasyon fonksiyonu ile daha önce belirtilen terimlerin dışında ayrıştırma sırasında sıfırdan farklı hale gelen terimlerin (○) yeri belirlenirken, kesik çizgi (--) ile çevrelenmiş alanlar olarak gösterilen Schur bileşenleri de, ortak serbestlik derecelerine ait terimlerin bulunduğu satırlara kadar uygulanan ayrıştırma işleminin yapıldığı Schur_hesaplama fonksiyonu ile belirlenir.

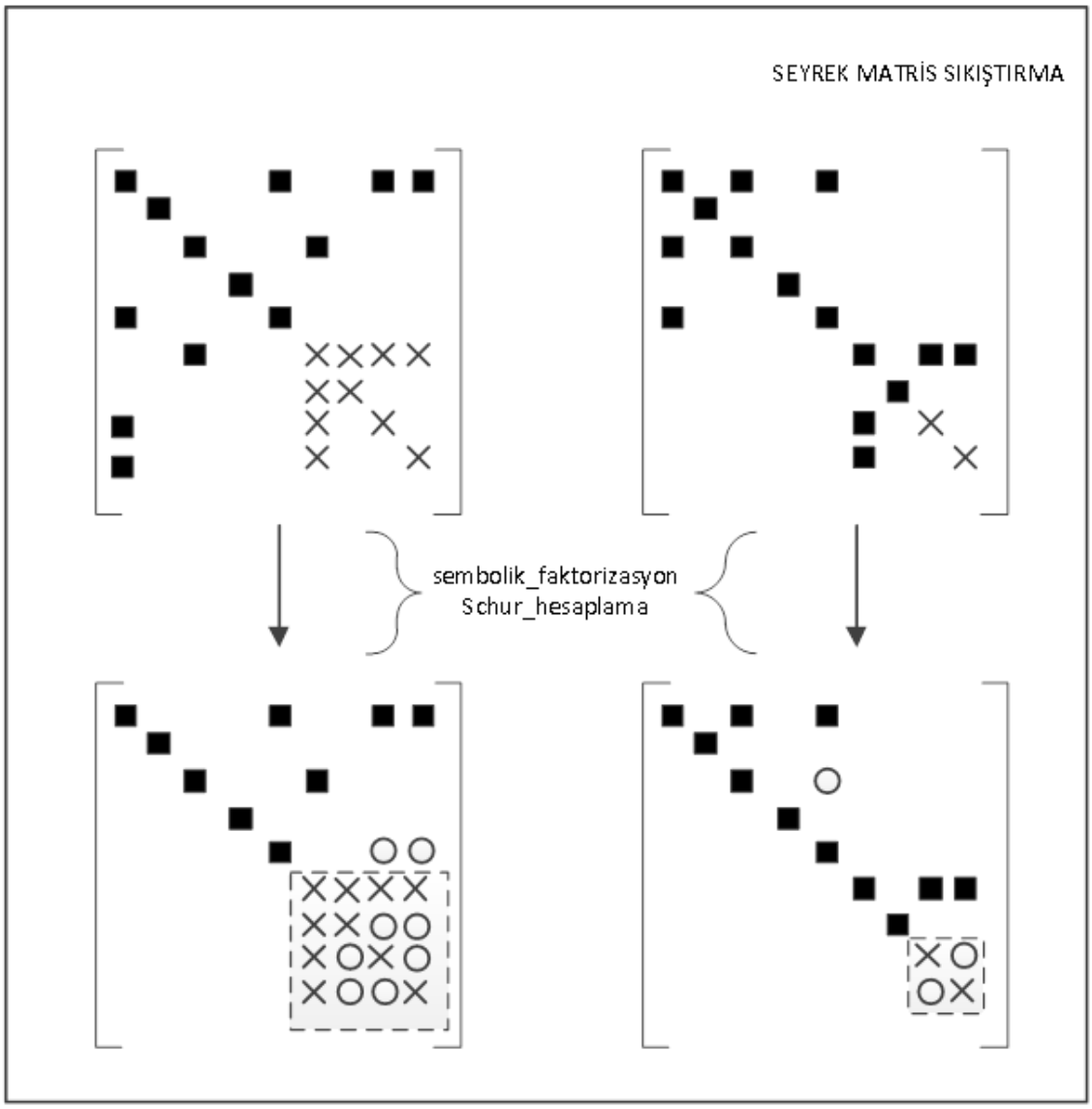
Dolu Matris Çözümü

Bu aşama haritalama, CULA_SGESV ve geri_toplama fonksiyonlarından oluşur.

Haritalama: Sıkıştırma işlemi sonucunda elde edilmiş Schur bileşenleri ve sıkıştırılmış kuvvet vektörleri haritalama fonksiyonuyla arayüz direngenlik matrisine ve arayüz kuvvet vektörüne dönüştürülür. Bu işlem temel olarak birçok elemandan oluşan bir sistem için eleman direngenlik matrislerinin kullanılmasıyla bütün bir sistemin direngenlik matrisinin oluşturulmasıyla aynıdır. Her altyapı sisteminden elde edilen Schur bileşenleri ve sıkıştırılmış kuvvet matrisleri birleştirilerek, sadece ortak serbestlik derecelerine ait denklem sistemi oluşturulur.

CULA_SGESV: CULA paketine ait Gauss Ayrıştırma yöntemini kullanarak dolu sistemlerin çözüldüğü fonksiyondur. Sisteme girdi olarak verilen arayüz direngenlik matrisi ve arayüz kuvvet matrisi çözülerek arayüz düğümlerindeki deplasman değerleri elde edilir.

geri_toplama: CULA_SGESV fonksiyonu ile elde edilen arayüz deplasman değerlerinin, haritalama fonksiyonuna benzer bir algoritmayla alt yapı sistemlerindeki karşılık geldiği serbestlik dereceleri belirlenir, daha sonra her altyapı sistemindeki geriye kalan serbestlik derecelerinin deplasman değerleri hesaplanır.



Şekil 5.5: İki seyrek matris için "Seyrek Matris Sıkıştırma" işlemi

Şekil 5.6'da sıkıştırma işlemleri tamamlanmış iki seyrek matris için "Sınır Matris Çözümü" gösterilmektedir. Şekilde en üst bölümde altyapı sistemlerine ait iki seyrek matrisin arayüz matrisine katkısı olacak Schur bileşenleri kesik çizgilerle çevrilmiş alanlar olarak gösterilmiştir. Daha sonra matrislere ait Schur bileşenleri haritalama fonksiyonuyla birleştirilerek arayüz sistemi oluşturulmaktadır. Şekilde arayüz direngenlik matrisi K_c, arayüz kuvvet vektörü F_c ve arayüz düğümlerinin deplasman vektörü de d_c olarak gösterilmektedir. Oluşturulan bu arayüz sisteminin CULA_SGESV fonksiyonuyla çözülmesiyle arayüz deplasman değerleri elde edilmektedir. Arayüz sistemini, alt yapı sistemlerindeki ortak serbestlik dereceleri oluşturduğu için, arayüz deplasman değerlerinin elde edilmesiyle, alt yapı sistemlerindeki ortak serbestlik derecelerine ait deplasman değerleri de elde edilmiş olur. Şekilde geri_toplama fonksiyonu olarak gösterilen bölümde elde edilen arayüz deplasman değerlerinin, arayüz

sistemine katılan ortak serbestlik derecelerine göre ilgili alt yapı sistemlerine dağılımı ve bu dağılım yapıldıktan sonra ortak serbestlik derecelerine ait deplasman değerleri bilinen, alt yapı sistemlerindeki geri kalan deplasman değerlerinin hesaplanması yapılır. Şekilde son olarak her bir alt yapı sistemi için elde edilmiş deplasman değerlerinin bir araya toplanarak tüm sistemin deplasman vektörünün (d) elde edilişi gösterilmektedir.

Çoklu Sınır Çözüm Algoritmasının Başarımının Sınanması

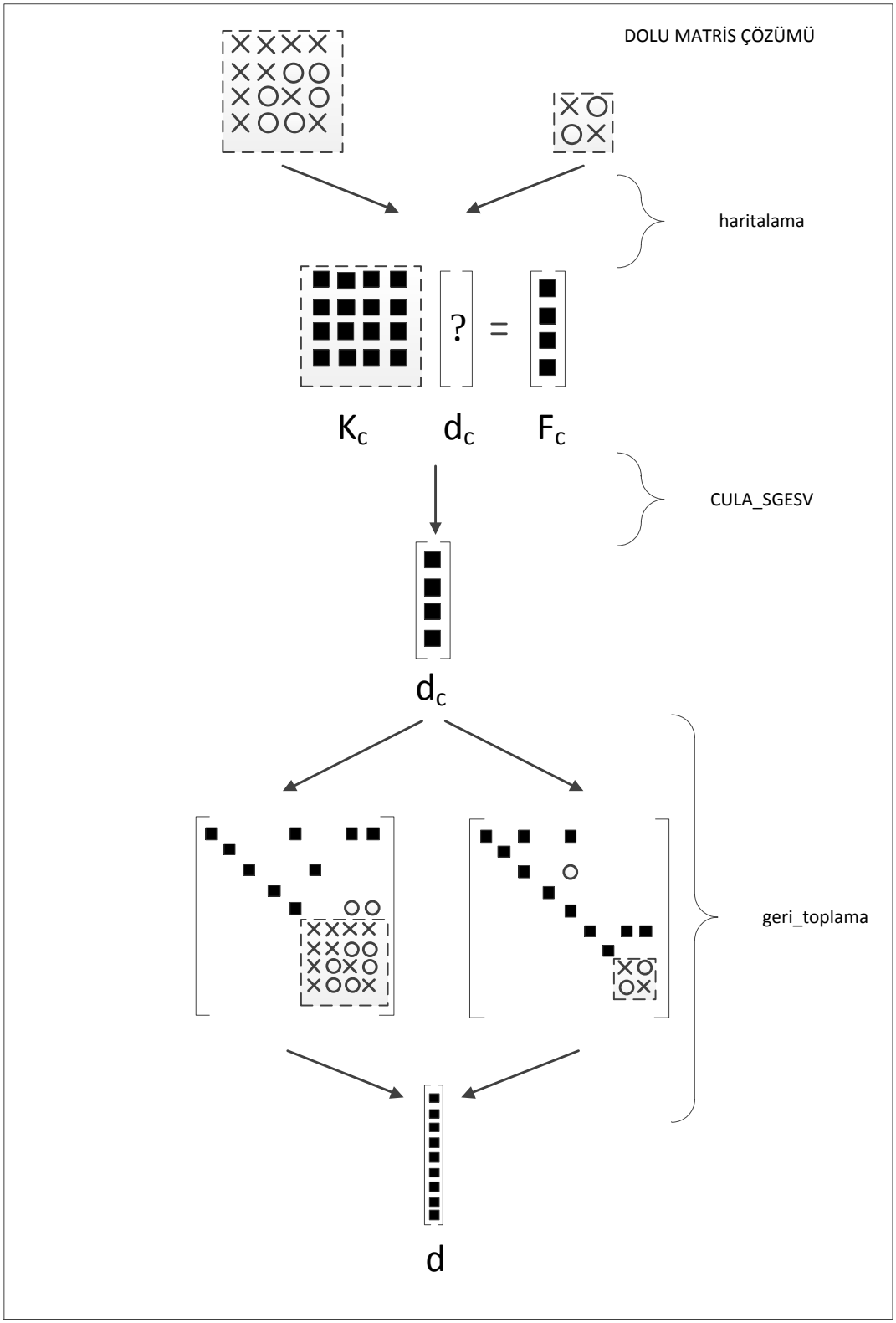
Çoklu-Sınır algoritmasının kullanıldığı seyrek matris çözücünün başarımının sınanması amacıyla farklı boyutlardaki test problemleri, çeşitli sayılarda altyapı sistemlerine parçalanarak çözülmüştür. Testler Geforce GTX275, Geforce GTX 580 Amp ve Tesla C2050 ekran kartlarında yapılarak, seyrek matris çözücünün başarımı farklı donanımlarda sınanmıştır. Kullanılan ekran kartlarının donanım özellikleri Tablo 5.1'de verilmiştir. Tablo 5.1'deki değerler doğrultusunda işlemci sayılarının ve işlem başarımlarının yüksekliği nedeniyle GTX 580 Amp ve Tesla C2050 ekran kartlarından GTX 275 ekran kartına oranla daha yüksek bir başarım beklenmektedir. Bunun yanında en büyük ana belleğe Tesla C2050 ekran kartının sahip olması sebebiyle, GTX 275 ve GTX 580 Amp ekran kartlarının bellek boyutlarını aşan test problemleri Tesla C2050 ekran kartında sınanabilmiştir. Ayrıca GTX 275 ekran kartından farklı olarak, GTX 580 Amp ve Tesla C2050 ekran kartları önceki donanımlara göre daha yüksek başarımlı olan Fermi mimarisi kullanılarak üretilmiştir.

Tablo 5.1: GTX275, GTX580 Amp ve Tesla C2050 ekran kartlarının donanım özellikleri

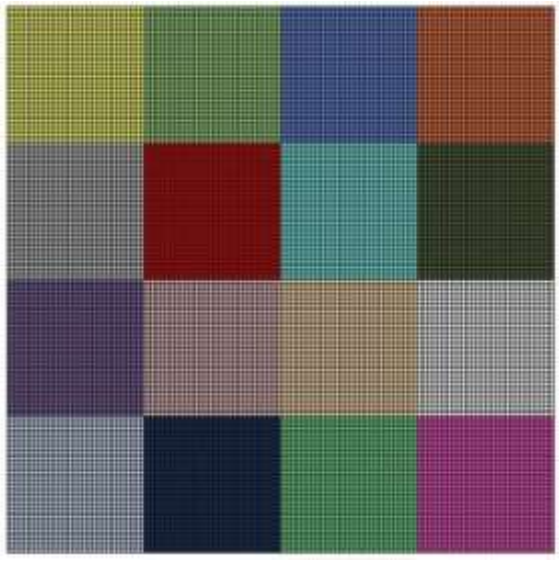
Teknik Özellik	GTX 275	GTX 580 Amp	Tesla C2050
İşlemci Sayısı	240	512	448
İşlem Başarımı (GFLOP)	1010	1580	1030
Ana Bellek (MB)	896	1536	2688
Paylaşımlı Bellek (KB)	16	48	48

Test problemleri olarak 6 serbestlik dereceli plak elemanlarından oluşan, 50×50 ve 160×160 elemanlı sistemler kullanılmıştır. Toplamda 15000 serbestlik derecesi bulunan 50×50 elemanlı sistem, 8, 16, 32 ve 64 adet altyapı sistemine bölünerek test edilmiştir. 160x160 elemanlı sistem ise 16, 32, 64 ve 128 adet altyapı sistemine parçalanmıştır. Test problemi olarak kullanılan sistemlerden 160×160 elemanlı sistemin temsili olarak 16 adet altyapı sistemine parçalanmış hali Şekil 5.7'de gösterilmiştir. Şekilde altyapıların her biri farklı renklerde temsil edilmiştir.

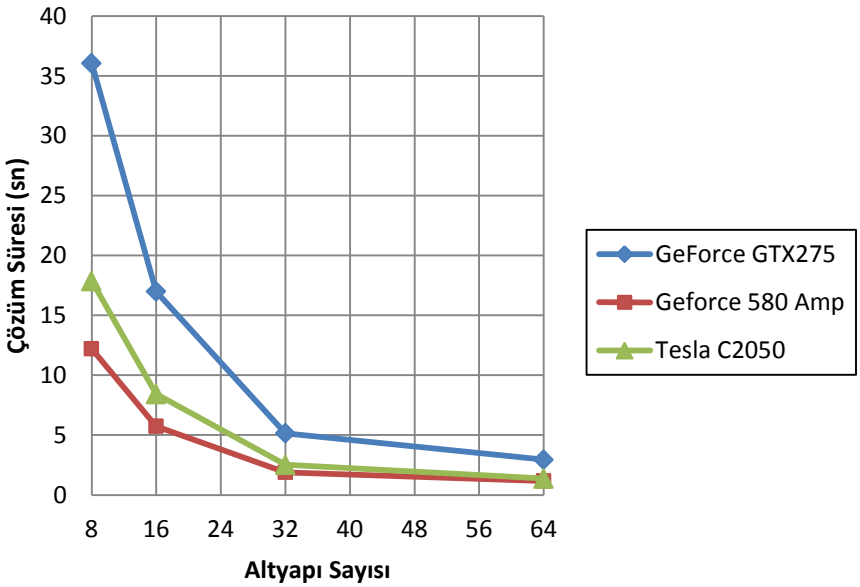
50×50 elemanlı sisteme ait test problemlerinin tümü, bütün ekran kartlarında test edilmişken, 160×160 elemanlı sistemin sınır denklemlerinin çözümü sırasında kullanılan verilerin boyutları, sistemin parçalandığı altyapı sayısının artmasıyla, GTX 275 ve GTX 580 Amp ekran kartlarının bellek kapasitesini aşmıştır. Bu sebepten dolayı toplam 153600 serbestlik derecesine sahip olan 160×160 elemanlı sistem GTX 275 ekran kartında 16 ve 32 adet altyapı sistemine parçalanarak, GTX 580 Amp ekran kartında ise 16, 32, ve 64 adet altyapı sistemine parçalanarak çözülmüştür. Şekil 15 ve Şekil 16'da test problemlerinin farklı donanımlarda, çeşitli sayıda altyapı sistemlerine parçalanarak, elde edilmiş çözüm süreleri verilmiştir.



Şekil 5.6: Sınır Matris Çözümü



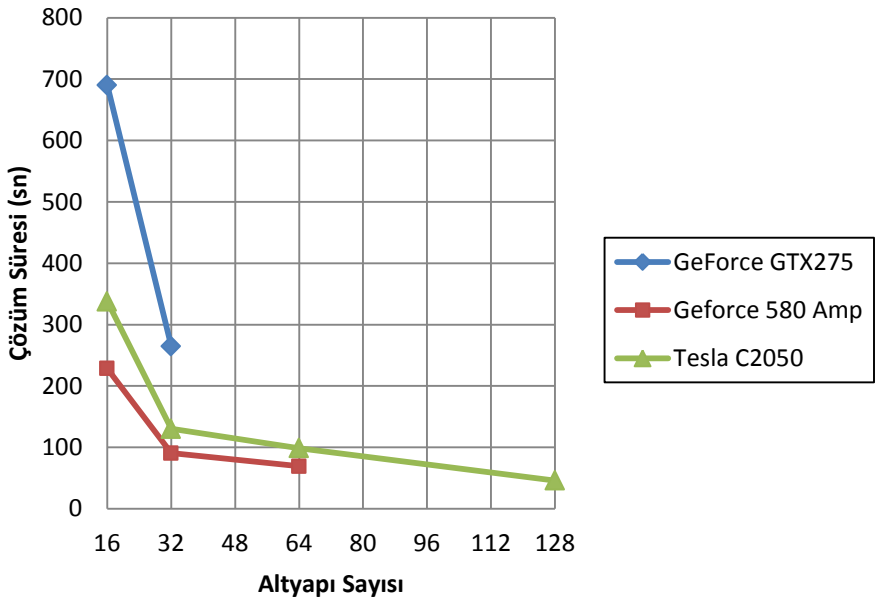
Şekil 5.7: 160×160 elemanlı sistemin temsili olarak 16 adet altyapı sistemine parçalanmış hali



Şekil 5.8: GTX 275, 580 Amp ve Tesla C2050 ekran kartlarında test edilen 8,16, 32 ve 64 adet altyapıya parçalanmış 50×50 elemanlı sistemin çözüm süreleri

Şekil 5.8'de 8, 16, 32 ve 64 adet altyapı sistemine parçalanmış 50×50 elemanlı sistemin farklı GPU donanımlarında elde edilmiş çözüm süreleri verilmiştir. Şekilde görüldüğü gibi çözüm süresi sistemin parçalandığı altyapı sistemlerinin sayısının artmasıyla azalmaktadır. Tesla C2050 ve GTX 580 Amp ekran kartlarından, sahip oldukları yüksek işlemci sayısı ve işlem başarımlarının yüksekliği sebebiyle beklenildiği gibi GTX 275 ekran kartına göre daha yüksek başarım elde edilmiştir.

Şekil 5.9'da 16, 32, 64 ve 128 altyapı sistemine parçalanmış 160×160 elemanlı sistemin farklı GPU donanımlarında elde edilmiş çözüm süreleri verilmiştir. Daha önceden de belirtildiği gibi GTX 275 ve 580 AMP ekran kartlarının bellekleri, altyapı sayısının yüksek olduğu bölümlemelerde sınır denklemlerinin çözümü için gereken verilerin boyutlarından küçüktür. Bu sebeple GTX 275 için sistem en fazla 32 parçaya 580 Amp için ise sistem en fazla 64 parçaya bölümlenerek çözülebilmiştir. Bu test probleminde de her ekran kartı için en kısa çözüm süreleri sistemin parçalandığı altyapı sayısının en yüksek olduğu yerlerde elde edilmiştir. Tesla C2050 ekran kartında sistem 128 adet altyapı ile de çözülebildiği için en kısa süre bu ekran kartıyla elde edilmiştir.

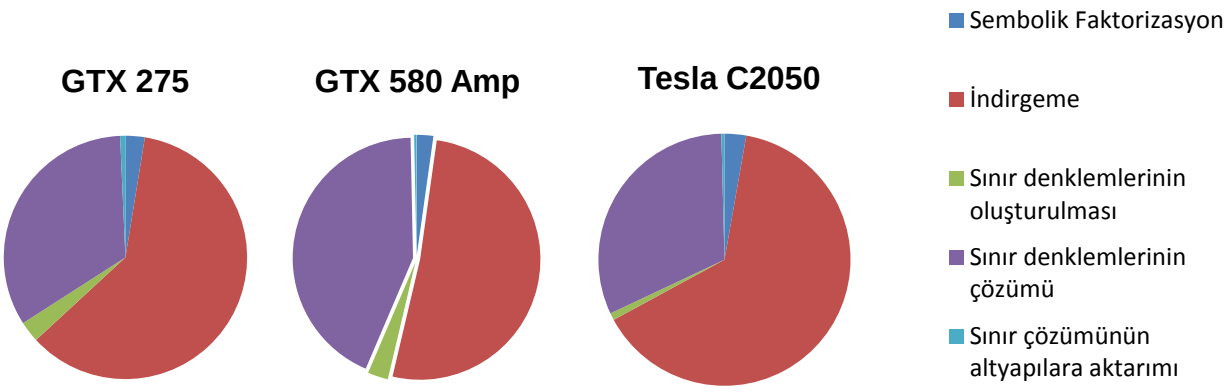


Şekil 5.9: GTX 275, 580 Amp ve Tesla C2050 ekran kartlarında test edilen 16, 32, 64 ve 128 adet altyapıya parçalanmış 160×160 elemanlı sistemin çözüm süreleri

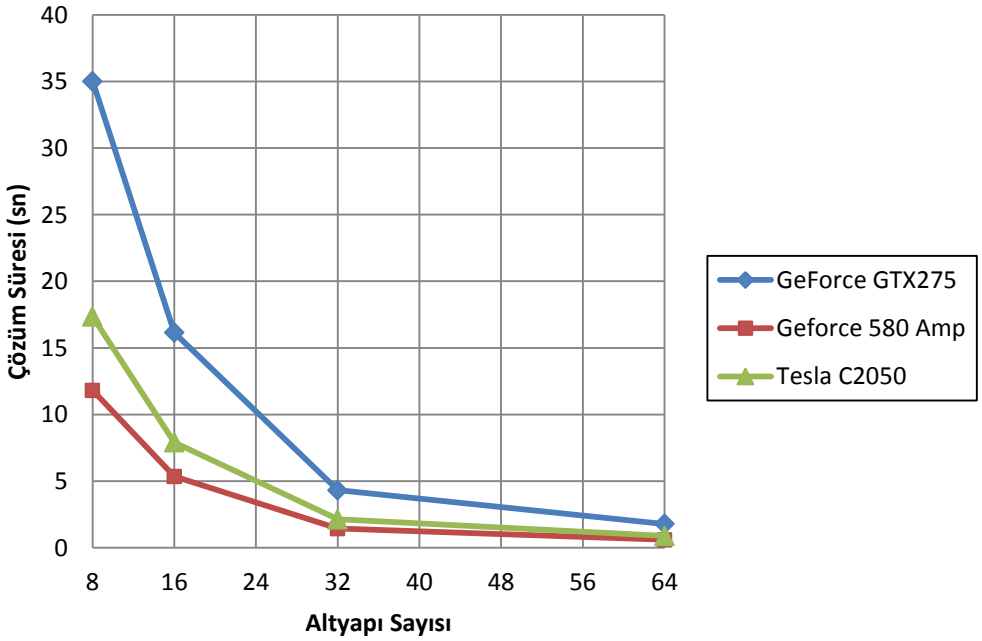
Çoklu sınır çözücünün başarımının geliştirilmesi için çözüm içerisindeki sembolik faktorizasyon, indirgeme, sınır denklemlerinin oluşturulması, sınır denklemlerinin çözümü ve sınır serbestlik dereceleri değerlerinin altyapılara aktarımı bölümlerinin çözüm süresine etkileri incelenmiştir. Bu bölümlerin çözüm süresi üzerindeki dağılımı, 64 adet altyapıya parçalanmış 50×50 elemanlı sistem için GTX 275, GTX 580 Amp ve Tesla C2050 ekran kartlarından elde edilen veriler Şekil 5.10'da verilmiştir.

Şekil 5.10'da çözüm esnasındaki işlemler sırasında geçen sürenin çözüm süresine etkisi görülmektedir. Şekilde görüldüğü gibi çözüm süresini en çok etkileyen bölümler, kırmızı renkte gösterilen indirgeme işlemi ve mor renkte gösterilen sınır denklemlerinin çözümüdür. Sistemin parçalandığı çeşitli altyapı sayılarına göre bu işlemlerde geçen süreler Şekil 5.11 ve Şekil 5.12'de gösterilmiştir.

Şekil 5.11'de görüldüğü gibi sistemin bölümlendiği altyapı sayısı arttıkça, toplam çözüm süresini en çok etkileyen işlemlerden biri olan indirgeme işleminde geçen süre azalmaktadır. Bunun yanında artan altyapı sayısına bağlı olarak sınır denklemlerinin sayısının artmasıyla, sınır denklemlerinin çözümünde geçen sürenin arttığı Şekil 5.12'de görülmektedir.



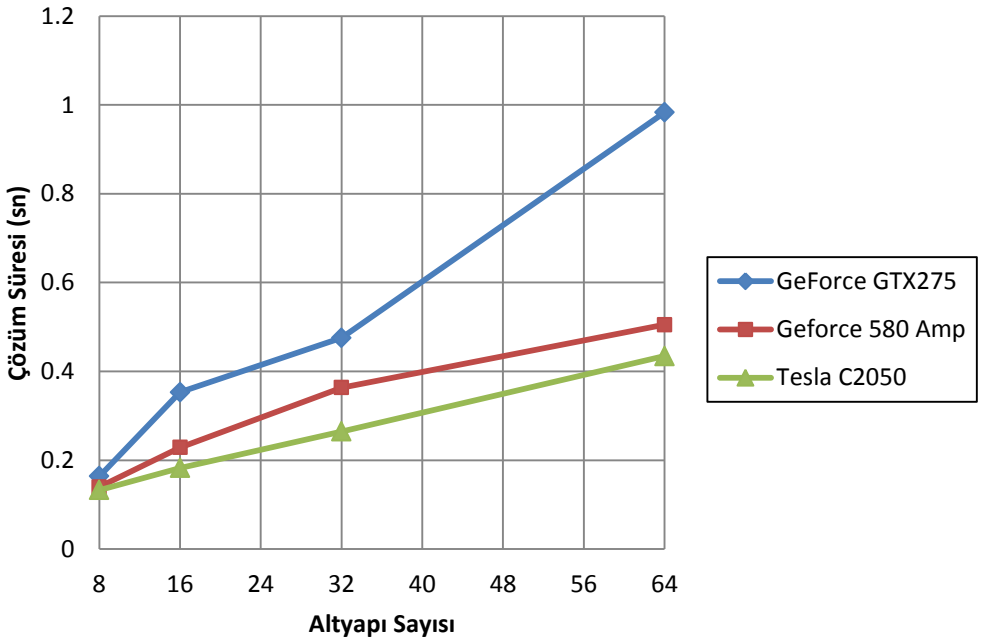
Şekil 5.10: Seyrek Çözücü içerisindeki bölümlerin 64 adet altyapıya parçalanmış 50×50 elemanlı sistemin çözüm süresine etkileri



Şekil 5.11: Çeşitli sayıda altyapılara parçalanmış 50×50 elemanlı sistemin çözümü sırasında indirgeme işleminde geçen süre

Sonuç olarak çoklu-sınır (Multiple-Front) çözüm algoritmasının kullanıldığı seyrek çözücünün başarımı sistemlerin parçalandığı altyapı sayısı arttıkça artmaktadır. Ancak artan altyapı sayısı sınır denklemlerinin boyutunu arttırmaktadır. Bu durum sınır denklemlerinin çözümü için gerekli olan verilerin ekran kartı hafızasında tutulamamasına yol açmakta, ayrıca sınır denklemlerinin çözümünde geçen süreyi arttırmaktadır. Test sonuçlarından da anlaşılacağı gibi sistemin parçalandığı altyapı sayısı arttıkça

sıkıştırma işlemi sırasında geçen süre giderek azalırken, sınır denklemlerinin çözümünde geçen süre de giderek artmaktadır. Dolayısıyla sistemin parçalandığı altyapı sayısı belirli bir değere ulaştıktan sonra, sınır denklemlerinin çözümü sistemin başarımını sınırlar hale gelecektir. Bu sebeple altyapı sistemlerin kademeli olarak sıkıştırıldığı ve çoklu-sınır çözüm algoritmasına göre daha küçük boyutlu sınır denklem sisteminin oluştuğu aşamalı çoklu sınır çözüm algoritmasını kullanan başka bir seyrek çözücünün geliştirilmesine karar verilmiştir.



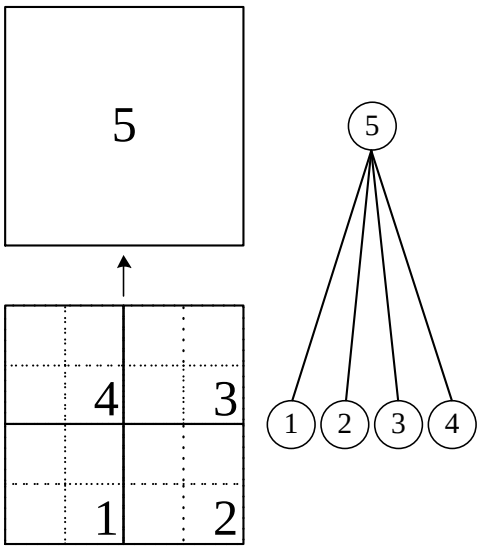
Şekil 5.12: Çeşitli sayıda altyapılara parçalanmış 50×50 elemanlı sistemin çözümü sırasında sınır denklemlerinin çözümünde geçen süre

Aşamalı Çoklu Sınır Seyrek Matris Çözücünün Geliştirilmesi

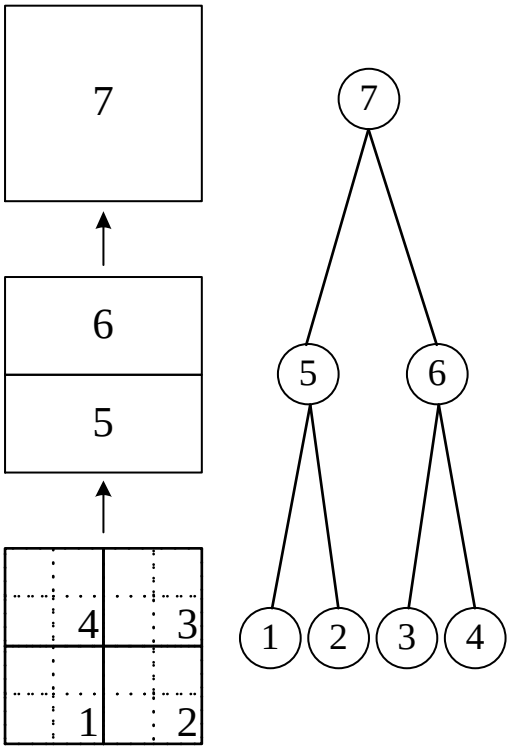
Aşamalı çoklu sınır çözüm algoritması temel olarak çoklu-sınır çözüm algoritmasına dayanmaktadır. İki çözüm algoritması arasındaki fark sıkıştırma işlemi sonucunda oluşan, altyapı sistemlerine ait Schur bileşenlerinin sınır sistemini oluşturma biçimleri arasındaki farktan kaynaklanır. Çoklu-sınır algoritmasında, altyapı sistemlerinin Schur Bileşenleri, bir kerede birleştirilerek tek bir sınır denklem sistemini oluşturulurken, aşamalı çoklu sınır çözüm algoritmasında ise Schur bileşenleri belirli gruplar halinde kendi içlerinde birleştirilir, daha sonra bu gruplar birbirleriyle birleştirilerek aşamalı olarak sınır denklemleri elde edilir. Birleştirme sırası birleştirme ağacı (assembly tree) olarak adlandırılan bir yapı altında tutulur. Şekil 5.13 ve Şekil 5.14'de, 4 adet altyapıya parçalanmış 4×4 elemanlı bir sistemin çoklu-sınır ve aşamalı çoklu sınır algoritmalarına göre oluşturulmuş birleştirme ağaçları örnekleri gösterilmiştir.

Şekil 5.13 ve Şekil 5.14'de 4×4 elemanlı sistem 4 adet altyapı sistemine parçalanmıştır. Şekillerin sol tarafında sonlu elemanlardan oluşan sistemlerinin birleşme sıraları, şeklin sağında da bu birleşme sıralarının tutulduğu birleşme ağaçları gösterilmiştir. Sonlu elemanlar kesikli çizgilerle gösterilirken, altyapılar düz çizgilerle gösterilmiştir. Çoklu sınır çözüm algoritması için örnek birleştirme ağacının

gösterildiği Şekil 5.13'de, 1, 2, 3 ve 4 numaralı altyapı sistemleri önce sıkıştırmış daha sonra sınır denklemlerinin hepsi tek aşamada birleştirilerek sistemin tamamını oluşturmuştur. Aşamalı Çoklu Sınır çözüm algoritması için örnek birleştirme ağacının gösterildiği Şekil 5.14'de ise, 1 ve 2 numaralı altyapılar sıkıştırıldıktan sonra sınır denklemleri birleştirilerek 5 numaralı altyapıyı oluştururken, 3 ve 4 numaralı altyapı sistemleri 6 numaralı altyapıyı oluşturmuşlardır. Daha sonra 5 ve 6 numaralı altyapı sistemlerine bir kez daha sıkıştırma işlemi uygulandıktan sonra bu altyapı sistemlerine ait Schur bileşenleri birleşerek tüm sistemi oluşturmuştur. Bu aşamalı birleştirme işlemi çoklu sınır çözüm algoritmasına göre avantajların temel kaynağıdır. Bu algoritmada en alt seviyede bulunan altyapılarda (1, 2, 3 ve 4 numaralı altyapılar) dolaylı adreslemenin olduğu ve başarımı düşük seyrek matris sıkıştırma işlemi yapılır, ikinci aşamada ise 5 ve 6 numaralı altyapı sistemlerinin sıkıştırılma işlemleri, başarımı yüksek dolu matris sıkıştırma yöntemleriyle yapılır. Bunların yanında en son oluşan sınır sisteminin boyutu çoklu sınır yöntemine göre daha küçük olduğu için, sınır denklemlerinin çözümü daha kısa sürede tamamlanır. Yüksek başarımlı dolu matris sıkıştırma yöntemlerinin kullanılmasına izin vermesi ve sınır çözüm işlemlerinin daha kısa sürede tamamlanması avantajlarının yanında, aşamalı çoklu sınır çözüm algoritmasının çoklu sınır çözüm algoritmasına göre bazı dezavantajları da bulunmaktadır. Geliştirilmesinin daha zor olması ve altyapıların başarımı iyileştirecek şekilde birleştirilmesini sağlayan bir birleştirme ağacının hazırlanması bu dezavantajların en önemlileridir.



Şekil 5.13: 4 adet altyapıya parçalanmış 4×4 elemanlı sistemin çoklu sınır algoritması için örnek birleştirme ağacı



Şekil 5.14: 4 adet altyapıya parçalanmış 4×4 elemanlı sistemin aşamalı çoklu sınır algoritması için örnek birleştirme ağacı

Birleştirme Ağacının Oluşturulması

Aşamalı çoklu sınır çözüm algoritmasının kullanılabilmesi için bu algoritmaya uygun indirgeme işlemlerinin aşamalı olarak yapılabileceği ve bu işlemlerin hangi sırayla yapılacağını belirten bir birleştirme ağacına ihtiyaç vardır. Bu sebeple aşamalı çoklu sınır algoritmasına uygun bir birleştirme ağacının oluşturulması için uygulaması karmaşık olmayan bir algoritma geliştirilmiştir. Bu algoritmaya göre en alt seviyede bulunan sistemin parçalandığı altyapıların birbirleriyle ortak olan serbestlik derecelerinin sayısı belirlenir. Bu sayının en büyük olduğu çiftler birbirleriyle birleşerek bir üst seviyedeki alt yapıyı oluştururlar. Altyapılar arasındaki ortak serbesttik derecelerinin sayısı aynı olduğu durumlarda, birleştirme işlemi altyapı numaralarının sırasına göre yapılır. Birleştirmeden sonra bir üst seviyede oluşan yeni altyapılar için aynı işlemler yapılır. Bu işlem bütün altyapılar birleştirilerek, tek bir sistem oluşana kadar devam eder.

Şekil 5.14'deki yapı sistemi ele alındığında, bu yapının birleştirme ağacının oluşturulması için öncelikle en alt seviyedeki altyapılar arasındaki ortak düğüm noktalarının sayısı belirlenir. Tablo 5.2'de bu altyapılar arasındaki ortak düğüm noktalarının sayısı verilmiştir.

Tablo 5.2: Şekil 5.14'de gösterilen sistemin alt yapıları arasındaki ortak düğüm noktalarının sayısı

ALTYAPI NO	1	2	3	4
1	-	3	1	3
2	3	-	3	1
3	1	3	-	3
4	3	1	3	-

Tablo 5.2'de görüldüğü gibi altyapılar arasında en fazla 3 adet ortak düğüm noktası bulunmaktadır. Bu değer birden fazla altyapı çifti için geçerli olduğu için, birleştirme işlemi altyapı numarası sırasına göre yapılır, sonuç olarak 1.altyapı, 2.altyapı ile, 3.altyapı da 4.altyapı ile birleştirilir. Bir üst seviyede 5. ve 6. altyapılar olmak üzere geriye sadece iki altyapı kaldığı için, bu altyapılar doğrudan birbirleriyle birleştirilir.

Aşamalı Çoklu Sınır Algoritması için Çoklu Yoğun Matrislerin İndirgenmesi

Aşamalı çoklu sınır çözüm algoritmasında, birleştirme ağacının en alt seviyesinde bulunan altyapıların indirgeme işlemleri, seyrek matris indirgeme metotlarıyla yapılırken, birleştirme ağacının ara basamaklarında bulunan altyapılara ait matrislerin indirgeme işlemleri, yoğun matris indirgeme metotları ile yapılır. Bu işlemlerin yapılabilmesi amacıyla grafik işlemcilerin kullanıldığı, birleştirme ağacında aynı seviyede bulunan bütün matrislerin paralel olarak indirgenmesini sağlayan bir algoritma gereklidir. GPU ile çalışan dolu matris indirgeme algoritmalarının mevcut kütüphanelerde bulunmaması, bu algoritmaları da içeren bir dolu matris çözücünün geliştirilmesini gerektirmiştir.

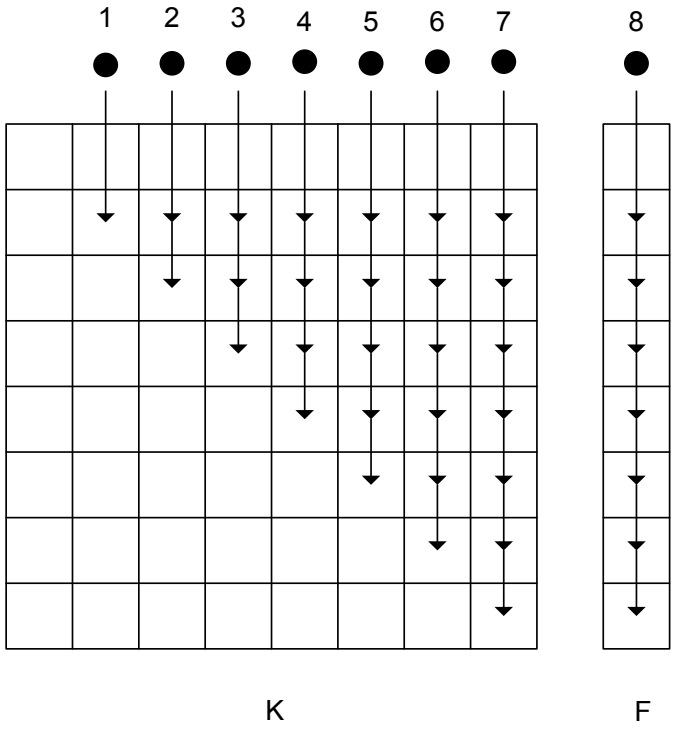
Çözücünün hem seyrek matrislerin çözüm algoritmasında hem de daha büyük sistemlerin sıkıştırma ve çözümünde kullanılacak olması sebebiyle çözücü, "Faktorizasyon ve Çözüm", "İndirgeme" olarak iki bölümden oluşmaktadır.

Ayrıştırma ve Çözüm

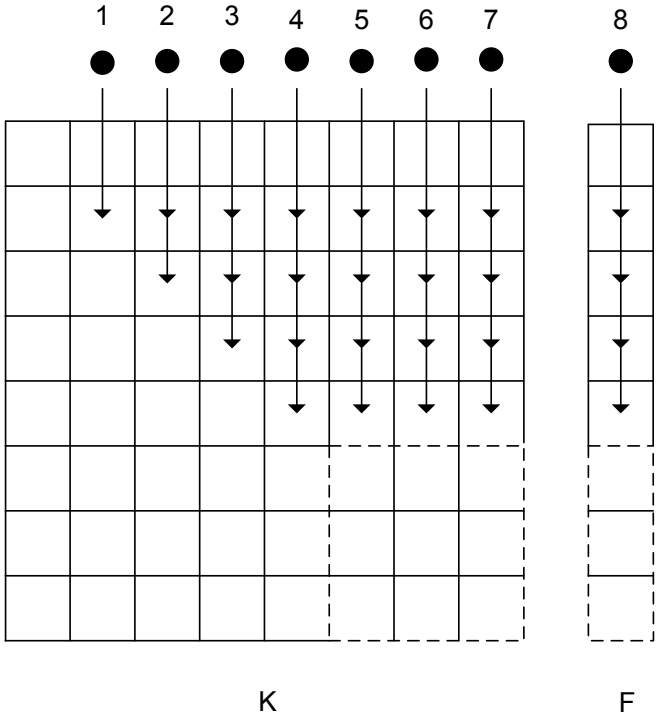
Karşılaşılan problemlerdeki dolu matrislerin simetrik olması sebebiyle, depolama boyutlarını ve işlem sayısını azaltmak amacıyla algoritmada matrislerin sadece üst üçgensel bölümleri kullanılmıştır. Algoritma temel olarak Gauss ayrıştırma işlemine dayanır. Seri algoritmadan farklı olarak her kolondaki terimlerin hesaplanmasından ekran kartı üzerindeki farklı bir işlemci sorumludur. Şekil 5.15'de algoritmadaki ayrıştırma işlemi gösterilmiştir. Şekilde 8x8 boyutunda dolu matris K ile 8x1 boyutundaki vektör F ile gösterilmiştir. İşlemcilerin siyah daireler ile gösterildiği şekilde oklar işlemcilerin yaptığı işlemleri belirtmektedir. Örneğin ikinci satırın hesaplanması işlemcilerin kendilerine ait ilk işlemleri bitirmesiyle tamamlanır. Her işlem sırasının tamamlanmasıyla yeni bir satır hesaplanmış ve bir sonraki işleme katılacak olan işlemci sayısı da bir azalmış olur.

İndirgeme

Gauss ayrıştırma işleminin belirlenen bir satırda durdurulmasıyla da dolu matrisin Schur Bileşeni, vektörün de sıkıştırılmış hali hesaplanmış olur. Şekil 5.16'da algoritmadaki sıkıştırma işlemi verilmiştir. Şekilde Schur bileşeni ve vektörün sıkıştırılmış hali kesik çizgilerle çevrilmiş alanlar olarak gösterilmiştir.



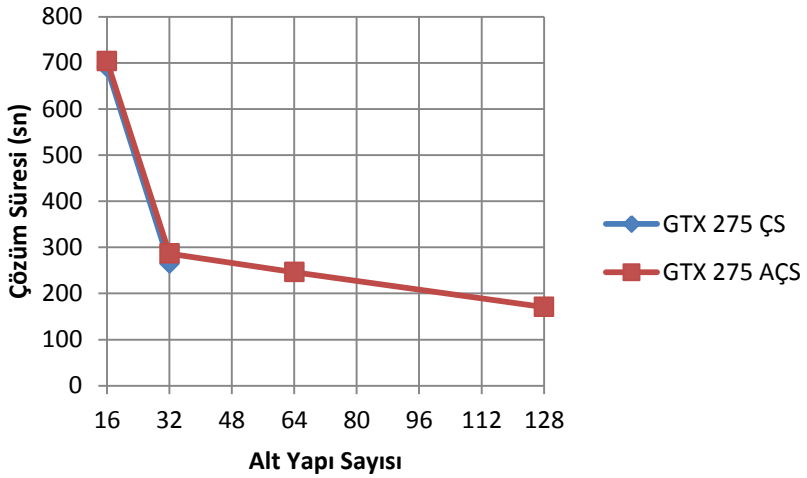
Şekil 5.15: Dolu Matris Ayrıştırılması



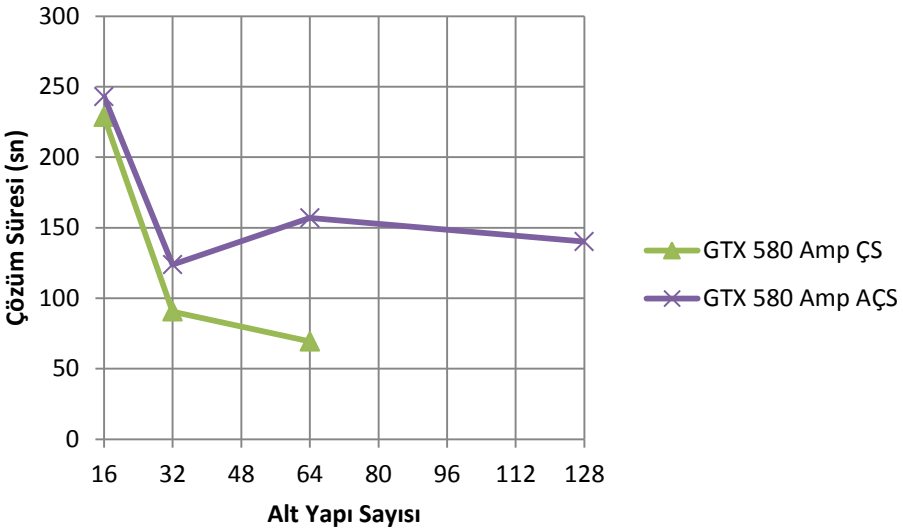
Şekil 5.16: Dolu Çözücü (Column Wise) İndirgeme

Aşamalı Çoklu Sınır Algoritmasının Kullanıldığı Seyrek Matris Çözücünün Başarımının Sınanması

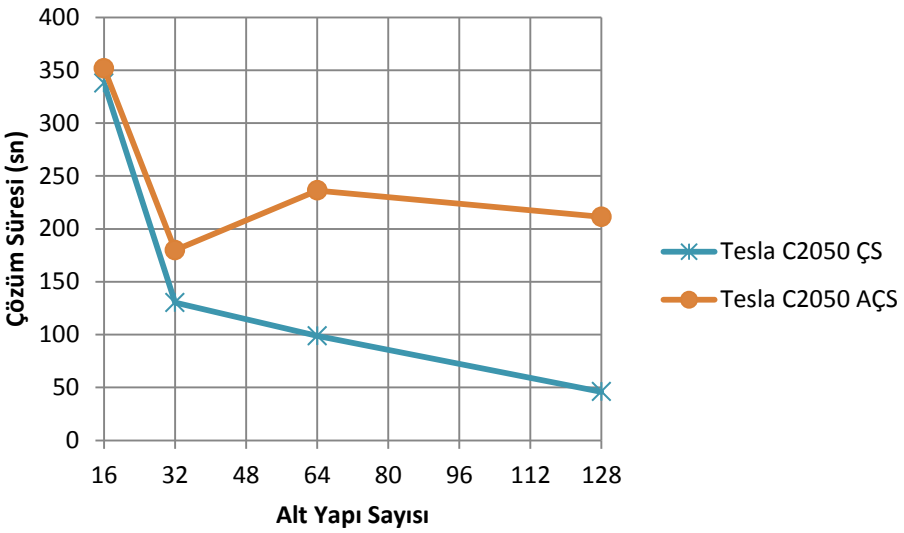
16, 32, 64 ve 128 adet altyapıya bölünmüş 160×160 elemanlı sistemin, çoklu sınır yöntemi ve aşamalı çoklu sınır yöntemi ile, GTX 275, GTX 580 Amp ve Tesla C2050 ekran kartlarından elde edilen çözüm süreleri, Şekil 5.17, Şekil 5.18 ve Şekil 5.19'de gösterilmiştir.



Şekil 5.17: Çeşitli sayıda altyapılara bölünmüş 160×160 elemanlı sistemin çoklu sınır (ÇS) ve aşamalı çoklu sınır (AÇS) yöntemiyle GTX 275 ekran kartından elde edilen çözüm süreleri



Şekil 5.18: Çeşitli sayıda altyapılara bölünmüş 160×160 elemanlı sistemin çoklu sınır (ÇS) ve aşamalı çoklu sınır (AÇS) yöntemiyle GTX 580 Amp ekran kartından elde edilen çözüm süreleri



Şekil 5.19: Çeşitli sayıda altyapılara bölünmüş 160×160 elemanlı sistemin çoklu sınır (ÇS) ve aşamalı çoklu sınır (AÇS) yöntemiyle Tesla C2050 ekran kartından elde edilen çözüm süreleri

Şekil 5.17'de verilen GTX 275 ekran kartından elde edilen sonuçlarda aşamalı çoklu sınır algoritmasının kullanıldığı seyrek matris çözücünn başarımının, sistemin bölündüğü altyapı sayısının artmasıyla arttığı gözlenmektedir. Bu artışın ana sebebi sistemin daha çok altyapıya parçalanmasıyla seyrek matris indirgeme bölümünde geçen sürenin kısalmasıdır. GTX 275 ekran kartının hafızasının test edilen sistem için yeterli olmaması sebebiyle altyapı sayısındaki artışın etkileri, çoklu sınır algoritmasının kullanıldığı 64 ve 128 adet altyapılı sistemler için gözlenememektedir. Bu sebeple 64 ve 128 adet altyapıya bölünmüş sistemin aşamalı çoklu sınır yöntemi ile çözümünden elde edilen süreler, 32 adet altyapıya bölünmüş sistemin çoklu sınır yöntemiyle çözümünden elde edilen sürelerden daha kısadır.

Şekil 5.18 ve Şekil 5.19'da gösterilen, GTX 580 Amp ve Tesla C2050 kartlarından elde edilen sonuçlarda ise altyapı sayısının artmasıyla, seyrek matris indirgeme süresi kısalırken, bileştirme ağacının ara aşamalarında uygulanan yoğun matris indirgeme süreleri artmaktadır. Ayrıca, yoğun matris sürelerinde geçen sürenin, sınır sisteminin çözümünde geçen süreden çok daha uzun sürmesi sebebiyle, GTX 580 Amp ve Tesla C2050 ekran kartlarında çoklu sınır algoritmasının kullanıldığı seyrek çözücüden elde edilen çözüm süreleri, aşamalı çoklu sınır yönteminin kullanıldığı seyrek çözücüden elde edilen çözüm sürelerinden daha kısadır. Dolayısıyla her ne kadar aşamalı çoklu sınır çözücü daha az bellek kullanarak daha büyük modellerin çözülmesini sağlamış olsa da çözüm süresi genelde çoklu sınır çözücünün sürelerinden daha yüksektir.

Yoğun Matrislerin Grafik İşlem Birimleri Yardımıyla İndirgenmesi

Günümüzde bütün seyrek matris çözücüler, seyrek matrislerden küçük boyutlu yoğun matrisler oluşturup, matris hesaplarını bu yoğun matrisler üzerinden yapmaktadırlar. Dolayısıyla gerek MİB'lerde, gerekse GİB'lerde çalışan çözücüler olsun, yoğun matrislerin indirgenmesi veya ayrıştırılmasında elde edilecek en ufak bir hızlanma, bu tip çözücülerin başarımını ciddi şekilde arttıracaktır. Bu sebepten dolayı bu bölümde GİB'lerin yoğun matris indirgemesindeki başarımları incelenmiştir.

Bilinmeyenler vektörü *x*, bilinmeyenlerin çarpanlarından oluşan matris *A*, her bir denklemin sabit eşitliğinden oluşan vektör *b* olarak adlandırılırsa, doğrusal bir denklem sistemi şu şekilde ifade edilebilir;

$$[A]\{x\} = \{b\} \tag{5.1}$$

Bu doğrusal denklem sisteminin çözümü iki ana basamaktan oluşmaktadır. İlk olarak *A* matrisi *L*L^T*'ye eşit olacak şekilde üçgen çarpan matrislerine ayrıştırılır. Sonrasındaysa, geri yerine koyma işlemiyle bilinmeyenler elde edilir.

Doğrusal denklem sistemlerinin çözümü satırlar (İng. *Row-wise*), sütunlar (İng. *Column-wise*) veya bloklar (İng. *Block-wise*) halinde olmak üzere farklı şemalar takip edilerek yapılabilir. Bilinmeyenlerin çarpanlarından oluşan *A* matrisi tam (İng. *Full Format*) veya sıkıştırılmış (İng. *Packed Format*) düzende tutulabilir (LAPACK 2012). Sıkıştırılmış düzen, matrisin sadece alt veya üst üçgen yarısını depoladığı için tam düzenin yarısı kadar belleğe ihtiyaç duymaktadır. Ancak, bu düzende bloklar halinde işlem yapabilmek mümkün değildir. O nedenle bu çalışmada *A* matrisinin tamamı bellekte tutulmuş ve algoritma bloklar halinde çalışacak şekilde tasarlanmıştır. Satırlar ve sütunlar halindeki şemalar ise ilgili blok boyutunu bire eşitleyerek elde edilebilmektedir.

N×N boyutlu bir matrisin *i* derecesine indirgenmesini (İng. *Static Condensation* veya *Schur Complement Computation*) tanımlayabilmek için öncelikle matrisin *i* numaralı satır ve sütunundan parçalanması gerekmektedir. A_{ii}, A_{ij}, A_{ji}, A_{jj} matrislerinin sırasıyla i×i, i×j, j×i ve j×j boyutludur ve (i+j)×(i+j) boyutlu A matrisi şöyle gösterilebilir;

$$A = \begin{bmatrix} A_{ii} & A_{ij} \\ A_{ji} & A_{jj} \end{bmatrix} \tag{5.2}$$

Eğer, A_{jj} matrisin tersi alınabiliyorsa, N×N boyutlu A matrisinin *i* derecesine indirgenmesi şu şekilde tanımlanabilir;

$$\widehat{\mathrm{A}}_{\mathrm{jj}} = \mathrm{A}_{\mathrm{jj}} - \mathrm{A}_{\mathrm{ji}}{\mathrm{A}_{\mathrm{ii}}}^{-1}\mathrm{A}_{\mathrm{ij}} \tag{5.3}$$

Bu tanımda yer alan A_{jj} matrisin tersini hesaplayarak yukarıdaki işlemi yapmaktansa, bütün matrisi *(i+1)* numaralı satırına kadar Cholesky yöntemiyle ayrıştırılmasıyla elde edilen $\widehat{A_{JJ}}$ de aynı sonucu verecektir (Zhang 2005).

Seri MİB Algoritması

Cholesky yöntemi, N×N bir simetrik pozitif tanımlı matrisi, üçgen matris L ve onun devrik halinin çarpımı şeklinde ayrıştırmaktadır. Bu seri algoritmada, *A* matrisinin sadece alt yarısı bellekte tutulmaktadır ve hesaplanan çarpanlar ilgili matris değerinin üzerine yazılmaktadır. İşlemin *k* basamağında çarpanlarına ayrıştırılmamış olan *(N – k * nb)×(N – k * nb)* boyutlu matrisi $A(k)$, ayrıştırma sonucunu $L(k)$ olarak adlandırırsak geri kalan işlem şu şekilde ifade edilebilir;

$$A(k) = L(k)L^T(k) \tag{5.4}$$

Blok boyutu *nb* ise ve bu blok boyutuna uygun olarak matrisler parçalanacak olursa şu eşitlik elde edilir;

$$\begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} = \begin{bmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{bmatrix} \tag{5.5}$$

Bu denklemde yer alan A_{11} matrisi *nb×nb*, A_{21} matrisi *(N – (k+1) * nb)×nb* ve A_{22} matrisi *(N – (k+1) * nb)×(N – (k+1) * nb)* boyutlarındadır. L_{11} ve L_{22} matrisleri ise çarpanlara ayrıştırma sonuçlarını içeren üçgen matrislerdir.

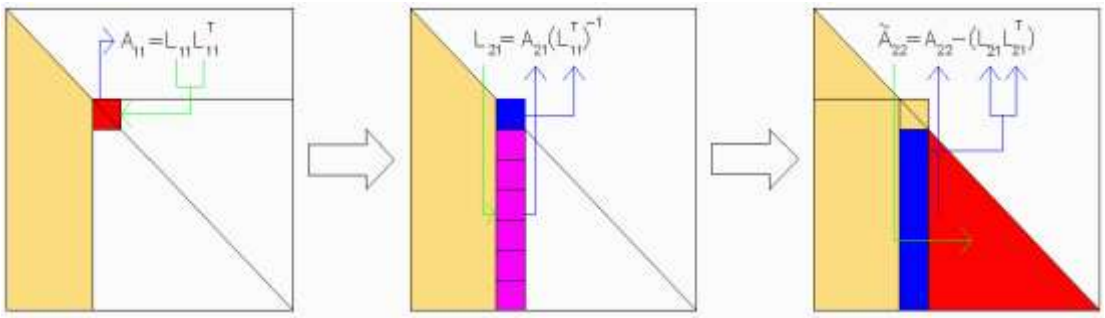
Bloklar halinde çalışan Cholesky çarpanlara ayrıştırma işlemi şu şekilde açıklanabilir. A_{11} matrisinin çarpanlarına ayrıştırılmasının sonucunda elde edilen L_{11} matrisinin bilindiği kabul edilirse, blok denklemlerini yeniden düzenleyerek

$$L_{21} \leftarrow A_{21}(L_{11}^T)^{-1} \tag{5.6}$$

ve

$$\tilde{A}_{22} \leftarrow A_{22} - L_{21}L_{21}^T = L_{22}L_{22}^T \tag{5.7}$$

elde edilebilir. Şekil 20, bu yöntemle L_{11} ve L_{21} kolonlarının nasıl hesaplandığını ve A_{22} matrisinin nasıl güncellendiğini göstermektedir. Bu basamağın sonucunda elde edilen *(N - nb)×(N - nb)* boyutlu $\tilde{A}_{22}$ matrisi ile bu işlemleri tekrar ederek çarpanlara ayrıştırma tamamlanır.



(a) (b) (c)

Şekil 5.20: Bloklu Cholesky ayrıştırması basamakları

LAPACK kütüphanesinde (Lapack 2012) yer alan bu yöntem aşağıda yer alan işlemleri içermektedir:

1. **POTF2**: Köşegende yer alan A_{11} blok matrisinin çarpanlarına ayrılması;
   $$A_{11} \rightarrow L_{11}L_{11}^T \tag{5.8}$$

2. **TRSM**: L_{21} sütun panelinin hesaplanması;
   $$L_{21} \leftarrow A_{21}(L_{11}^T)^{-1} \tag{5.9}$$

3. **GEMM + SYRK**: Geriye kalan $\tilde{A}_{22}$matrisinin güncellenmesi;
   $$\tilde{A}_{22} \leftarrow A_{22} - L_{21}L_{21}^T = L_{22}L_{22}^T \tag{5.10}$$

Bu işlemlerin tekrar tekrar $\tilde{A}_{22}$ matrisine uygulanmasıyla A matrisinin Cholesky çarpanları elde edilebilir.

MİB-GİB Karma Ortam Uyarlaması

Yöntem

Burada detayları aktarılacak olan MİB ve GİB'i bir arada kullanabilen Cholesky çarpanlara ayrıştırma yöntemi, temelde LAPACK kütüphanesinde (Lapack 2012) yer alan yöntemden esinlenmiştir. Bu algoritma POTRF, TRSM, GEMM ve SYRK fonksiyonlarına yapılan bir dizi çağırımdan oluşmaktadır. Kısaca açıklamak gerekirse seçilmiş olan boyutlardaki A_{11} küçük blok matrisi ayrıştırılır ve bu sonuçlar kullanılarak L_{21} kolon paneli hesaplanır. Sonrasındaysa geriye kalan A_{22} matrisi güncellenir (bkz. Şekil 5.20).

Karma sistemlerde verilerin sistemi oluşturan donanımlar arasında sürekli ileri geri aktarılması, sistemin başarımını düşürmektedir. Bu nedenle verilerin en uygun donanımda işlenmesi ve donanımlar arasında olabildiğince az veri aktarımı yapılması gerekmektedir. Geliştirilen algoritma bu temele dayanmaktadır. Algoritma, çarpanlara ayrılacak *A* matrisinin tamamının MİB belleğinden GİB genel aygıt belleğine kopyalanmasıyla başlamaktadır. Ancak bu matrisin GİB'in genel aygıt belleğine sığabilecek boyutlar da olması ve bu boyutun GİB kernel fonksiyonlarıyla uyumlu olacak şekilde blok boyutunun katı olması gerekmektedir. Blok boyutunun katı olmayan boyutta olan matrisler ise bu boyuttan büyük, en küçük katlı boyuta ulaşana kadar köşegeni 1, geriye kalan elemanları 0 ile doldurarak uygun boyuta getirilir.

Algoritma, Denklem 5.5'de yer alan *nb*×*nb* boyutlu A_{11} matris bloğunun, Şekil 5.20 (a)'da gösterildiği gibi çarpanlarına ayrılmasıyla başlayacaktır. Bu matris bloğu tercihen 16 ile 512 arasında boyutlarda olabilmektedir. Bu boyutlardaki bir matrisin bloklu bir metotla çarpanlarına ayrıştırılmasındansa MİB aracılığıyla bloksuz olarak çarpanlarına ayrıştırılması daha hızlı sonuçlar üretmektedir. Bu nedenle A_{11} matrisi MİB tarafından çarpanlarına ayrılır ve sonuç GİB genel aygıt belleğinin ilgili kısmına kopyalanır. Sonrasında GİB tarafından kolon panellerinin hesaplanmasına başlanılır. Her iş grubu kendi sorumluluğunda olan L_{11} kolon parçasını ve ilgili A_{21} matrisini kendi paylaşımlı hafızasına kopyalar ve TRSM kernel fonksiyonu yardımıyla L_{21} kolon bloğunu hesaplar. (bkz. Şekil 5.20 (b)) Sonuçlar GİB genel aygıt belleğinde ilgili bölümün üzerine kopyalanır. Bu işlemin ardından geriye kalan matrisin güncellenmesi gerekmektedir. Benzer şekilde yine her iş grubu kendi sorumluluğundaki L_{21} satır panelini ve ilgili A_{22} kolon bloğunu kendi paylaşımlı hafızasına kopyalar ve SYRK ile GEMM kernel fonksiyonlarını kullanarak kolon bloğunu günceller. (bkz. Şekil 5.20 (c)) Sonuçlar, yine genel aygıt belleğindeki ilgili yerlerin üzerine yazılır. Algoritma, yukarıda açıklanan işlemlerin her seferinde bir sonraki A_{11} köşegen blok matrisinin çarpanlarına ayrılmasıyla tekrarlanır.

Eniyileme

Detayları anlatılan karma ortam yoğun matris indirgeme algoritması için gerekli olan bütün kernel fonksiyonları CUDA içerisinde yer alan CUBLAS kütüphanesi (CUBLAS 2012) tarafından sağlanmaktadır. Ancak başarımı daha yukarıya taşıyabilmek amacıyla bunlara alternatif eniyilenmiş kernel fonksiyonları geliştirilmiştir. Başarımının arttırılması için iğnelenmiş bellek kullanımı, paylaşımlı bellek kullanımı ve döngü geri sarma (İng. *Loop Unrolling*) gibi bazı teknikler uygulanmıştır.

Bir önceki bölümde açıklandığı üzere her bir basamakta A_{11} matris bloğunun MİB'de çarpanlarına ayrıştırılmasıyla elde edilen sonuçlar MİB'den GİB'e kopyalanmaktadır. İğnelenmiş bellek kullanımıyla MİB belleğine GİB'in direk erişimi sağlanmış ve kopyalama başlangıç gecikmeleri (İng. *Startup Latencies*) en aza indirilmiştir. Böylece *cudaMemcpy* çağırımları hızlandırılmıştır.

Volkov çalışmasında (Volkov and Demmel 2008), iş parçacıklarını olabildiğince paralelleştirmenin (İng. *Multithreading*) aritmetik ve bellek gecikmelerini gizlemede tek yol olmadığını göstermiştir. Kullanılan iş

parçacığı sayısını azaltarak, kernel fonksiyonu içerisinde, yazmaçlar yardımıyla koşut seviyesinde paralelliği arttırarak bağımsız işlemlerin ivedi şekilde çalıştırılmasını sağlamaktadır. Yazmaçlar, işlem birimi tarafından en hızlı şekilde erişilebilen bellekler oldukları için paylaşımlı bellek kullanımında ortaya çıkan bellek gecikmelerini gizleyebilmektedirler. Ancak paylaşımlı bellek, genelde her kernel fonksiyonunun sonunda yer alan sonuç hazırlama safhasında iş parçacıklarının birbirleriyle iletişim kurabilmeleri açısından ciddi bir öneme sahiptir. GİB'ler için yazmaç sayısı sabit olduğu için artan iş parçacığı sayısı, her bir iş parçacığı başına düşen yazmaç sayısını düşürmektedir. Bu nedenle iş parçacığı sayısı her iş parçacığına yeterli miktarda yazmaç sağlayabilecek kadar az, iş parçacığı paralelliğini sağlayacak kadar çok olmalıdır. Volkov çalışmasında 64 adet iş parçacığının çalıştırılmasının aritmetik ve bellek gecikmelerini gizlemek için yeterli olduğu sonucuna varmıştır (Volkov and Demmel 2008). Bu nedenle, bu çalışmada da blok boyutu 64 olarak seçilmiştir.

Bu çalışmada uygun yerlerde paylaşımlı bellekten faydalanılmaktadır. Örneğin, kolon panellerinin güncellenmesi ve matrislerin matrislerle çarpımında paylaşımlı bellek kullanılmıştır. Veriler bellekte sürekli olarak tutulduğundan ve verilere doğrusal olarak iş parçacığı indeksleri ile ulaşılarak paylaşımlı bellek erişim sıkıntılarının (İng. *Bank Conflicts*) önüne geçilmiştir.

Son olarak döngü geri sarma işlemi yardımıyla döngülerde her basamakta yapılan işaretçi işlemleri ve döngü sonlandırma işlemleri gibi döngü kontrol işlemleri elenebilir. Döngüler yerine döngü içerikleri tek tek açılarak yazılabilir ya da *#pragma unroll* yönergesi ile derleyiciye bu işlemi her döngüye uygulaması söylenebilir. Bu yöntemle azaltılan işlem sayısı nedeniyle daha iyi bir başarım elde edilmektedir. Rank güncellemeleri sırasında bu yöntem uygulanmıştır.

Başarım

Sınama Düzeneği ve Yöntemi

Geliştirilmiş olan algoritma, dört farklı sistemde sınanmıştır. Sunulan algoritmanın başarımının sınanması amacıyla sadece GALA isimli bu karma algoritma değil, Intel'in sağlamış olduğu salt MİB kullanan MKL' de yer alan Cholesky algoritması ve salt GİB kullanan CULA kütüp hanesinde yer alan Cholesky algoritması da karsılaştırma için teste tabii tutulmuştur.

Tablo 5.3: Başarım sınama düzenekleri

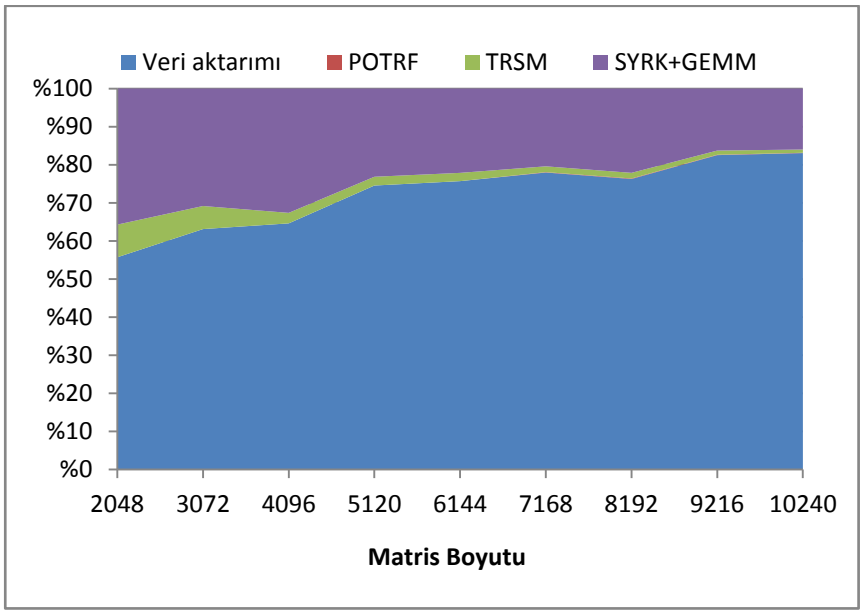
	SİSTEM I	SİSTEM II	SİSTEM III	SİSTEM IV
MİB	Intel i7 860	Intel i7 860	Intel Q8400	Intel E5620
Bellek	4GB	4GB	4GB	24GB
GİB	GTS250	GTS450	GTX580 AMP	Tesla C2050
Çekirdek Sayısı	128	192	512	448
İşlemci Hızı (MHz)	1836	1566	1544	1150
Bellek Hızı (MHz)	1100	1804	2004	3000
Bellek	512 MB	1 GB	1536 MB	2688 MB
Bellek Arayüzü	256-bit	128-bit	384-bit	384-bit
Bellek Bant Genişliği (GB/s)	70.4	57.7	192.4	144

Testler sırasında, girdi olarak 2048×2048 boyutlu rastgele matristen başlayarak her bir testte boyutu 1024 arttırarak donanıma sığabilen en büyük matris boyutuna kadar devam edilmiştir. İğnelenmiş bellek kullanıldığından hafızaların tam kapasite olarak kullanılabilirliği garanti edilememektedir (CUDA 2012). Her bir test basamağında aynı değerleri içeren aynı boyuttaki matrisler son satıra kadar farklı algoritmalar aracılığıyla indirgenmiştir. Testler işlem hassasiyetlerine göre iki ayrı grupta, uygun olan donanımlar ve algoritmalarla yapılmıştır.

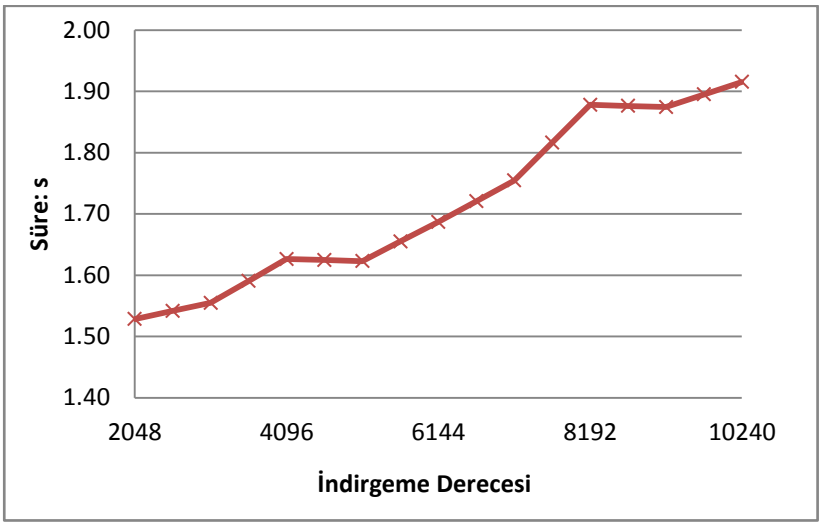
Tek Duyarlıklı Test Sonuçlarının İrdelenmesi

Bu çalışmada yer alan algoritma üç ana hesap basamağı ve veri alış-verişi basamaklarından oluşmaktadır. Şekil 5.21, bu basamaklar için harcanan zamanın bütün hesap süresi içerisindeki yüzdesini göstermektedir.

Açıkça görüldüğü üzere, veri alış-verişi en baskın işlemdir. Geriye kalan matrisin ve kolon panelinin güncellenmesi en çok zaman alan ikinci ve üçüncü işlemlerdir. Ancak bu işlemler GİB tarafından yapıldığı için daha iyi bir şekilde ölçeklenmektedir. Bu nedenle, matris boyutları büyüdükçe veri alış-verişi daha baskın hale gelmektedir. Ayrıca, blok boyutu matris boyutuna nazaran çok küçük olduğu için POTRF kayda değer bir zaman almamaktadır.



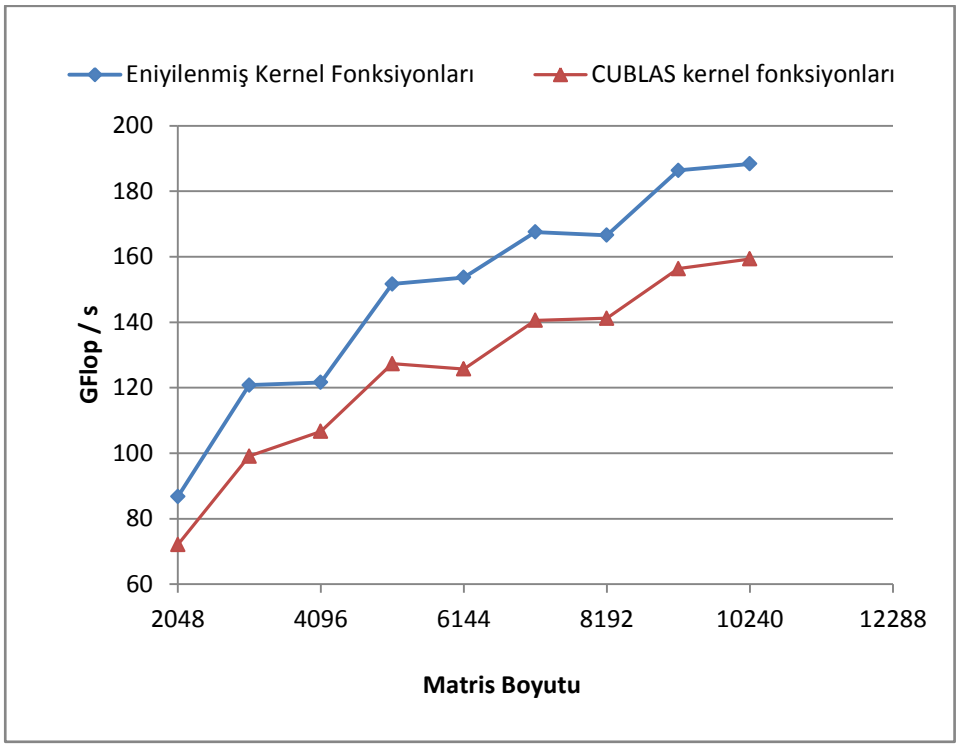
Şekil 5.21: İndirgeme süresi detayları (Sistem I)



Şekil 5.22: 10240×10240 matrisin farklı derecelere indirgenmesi

Karşılaştırma amacıyla son satıra kadar işletilen, ancak kısmi çarpanlara ayırma olarak çalıştırılmak üzere tasarlanmış olan algoritmanın, 10240×10240 boyutundaki matrisin farklı derecelere indirgenmesiyle elde edilen sonuçlar Şekil 5.22'de yer almaktadır. Görüldüğü üzere karmaşıklık seviyesi O($n^3/3$) olan seri algoritmanın karma ortam için yeniden düzenlenmesiyle parabolik bir karışıklık elde edilmektedir. Şekil 5.22'de görüldüğü üzere bu girdi matrisinin yarısına kadar indirgenmesi 1.61 saniye sürerken son satırına kadar indirgenmesi 1.92 saniye sürmektedir.

Kernel fonksiyonlarına yapılan eniyilemeler sonucunda elde edilen başarım artışı Şekil 5.23'den incelenebilir. Grafikte de görüldüğü üzere matris boyutu büyüdükçe, veri aktarımı gecikmelerini gizlemek amacıyla yapılmış olan eniyilemeler sonucu, elde edilen başarım artışı da artmaktadır. Kernel fonksiyonlarına yapılmış olan eniyilemeler ortalama olarak başarımı %20 oranında arttırmıştır.

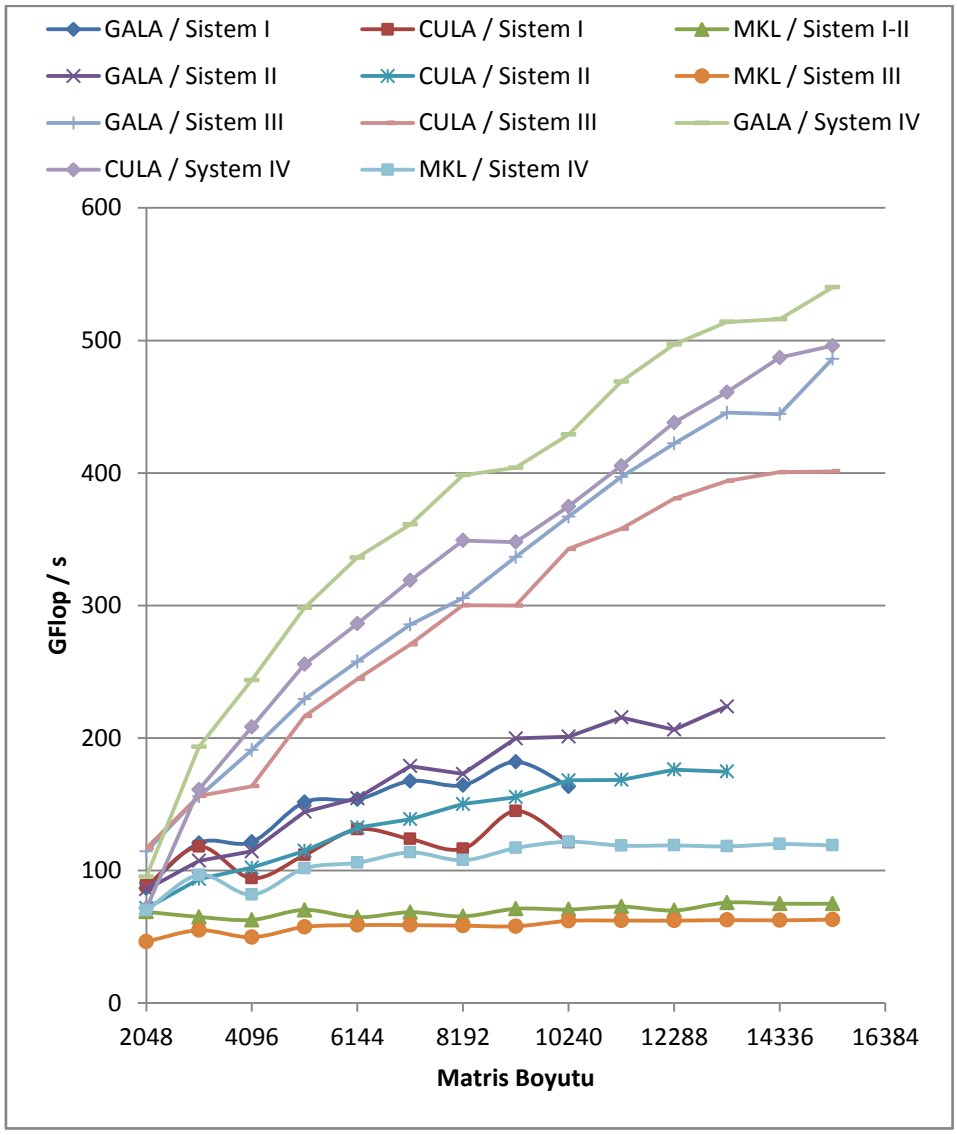


Şekil 5.23: Eniyilemeler sonucu elde edilen başarım artışı

Şekil 5.23'de göze çarpan bir diğer olguysa algoritmanın hızındaki dalgalanmalardır. Bu dalgalanmaların iki temel nedeni bulunmaktadır. Donanım içerisinde veriler aktarılırken belli boyutlardaki paketler halinde aktarılmaktadır. İletilecek olan verilerin boyutlarının bu paket boyutunun katı olmaması durumunda başarım düşmektedir. Bu durumun diğer bir nedeni ise iş parçacıklarının 32'li işlemci grupları halinde işletilmesidir. Eğer, işlem grubunu tam kapasite kullanabilecek boyutta veriler işlenecek olursa, algoritmanın başarımı artmaktadır.

Şekil 5.24'de MKL, CULA ve GALA kütüphanelerinin dört farklı sistemde elde edilen işlem hızları yer almaktadır. 9216 boyutundaki matris Sistem I kullanılarak ayrıştırıldığında MKL, CULA ve GALA algoritmalarının çözüm süreleri sırasıyla 3.84, 1.80 ve 1.40 saniyedir. Bu testte elde edilen en yüksek işlem hızları ise 71, 145 ve 182 GFlop/s olarak ölçülmüştür. Bu test için GALA algoritması, MKL ile karşılaştırıldığında 2.56 kat, CULA ile karşılaştırıldığındaysa 1.26 kat daha hızlı çözüm üretmiştir.

Sistem 2'de yapılan sınamalarda elde edilen en yüksek işlem hızı 224 GFlop/s iken, MKL ve CULA' ya nazaran hızlanma sırasıyla 2.95 ve 1.28 olmuştur. Son olarak, Sistem 3'te yapılan testlerde GALA algoritmasına ait en yüksek işlem hızı 486 GFlop/s olarak ölçülmüş ve bu sonucun MKL ve CULA ile karşılaştırılması sonucu sırasıyla 7.71 ve 1.21 kat hızlanma gözlenmiştir. Hızlanmadaki bu sıçrama Sistem 3'te diğer sistemlere nazaran daha yavaş bir MİB ve çok daha kuvvetli bir grafik işlem birimi yer almasının sonucudur.



Şekil 5.24: Tek duyarlıklı test sonuçları

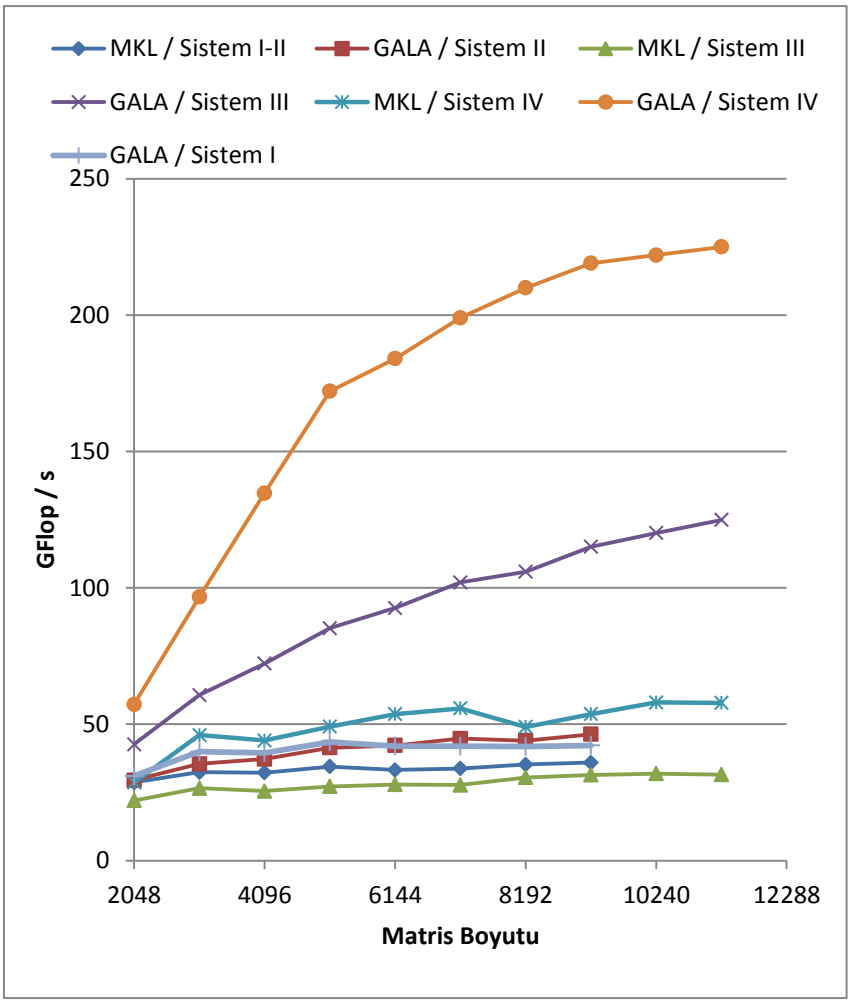
Sistem 4'te NVIDIA firması tarafından son kullanıcıların grafik işlem ihtiyaçlarından öte yüksek başarımlı hesaplama araştırmacıları için özel hazırlanmış olan Tesla kod adlı GAGİB yer almaktadır. Üretici bu ürününde araştırmacılara daha yüksek başarımla çift duyarlıklı işlem yapabilme fırsatının yanında daha yüksek hafıza ve daha hızlı donanım içi veri alış-verişi imkanı sunmuştur. Ayrıca bu sistemde yer alan MİB' de diğer sistemlerdekilere göre işlem kapasitesi ve veri alış-veriş hızı daha yüksek bir işlemcidir.

Bu sistemde yapılan sınamalarda, geliştirilmiş olan algoritma 540 GFlop/s işlem hızına ulaşırken CULA'ya göre 1.09 kat hızlanma elde etmiştir. Tesla GAGİB'ler de üretici tarafından donanımsal olarak hafıza gecikmeleri eniyilendiği için bu çalışmada uygulanmış olan eniyilemeler nispeten etkisini yitirmiştir. Sadece MİB kullanan MKL'ye göre ise 4.54 kat hızlanma elde edilebilmiştir. Bu sistemde yer alan MİB'in işlem hızı ve kapasitesi düşünüldüğünde bu düşüş olağan karşılanabilir.

Çift Duyarlıklı Test Sonuçlarının İrdelenmesi

Çift duyarlıklı hassasiyet kullanılarak yapılan çözümlerde ticari bir ürün olan CULA kullanılamamıştır. Bu nedenle geliştirilen algoritma yalnız MKL ile karşılaştırılmıştır.

Şekil 5.25'de gerçekleştirilmiş olan çift duyarlık hassasiyetli test sonuçları yer almaktadır. Sistem 1 ve 2' de MKL'nin çift duyarlıklı işlemlerde tek duyarlıklı işlemlere göre yarı yarıya yavaşladığı görülmektedir. Buna karşın, GALA, 3 ila 4.3 kat yavaşlamıştır. Ancak, geliştirilmiş olan karma algoritma hala, MKL'den ortalama olarak 1.30 kat daha iyi bir başarım elde etmiştir. Kendi serisinin en iyileştirilmiş GİB'lerinden birine sahip olan Sistem 3'te 125 GFlop/s işlem hızı ölçülürken, başarımı donanımsal olarak arttırılmış Tesla kod adlı GAGİB'in yer aldığı Sistem 4'te ise işlem hızı 225 GFlop/s olarak belirlenmiştir. Üreticinin çift duyarlık hassasiyetli işlemler için gerçekleştirmiş olduğu eniyilemeler düşünüldüğünde bu beklenen bir sonuçtur. Geliştirilmiş olan algoritma, Sistem 3'te MKL' ye göre 1.9 ila 3.7 kat daha yüksek başarım elde ederken, Sistem 4'te bu başarım artışı 2 ila 4.3 kat olarak ölçülmüştür.



Şekil 5.25: Çift duyarlıklı test sonuçları

Grafik Kartlar Üzerinde Çalışan Doğrusal Olmayan Dinamik Belirtik Çözümleme Algoritması

İkinci bölümde anlatılan ve MİB'ler (Central Processing Unit – Merkezi İşlem Birimi) üzerinde çalışan dinamik doğrusal olmayan belirtik algoritmanın sistem matrislerine gereksinim olmayışı ve eleman bazında çözüm özelliği sınırlı bellek kaynaklarına sahip ve paralelliği belli bir algoritmanın birçok veriye uygulanmasına dayanan grafik kartları için büyük avantajlardır. Bu avantajlardan ve grafik kartlarının sunduğu üstün performanstan yararlanabilmek için belirtik algoritma GİB'ler (Graphical Processing Unit – Grafiksel İşlem Birimi) üzerinde çalışacak şekilde değiştirilerek Panthalassa Çözümleme Platformuna eklenmiştir.

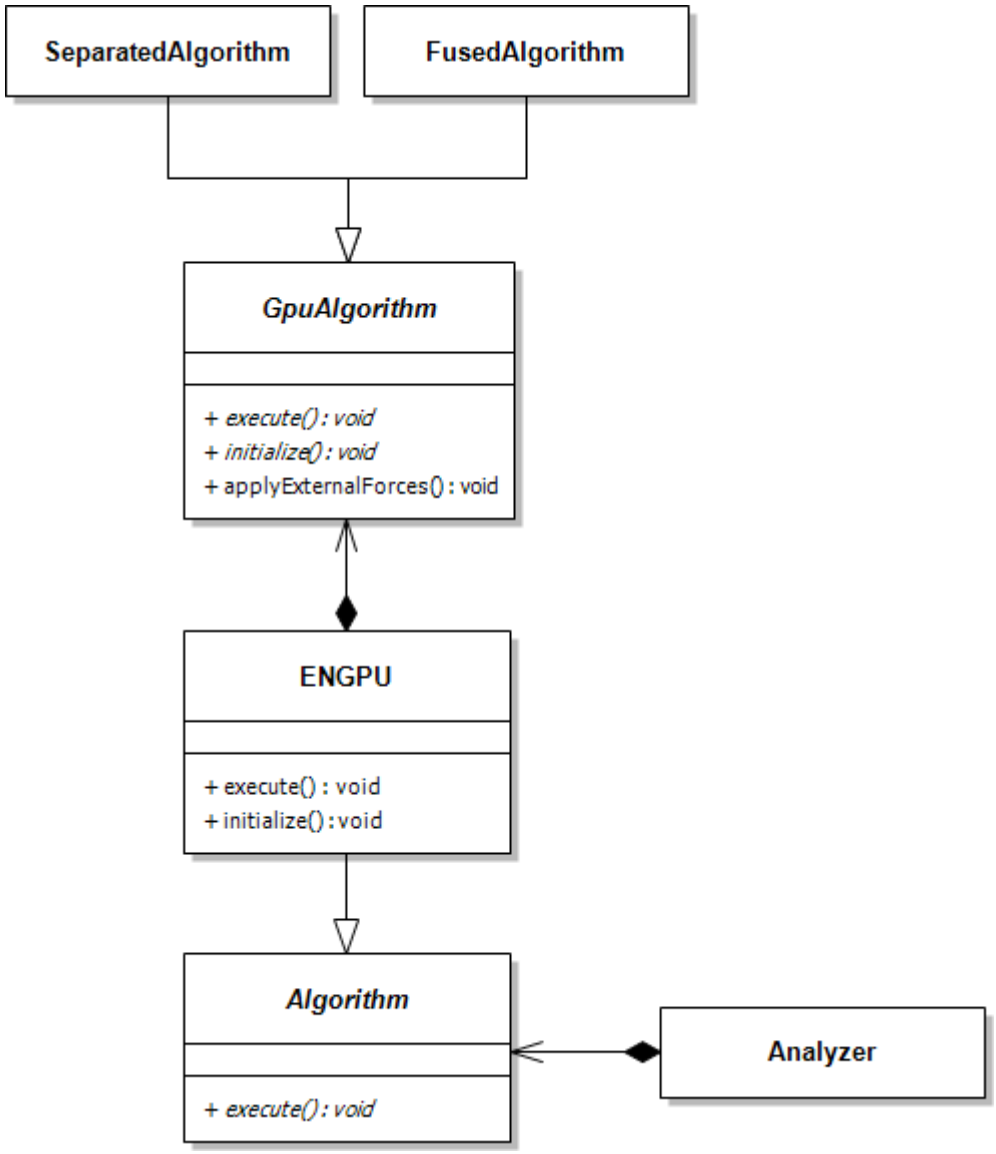
Doğrusal Olmayan Dinamik Belirtik Çözümleme Algoritmasının Uygulanması

Belirtik algoritmanın grafik kartlarını kullanan uygulaması *ENGPU* Sınıfıyla (Şekil 5.26) yapılmıştır. *ENGPU* Sınıfı *Algorithm* Sınıfından türer ve bu sınıfın *execute* fonksiyonunu yenileyerek algoritmayı uygular. *ENGPU* Sınıfı Panthalassa'dan sonlu elemanlar modelini alır, modeli grafik kartına gönderir, grafik kartını kullanarak çözümlemeyi yapar ve çözümleri önce işlemciye daha sonra da Panthalassa veri tabanına ulaştırır. Çözümleme işleminin iki farklı uygulaması *ENGPU* Sınıfına eklenmiştir: Ayrık ve kaynaşık uygulamalar. Ayrık uygulama *SeparatedAlgorithm* Sınıfı ile kaynaşık uygulama *FusedAlgorithm* Sınıfı ile uygulanmıştır. Bu iki uygulama sadece belirtik algoritmanın ana denkleminin (Denklem 2.43) çözüm kısmını içerirler, algoritma işleyişinin kalan kısımları *ENGPU* Sınıfında uygulanmıştır. Giriş dosyasında verilen bir seçenekle kullanıcı bu iki uygulamadan hangisinin çalışacağına karar verebilir.

ENGPU Sınıfı algoritmanın yürütülmesi işlemi için *execute* Fonksiyonun yanı sıra *initialize* isimli bir fonksiyonu da kullanır. *initialize* Fonksiyonu analizin başında bir kez çağrılır. Bu fonksiyon kullanıcının seçtiği belirtik çözümlüme uygulamasının (ayrık ya da kaynaşık) *initialize* Fonksiyonunu çağırır ve Panthalassa sonlu elemanlar modeli bilgisini grafik kartına gönderir. Bilgisayarın belleği ve grafik kartı arasındaki iletişim kilitli bellek sayfası (page locked) yöntemi kullanılarak uygulanmıştır. Bu yöntemde grafik kartıyla iletişimde kullanılacak bellek bölümleri önceden işletim sistemi ve grafik kartı sürücüsüne kaydedilir ve böylece grafik kartının aynı anda hem işlem yapması hem de bilgisayar belleği ile iletişimde bulunması sağlanır.

execute Fonksiyonu analizin her zaman adımında *Analyzer* objesi tarafından çağrılır. Bu fonksiyonda belirtik çözümleme yapılır ve çözümler grafik kartından Panthalassa veri tabanına aktarılır. Grafik kartında çözümleme her eleman için ayrı ayrı yapılır. Eleman çözümlerinin birleştirme işi ise bilgisayar işlemcisine aktarılmıştır. Bunun sebebi birleştirme işlemi sırasında ortaya çıkacak olan birçok rasgele bellek okuma işleminin grafik kartlarına uygun olmayışıdır.

ENGPU Sınıfı belirtik algoritmanın uygulamasını kullanıcının seçtiği ve *GpuAlgorithm* Sınıfından türeyen ayrık ve kaynaşık uygulamalardan biriyle yapar. *GpuAlgorithm* Sınıfının iki sanal fonksiyonu vardır: *initialize* ve *execute*. *initialize* Fonksiyonu analizin başında bir kez, *execute* Fonksiyonu ise her zaman adımında *ENGPU* nesnesi tarafından çağrılır. Sonlu elemanlar modeline gelen dış etkilerin (kuvvet, deprem vb.) çözümlemeye olan katkıları ise *GPUAlgorithm* uygulamasından bağımsız olarak *applyExternalForces* Fonksiyonunda yapılır. *applyExternalForces* Fonksiyonu gerekli GİB algoritmasını çağırarak bu işlemi gerçekleştirir. Bu işlemin grafik kartında yapılması işlemi direk olarak hızlandırmaz ancak işlemci grafik kartı arasında bilgi akışını gereksiz kılarak dolaylı bir hızlanma sağlar. Belirtik çözümleme işleminin yapıldığı *SeparatedAlgorithm* ve *FusedAlgorithm* ileriki iki bölümde açıklanmıştır.



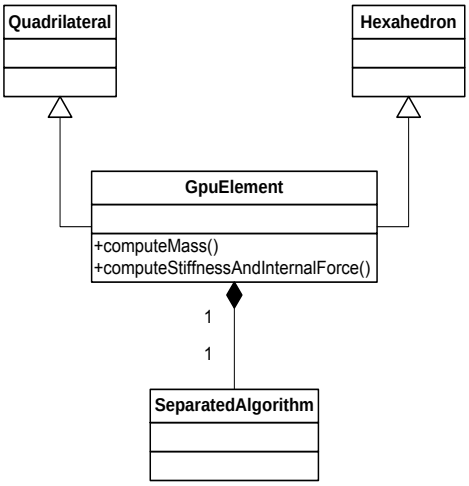
Şekil 5.26: ENGPU Sınıf Diyagramı

Ayrık Algoritma, *SeparatedAlgorithm* Sınıfı

CUDA programlama dilinde grafik kartında çalışan kodlar çekirdek (kernel) adı verilen özel fonksiyonlar içinde çalışırlar. Ayrık algoritma iki temel çekirdek içerir. Çekirdeklerden ilki Denklem 2.43'de kullanılan eleman matrislerini hesaplar ve grafik kartının beleğine kaydeder. İkinci çekirdek ise bu matrisleri kullanarak Denklem 2.43'ü çözer ve çözümleri her eleman için ayrı ayrı grafik kartının beleğine kaydeder. İki işlemin farklı çekirdeklere ayrılması analiz sırasında farklı elemanların kullanılabilmesini kolaylaştırır. Eleman matrislerini hesaplayan çekirdek farklı eleman tipleri için ayrı ayrı yazılır ve algoritmanın farklı elemanlar ile kullanılabilmesi sağlanır. Grafik kartında bir çekirdeğin başlatılmasının belli bir süre alır ancak ayrı çekirdeklerin kullanılması çekirdeklerin boyutlarını küçülttüğünden bazı durumlarda grafik kartının kısıtlı kaynakları daha verimli kullanılabilir.

Şekil 5.27 ayrık uygulamanın sınıf diyagramını gösterir. *GpuElement* sınıfı ile eleman matrislerini hesaplayan çekirdeklerin farklı eleman tipleri için farklı şekillerde yazılması sağlanmıştır. *GpuElement*

Sınıfının iki sanal fonksiyonu vardır: *computeMass* ve *computeStiffnessAndInternalForce*. *computeMass* ilgili eleman tipi için kütle matrisini hesap edip grafik kartı beleğine yazan çekirdeği çalıştırır. *computeStiffnessAndInternalForce* Fonksiyonu ise direngenlik ve iç kuvvetler matrislerini hesaplayan çekirdeği çalıştırır. Panthalassa'ya belirtik algoritmanın grafik kartlarında çözümü için *GpuElement* Sınıfından türeyen *Quadrilateral* Sınıfı dörtgen membran elemanı ve *Hexahedron* Sınıfı ile dörtgen prizma elemanı eklenmiştir.



Şekil 5.27: Ayrık Uygulama Sınıf Diyagramı

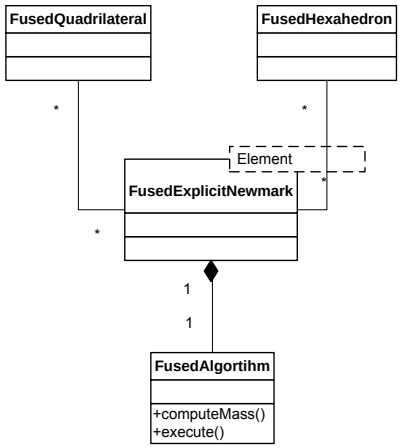
SeparatedAlgorithm Sınıfının *intialize* Fonksiyonunda modeldeki her eleman tipi için *computeMass* fonksiyonu çağrılarak kütle matrisleri hesaplanır. Bu matrisler analiz boyunca değişmeden kullanılırlar. *execute* Fonksiyonunda ise *computeStiffnessAndInternalForce* Fonksiyonu ile her zaman adımında değişen direngenlik ve iç kuvvet matrislerini hesaplanır. Daha sonra Denklem 2.43'ü çözen çekirdek kullanılarak belirtik çözüm gerçekleştirilir. Grafik kartında çekirdekler birçok izlek (thread) kullanılarak başlatılır. Her izlek farklı bir veri bölgesinde çalışır ve paralellik sağlanmış olur. Bu uygulamada her çekirdek tek bir sonlu eleman üzerinde çalışacak şekilde ayarlanmıştır. Eleman matrislerinin hesaplanması çok sayıda değişken gerektirdiğinden tüm değişkenler grafik kartının yazmaçlarında tutulamaz. Bu yüzden bilgilerin yazmaçlara göre çok daha yavaş olan grafik kartının belleğinden okunması zorunlu hale gelmiştir. Performans kaybının önüne geçmek için çekirdekler grafik kartı ön belleğini daha performanslı olarak kullanılmasının sağlayan *cudaFuncCachePreferL1* seçeneği ile başlatılmışlardır. Yine ön belleğin etkili olarak kullanılabilmesi için kullanılan veri yapıları çalışan tüm izleklerin ulaştığı bölgeleri yan yana getirecek şekilde düzenlenmiştir. Şekil 5.28 bu şekilde tasarlanmış bir veri yapısının örneği sunar. Şekilde gösterilen veri yapısı modeldeki tüm sonlu elemanlar için deplasman vektörlerini tutan bir veri yapısıdır. Veri yapısı ilk olarak her vektörün ilk elemanını daha sonra yine sırayla 2. 3. ve diğer elemanlarını tutması için tasarlanmıştır. Bu şekilde çalışan her izlek kendilerine düşen eleman için vektörün ilk elemanını bellekten istediğinde veriler topluca alınabilecek ve performans artışı sağlanacaktır. Ayrıca eleman matrislerinin simetrik özelliği de veri yapılarında kullanılmış ve matrislerin sadece yukarı üçgen kısmı bellekte tutulmuştur.

0	0	0	0	...	1	1	1	1	...

Şekil 5.28: Grafik Kartlarında Ön Bellek Performansını Arttıran Veri Yapısı Örneği

Çakışık Algoritma, *FusedAlgorithm* Sınıfı

Çakışık algoritmanın uygulaması ayrık uygulamanın hemen hemen aynısıdır (Şekil 5.29). Çakışık uygulamanın ayrıktan farkı eleman matrisleri ve Denklem 2.43'ün çözümü için gereken farklı çekirdeklerin meta programlama tekniğiyle tek bir çekirdekte birleştirilmesidir. Bu şekilde birden fazla çekirdeğin bağlatılması ile doğan performans kayıplarınınım önüne geçilir. Çakışık uygulamanın dezavantajı ise birleşen çekirdeklerin kısıtlı grafik kartı kaynaklarını fazlaca kullanıp grafik kartını yavaşlatma ihtimalidir.

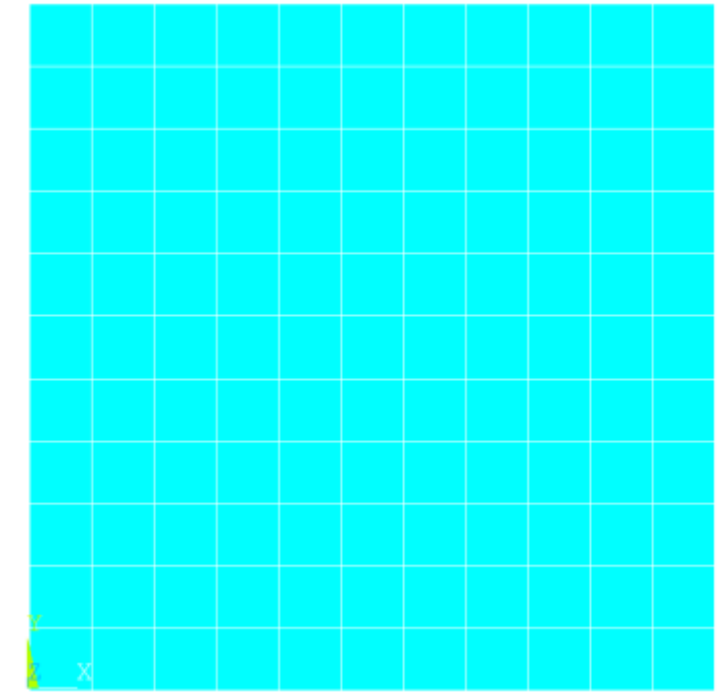


Şekil 5.29: Çakışık Algoritma Sınıf Diyagramı

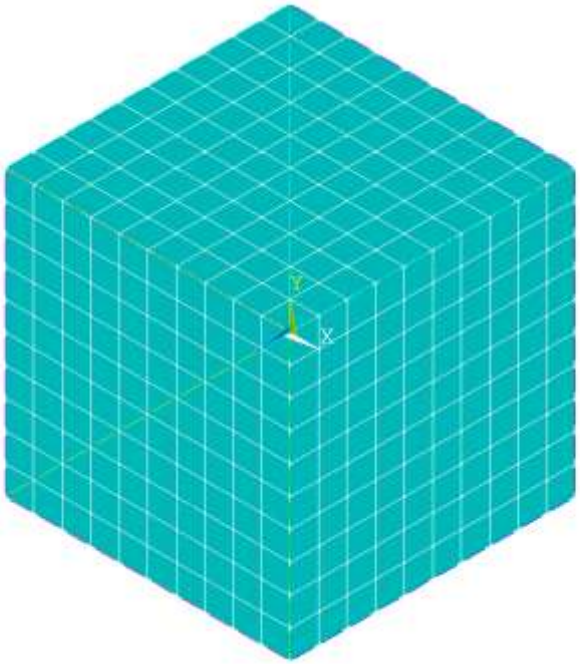
FusedAlgorithm dörtgen membran ve prizma elemanlarını *FusedQuadrilateral* ve *FusedHexahedron* sınıfları ile uygular. *FusedExplicitNewmark* Sınıfı parametrik bir sınıftır. Sınıfa *FusedQuadrilateral* ve *FusedHexahedron* sınıfları derleme zamanında parametre olarak verilir. *FusedExplicitNewmark* Sınıfı bu sınıfları kullanarak tüm belirtik çözümlemeyi gerçekleştirir. Derleyici kullanılan fonksiyonları ve *FusedExplicitNewmark* Sınıfının çekirdeğini iç içe geçirir ve tek bir çekirdek üretir.

Doğrusal Olmayan Dinamik Belirtik Çözüm Algoritmasının Doğrulanması

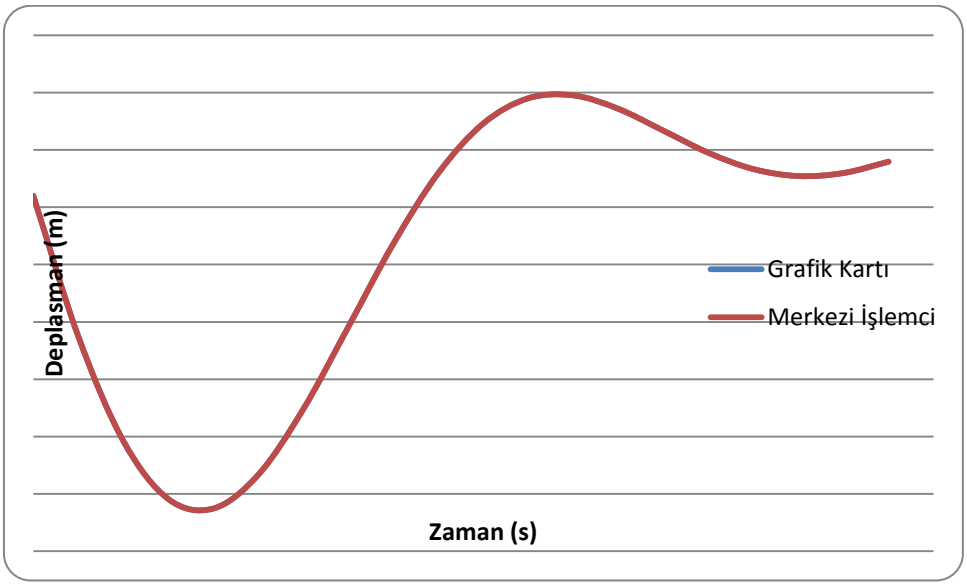
Grafik kartı üzerinde çalışan belirtik algoritmanın doğrulaması için dörtgen membran ve dörtgen prizma elemanlarından oluşan iki model (Şekil 5.30 ve Şekil 5.31) hem merkezi işlemci üzerinde çalışan hem de grafik kartlar üzerinde çalışan belirtik algoritmalar (ayrık ve çakışık) ile çözülmüştür. Modellere en üst sol köşelerinden anlık bir kuvvet uygulanmış ve bir saniye boyunca yükün uygulandığı düğüm noktasının deplasmanları kaydedilmiştir.



Şekil 5.30: Belirtik Algoritma Doğrulaması: Dörtgen Membran Model

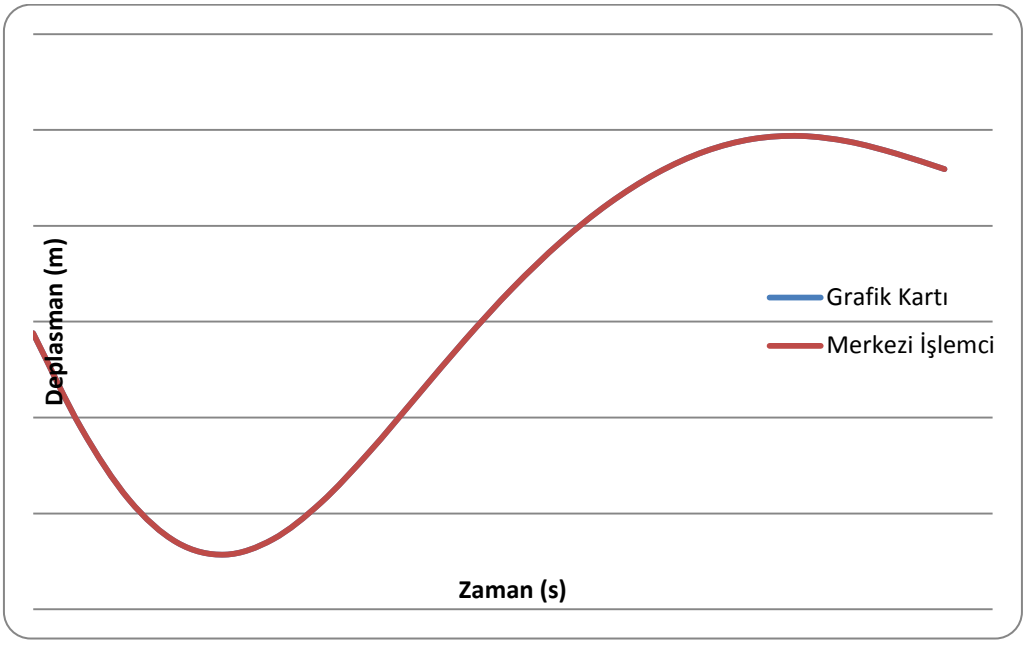


Şekil 5.31: Belirtik Algoritma Doğrulaması: Küp Modeli



Şekil 5.32: Belirtik Algoritma Doğrulaması Dörtgen Membran Model

Kullanılan iki algoritma da tamamen aynı oldukları için modellerden elde sonuçlar aynı olmalıdır. Şekil 5.32 ve Şekil 5.33'de modellerden alınan sonuçlar gösterilmiştir. Beklenildiği gibi sonuçlar iki algoritma için de tamamen aynıdır.



Şekil 5.33: Belirtik Algoritma Doğrulaması Dörtgen Prizma Model

Doğrusal Olmayan Dinamik Belirtik Çözüm Algoritmasının Başarım Testleri

Başarım testlerinde ayrık ve kaynaşık belirtik çözümleme uygulamaları, dördüncü bölümde anlatılan çok işlemcili sürümle değişik büyüklükte iki ve üç boyutlu modeller (Tablo 5.4) üzerinde karşılaştırılmıştır. Her model büyüklüğü için iki ve üç boyutlu modeller doğrusal olmayan dörtgen membran ve prizma elemanları kullanılarak hazırlanmıştır. İki boyutlu modeller dikdörtgen ve üç boyutlu modeller küp şeklindedir. Tüm modeller Loma Prieta Depreminin Treasure Adası kayıtları kullanılarak yüklenmiş ve analiz zamanları gözlenmiştir. Performans karşılaştırılması için kısa sureli analizler yeterli olduğu için iki boyutlu modeller 100 zaman adımı, üç boyutlu modeller 20 zaman adımı boyunca analiz edilmiştir. Grafik kartı uygulamaları Nvidia Tesla C2050 grafik kartı üzerinde test edilmiştir. MİB uygulaması ise yeni SuperMicro bilgisayar kümesi üzerinde denenmiştir. Yeni bilgisayar kümesi altı bilgisayardan kurulu olup, her bilgisayarda iki tane dört çekirdekli Intel Xeon işlemci ve 24GB RAM vardır. Bilgisayarlar 40 Gbit hızında Infiniband teknolojisiyle birbirlerine bağlanmıştır

Tablo 5.4: Testlerde Kullanılan Model Büyüklükleri

Model	Eleman Adedi
Model 1	10,000
Model 2	20,000
Model 3	40,000
Model 4	100,000
Model 5	250,000
Model 6	500,000
Model 7	1,000,000

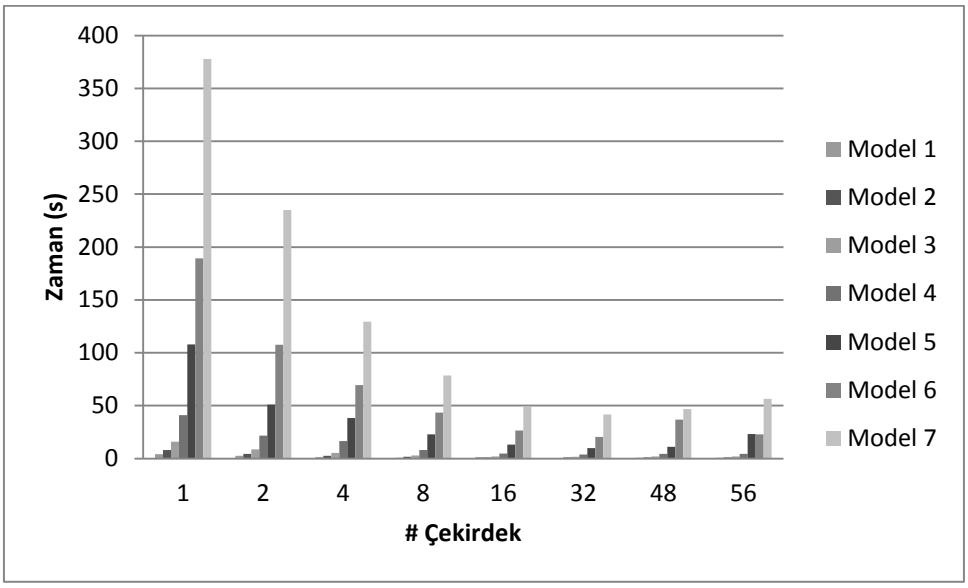
İki Boyutlu Modeller

Şekil 5.34'de CPU temelli uygulama için analiz sürelerini gösterilmiştir. Çözüm süreleri 48 çekirdeğin kullanıldığı teste kadar artan çekirdek sayısıyla azalmıştır. 56 çekirdek kullanıldığında oluşan performans kaybının sebebi iki boyutlu elemanların iletişim süresini karşılayacak kadar hesaplama içermemesidir.

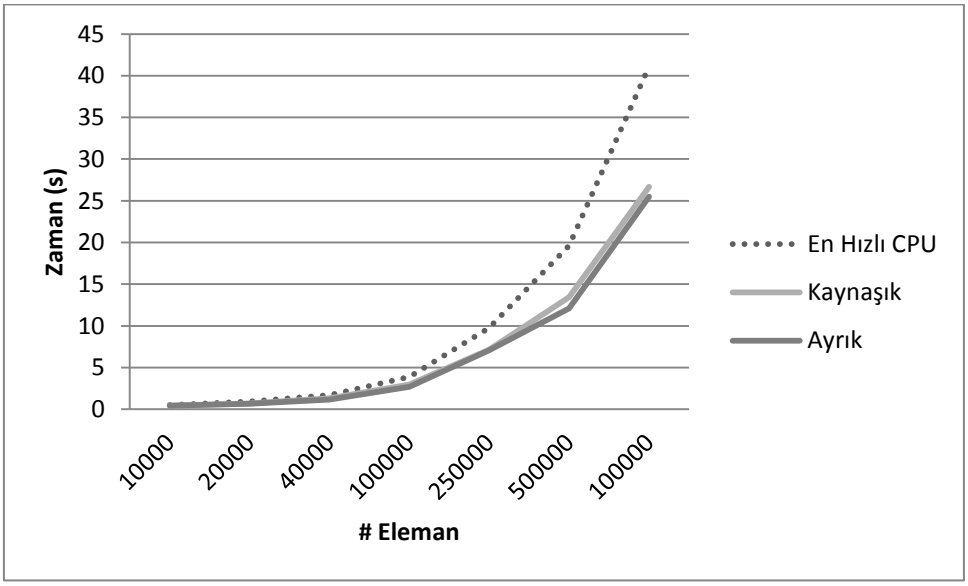
Şekil 5.35'de ise MİB uygulamasından elde edilen en hızlı sürelerin grafik kartı uygulamalarının analiz süreleri ile karşılaştırılması sunulmuştur. Hem ayrık hem de kaynaşık uygulamalar MİB uygulamasından daha kısa analiz sürelerine ulaşmıştır. Modeldeki eleman sayısı arttıkça grafik kartı uygulamaları MİB uygulamasına kıyasla artan bir performansa sahiptirler. Bunun sebebi artan eleman sayısının MİB uygulamasının ihtiyaç duyduğu iletişimi de arttırmasıdır. Ayrık uygulama kaynaşık uygulamaya göre az da olsa daha hızlıdır.

Üç Boyutlu Modeller

Şekil 5.36'da MİB uygulamasının üç boyutlu modeller üzerindeki performansı gösteriliştir. Kullanılan çeikirdek sayısı arttıkça analiz süresi kısalmıştır. Prizma elemanlar yeterli hesaplama içerdikleri için iki boyutlu elemanlarda yaşanan performans kaybı üç boyutlu modellerde yaşanmamıştır.

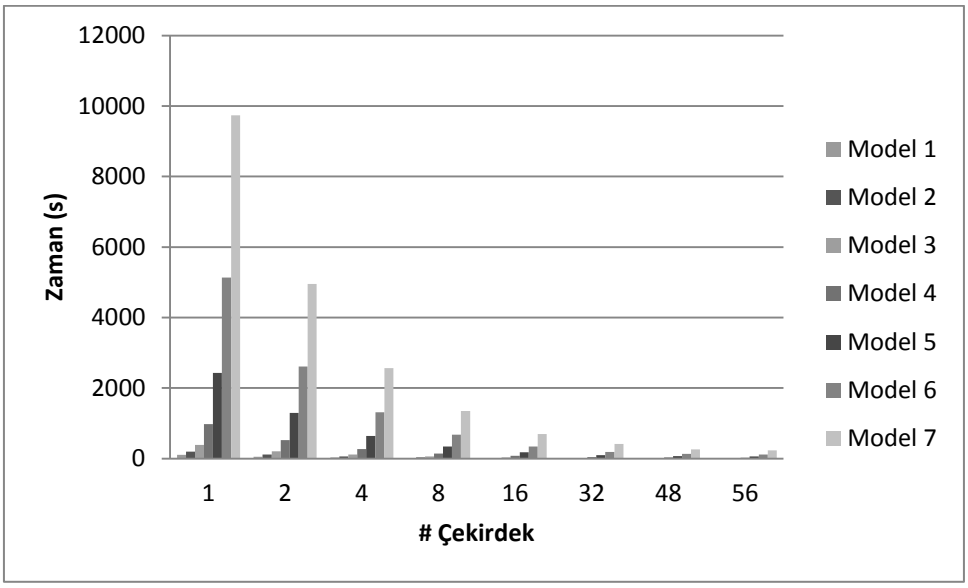


Şekil 5.34: İki Boyutlu Modellerde MİB Temelli Uygulama İçin Zamanlamalar

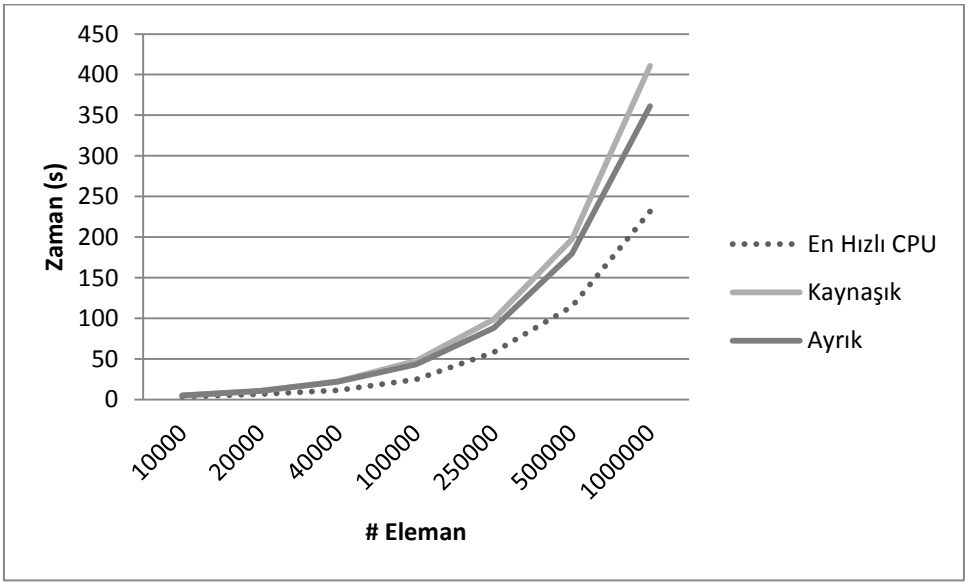


Şekil 5.35: İki Boyutlu Modellerde MİB ve GİB Temelli Uygulamaların Zamanlama Karşılaştırmaları

Şekil 5.37 değişik modeller için en hızlı MİB uygulamasının grafik kartı uygulaması ile karşılaştırılmasını gösterir. İki boyutlu modellerde MİB uygulaması grafik kartı uygulamalarına göre daha hızlıdır. İki boyutlu modeller üzerinde yapılan testlerdekine benzer olarak ayrık uygulama kaynaşık uygulamadan biraz daha hızlıdır. Grafik kartı uygulamaları MİB uygulamasının 32 çekirdek kullanarak ulaştığı performansa benzer bir performansa ulaşmışlardır.



Şekil 5.36: Üç Boyutlu Modellerde MİB Temelli Uygulama İçin Zamanlamalar



Şekil 5.37: İki Boyutlu Modellerde MİB ve GİB Temelli Uygulamaların Zamanlama Karşılaştırmaları

Bölüm 6

Sonuçlar, Değerlendirmeler

Proje bünyesinde paralel sistemlerde yapı mühendisliğinde sıkça kullanılan sonlu elemanlar modellerinin çözümlenmesi için bir yazılım platformu geliştirilmiştir. Platform bünyesinde zengin bir sonlu eleman kütüphanesi, genişletilmeye uygun bir malzeme modeli kütüphanesi ve seri ve paralel sistemlerde çalışabilen çeşitli çözümleme algoritmalarını barındırmaktadır. Nesne yönelimli veri yapısına sahip çözümleme motoru, "plug-in" teknolojisinin yardımıyla dışarıdan yapılacak eklemelerle genişletilebilecek bir şekilde tasarlanmıştır. Bu sayede yazılım platformunun yazılım koduna ihtiyaç duyulmadan farklı amaçlar için kullanılması ve zenginleştirilmesi mümkün hale gelmiştir. Bütün bu çalışmalara ek olarak yeni nesil ekran kartlarının yapı mühendisliği problemlerinde kullanılabilirliği sınanmış ve özellikle belirtik tümleştirme yöntemi kullanılarak gerçekleştirilen zaman alanında dinamik çözümlemelerde çok ciddi kazanımlar elde edilmiştir. Proje bünyesinde gerçekleştirilen çalışmalar sonucunda aşağıda özetlenen bulgular elde edilmiştir.

- Sonlu elemanlar yöntemi yaklaşık bir yöntem olup elde edilen sonucun analitik sonuca yakınlığı kullanılan eleman formülasyonuna ve geometrisine bağlıdır. İkinci bölümde detayları verilen çok çeşitli sonlu eleman geliştirilmiş ve çeşitli doğrulama problemleri çözülmüştür. Doğrulama problemleri sonucunda her bir elemanın iyi sonuç verdiği problem tipi olduğu ve eleman geometrilerindeki bozulmanın sonuçlarda ciddi hatalara yol açtığı görülmüştür. Bu sebepten dolayı, yapı mühendisliği problemleri için özel geliştirilen bir yazılım olsa bile bu tip bir çözümleme motoru mutlaka farklı tipte sonlu eleman formülasyonlarını bünyesinde barındırmalıdır.
- Yazılım platformunun motoru nesne yönelimli veri yapısına sahiptir. Ayrıca "plug-in" teknolojisiyle dışarıdan eklenen programları da platform bünyesinde kullanmak mümkündür. Platformun genişletilebilir bir yapıya sahip olmasını sağlamak, diğer bir deyişle, dışarıdan farklı eleman ve malzeme modellerinin eklenmesinin yanı sıra çözümleme algoritmalarının da eklenip kullanılmasını sağlamak asıl hedeftir. Bu amaçla ilk olarak model veri tabanını kuran algoritmalarla, çözümleme algoritmaları birbirlerinden ayrılmıştır. Bu sayede çözümleme algoritmaları model veri tabanıyla, veri tabanının nasıl oluştuğuyla ilgilenmeden veri tabanıyla direk olarak çalışmaktadır. Çözümleme algoritmaları için geliştirilen ve *Analyzer* sınıfının bir parçası olan *Algorithm* sınıfı sayesinde dışarıdan eklenen algoritmalar model veri tabanıyla birebir ilişkiye girebilmekte ve *Timetable, Partitioning Job* gibi yardımcı sınıflarla da çalışabilmektedir. Bu durum dışarıdan farklı veri yapılarını okuyan algoritmaların eklenmesine olanak vermiştir. Benzer şekilde çözümleme motorundan dışarıya (bir dosyaya veya başka bir programa) veri aktarımı için *Tracker* sınıfı yaratılmıştır.
- Platform çözümleme motoru paralel hesaplama için kullanılabilecek yapıları da içermektedir. Alt yapı tabanlı çözücüler gibi modelin bölümlenmesine ihtiyaç duyan algoritmalar için "*PatritioningJob*" sınıfı yaratılmış ve farklı grafiksel gösterimlerin ve bölümleme algoritmalarının eklenmesi

sağlanmıştır. Bu bağlamda yapının noktasal ve eleman grafiksel gösterimlerini içeren sınıflar ve METIS ve PARMETIS isimli iki bölümleme kütüphanesi de dışarıdan platforma eklenmiştir.

- Platform motorunun genişletilebilirliği ısı iletimi problemlerinin çözümlenmesiyle ilgili elemanların ve algoritmaların dışarıdan eklenmesiyle sınanmıştır. Bu sayede her bir düğüm noktasında tek bir serbestlik derecesinin bulunması, hem veri girişinde hem de hesaplama aşamasında farklı sınır koşullarının tanımı, sistem matrislerinin tümlenmesi aşamasında motordaki genel algoritmaların kullanımı ve çıktıların hazırlanması sırasında kullanılan algoritmalar sınanmıştır. Sonuç olarak, platform motoru seri ısı iletimi problemlerinin çözümlenmesi için de başarıyla kullanılabilmiştir.
- Doğrusal statik, doğrusal olmayan statik ve örtük tümleştirme yöntemleriyle gerçekleştirilen çözümlemelerin ortak yanı hepsinin sonuçta bir doğrusal denklem sisteminin çözülmesine ihtiyaç duymalarıdır. Bu sebepten dolayı ilk olarak farklı paralel denklem çözücüler platforma eklenmiş ve başarımları farklı yapısal modellerle ve paralel sistemlerde sınanmıştır. Toptan çözüm yöntemi olarak adlandırılan ilk çözücü literatürde sıkça adı geçen ve aşamalı çoklu sınır yönteminin kullanan MUMPS'ın bir uygulamasıdır. İkinci yöntem ise alt-yapı tabanlı bir çözücü olup veri hazırlama aşamasından sınır denklem çözümü aşamasına kadar her aşaması en iyileştirilmiştir. Gerçekleştirilen karşılaştırmalı çalışma sonucunda toptan çözücülerin başarımının genelde alt-yapı tabanlı çözücülerden iyi olduğu, alt-yapı tabanlı çözücülerin ise model geometrisi karmaşıklaştıkça başarımlarının toptan çözücülere göre iyileştiği görülmüştür. Alt-yapı tabanlı çözücülerin başarımını düşüren en önemli etken büyük boyutlu sınır denklem sisteminin çözümü sırasındaki yoğun veri alış verişidir.
- Paralel toptan ve alt-yapı tabanlı çözücülerin başarımları, örtük tümleştirme yöntemleri kullanılarak gerçekleştirilen zaman alanında doğrusal çözümleme için de karşılaştırılmıştır. Matris sisteminin sadece bir defa ayrıştırılmasına ama yük vektörlerinin çok sefer çözülmesine ihtiyaç duyan bu çözümlemede de benzer sonuçlar elde edilmiştir. Düzgün geometrili bir modelde her iki yöntem de birbirlerine yakın sonuçlar vermiş ancak bilgisayar sayısı arttıkça toptan çözücülerin çözüm hızı daha çok azalmıştır. Sonuç olarak, gerek dağıtık, gerekse paylaşık bellekli sistemlerde toptan çözücüler diğer alternatiflere göre daha yüksek paralel verimlilikte çalışmaktadır.
- Dinamik çözümleme için iki tip tümleştirme yöntemi kullanılmaktadır; örtük ve belirtik. Hesaplama bakış açısına göre örtük yöntemde sistem matrislerinin oluşturulup ayrıştırılması gerekmektedir. Belirtik yöntemde ise bütün hesaplamalar eleman düzeyinde gerçekleştirilip, sistem düzeyinde sadece vektörlerin oluşturulması yeterlidir. Bu durum birazdan detaylarına değinileceği gibi paralel hesaplama açısından çeşitli avantajları da beraberinde getirmektedir. Proje bünyesinde örtük ve belirtik algoritmalar karşılaştırılmış ve şu sonuçlara ulaşılmıştır:
  - Belirtik algoritmaların örtük algoritmalara göre paralel verimliliği daha yüksektir. Diğer bir deyişle belirli bir işlemci sayısıyla elde edilen hızlanma daha yüksektir. Örnek doğrusal modelde 32 işlemci kullanıldığında örtük algoritmada 10.4, belirtik algoritmada ise 13.65 kat hızlanma elde edilmiştir. Doğrusal olmayan modelde belirtik algoritma 48 işlemciyle 40.3 kat, örtük algoritma ise 16.2 kat hızlanmıştır.
  - Yine karşılaştırma için çözülen örnek modelde belirtik algoritmayla tek bir zaman adımı aynı paralel sistemde daha hızlı çözülmüştür.
  - Örtük algoritmanın bellek ihtiyacı daha yüksektir. Örtük algoritmayla bellek yetersizliği nedeniyle çözülemeyen boyutlardaki modeller belirtik algoritmayla çok rahat çözülmüştür. Belirtik algoritmayla toplam 144 GB bellekli bir sistemde 48 milyon denklem çözülebilmiştir.
  - Diğer taraftan örtük algoritmayla elde edilen sonuçları belirtik algoritmayla elde edebilmek için daha küçük zaman adımlarına ihtiyaç duyulmaktadır. Örnek problemde aynı sonuçların elde edildiği zaman adımları kullanıldığında (örtük için 0.02 saniye, belirtik için 0.0005 saniye) örtük algoritmanın toplam çözümleme süresinin daha kısa olduğu görülmüştür.

- Yeni nesil ekran kartları kullanılarak iki farklı seyrek matris çözücü geliştirilmiştir; çoklu sınır ve aşamalı çoklu sınır. Farklı grafik işlemciler kullanılarak gerçekleştirilen örnek model çözümlerinde çoklu sınır çözücülerinin aşamalı çoklu sınır çözücülerinden daha iyi başarım gösterdiği ancak yüksek bellek kullanımı nedeniyle büyük matrisleri çözemediği görülmüştür. Ancak GİB çözücülerinden elde edilen çözüm süreleri paralel MİB çözücülerinden oldukça yavaştır. Bu sebepten dolayı sadece GİB kullanarak doğrudan seyrek matris çözümü yapmak verimli bir yaklaşım değildir.
- MİB-GİB karma ortamlar için yoğun matris indirgeme ve çözme algoritmaları geliştirilmiştir. MİB - GİB karma kullanıldığı durumlarda sadece MİB için geliştirilen çözücülerden çok daha hızlı sonuçlar elde edilmiştir. Ayrıca çeşitli en iyileştirmeler sayesinde GİB için geliştirilen standart yoğun matris çözücülerinden daha hızlı sonuçlar elde edilmiştir.

- GİB'lerin paralel belirtik çözümlemeler için çok uygun donanımlar olduğu görülmüştür. GİB'nin bellek sınırları içindeki modellerde iki ve üç boyutlu modeller çözümlenmiş ve özellikle üç boyutlu modellerde 48 çekirdekli MİB çözümlerine göre ciddi hızlanmalar elde edilmiştir. Bu sonuç GİB'lerin MİB'lerinin önemli bir alternatifi olduğunu göstermektedir.

Kaynakça

ABAQUS, Simulia, www.simulia.com

Allmann, D.J., ”A Compatible Triangular Element Including Vertex Rotations for Plane Elasticity Analysis”, Computers&Structures, Vol.19, , pp.1-8 (1984)

Amestoy P.R., Duff I. S., and L’Excellent J.-Y. “Multifrontal Parallel Distributed Symmetric and Unsymmetric Solvers”, Computer Methods in Applied Mechanics and Engineering, 184 501-520 (2000)

Amestoy P. R., Davis T. A., and Duff I. S., An approximate minimum degree ordering algorithm, SIAM Journal on Matrix Analysis and Applications 17(4), pp. 886_905, (1996)

Amestoy P. R., Duff I. S., and L'Excellent J.-Y., Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods Appl. Mech. Eng. 184, pp. 501_520, (2000).

Amestoy P. R., I. S. Duff, L'excellent J.-Y., and Li X. S., Analysis and comparison of two general sparse solvers for distributed memory computers, ACM Trans. Math. Softw. 27(4), pp. 388_421, (2001)

Amestoy P. R., Duff I. S., and Koster J., Mumps: A general purpose distributed memory sparse solver,_ in In Proc. PARA2000, 5th International Workshop on Applied Parallel Computing, pp. 122_131, Springer-Verlag, (2000)

ANSYS, ANSYS Inc., www.ansys.com

Ashcraft C., Eisenstat S., and Liu J.,. “A Fan-in Algorithm for Distributed Sparse Numerical Factorization”, SIAM Journal on Scientific and Statistical Computing, 11 593-599 (1990)

Ashcraft C., Grimes R.G.,. "SPOOLES: An Object-Oriented Sparse Matrix Library", Proceedings of 1999 SIAM Conference on Parallel Processing for Scientific Computing, March 22-27. (available at http://www.netlib.org/linalg/spooles) (1999)

Ashcraft C., The Fan-both Family of Column-based Distributed Cholesky Factorization Algorithms, Graph Theory and Sparse Matrix Computation, The IMA Volumes in Mathematics and its Applications, Springer, Berlin, New York, 56, 159-190 (1993)

Bahcecioglu T., Ozmen S., Kurc O., A Comparative Study on Two Different Direct Parallel Solution Strategies for Large-Scale Problems, Proceedings of the First International Conference on Parallel, Distributed and Grid Computing for Engineering, Pecs, Hungary, (2009)

Batoz, J.L, Bathe, K.L. and Ho, L.W., “A Study of Three Node Triangular Plate Bending Elements”, Int. Jour. Num. Meth. Engng., Vol.15, pp. 1771-1812 (1980)

Batoz, J.L., and Tahar, M.B., “Evaluation of a New Quadrilateral Thin Plate Bending Element”, Int. Jour. Num. Meth. Engng., Vol.18, pp. 1655-1677 (1982)

Beaumont O., Boudet V., Petitet A., Rastello F., “A proposal for a heterogeneous cluster ScaLAPACK (Dense Linear Solvers)”, IEEE Transactions on Computers, 50 (10), 1052-1070, (2001)

Belytschko T., Liu W. K., Moran B., Nonlinear Finite Elements for Continua and Structures, Wiley & Sons, (2004)

Berger M.J., Bokhari S.H., A Partitioning Strategy for Non-uniform Problems on Multiprocessors, IEEE Trans. Computers, C 36, 570-580 (1987)

Berglund G. Z. M. and Leeuw S. W. de, A study into the feasibility of using two parallel sparse direct solvers for the helmholtz equation on linux clusters: Research articles, Concurr. Comput. : Pract. Exper. 18(7), pp. 749_769, (2006)

Bitzarakis S., Papadrakakis M., Kotsopulos A., Parallel Solution Technique in Computational Structural Mechanics, Comput. Methods Appl. Mech. Engrg. 148 75-104 (1997)

Bitzarakis S., Papadrakakis M., and Kotsopulos A.,. “Parallel Solution Technique in Computational Structural Mechanics”, Computer Methods in Applied Mechanics and Engineering, 148 75-104 (1997)

Bolz, J., Farmer, I., Grinspun, E. and Schröder, P. Sparse matrix solvers on the GPU: conjugate gradients and multigrid. ACM Trans. Graph. 22(3) 917–924 (2003).

The Boost Graph Library (BGL) http://www.boost.org/doc/libs/1_51_0/libs/graph/doc/index.html

Buatois L., Caumon, G., Lévy B., Concurrent number cruncher - A GPU implementation of a general sparse linear solver International Journal of Parallel, Emergent and Distributed Systems, Vol 24(3):205-223 (2009)

Cao Z., Xu S., Xue W., Chen W., Improving Dense Linear Equation Solver on Hybrid CPU-GPU System, 10th Int. Sym. on Pervasive Systems, Algo., and Networks, 556-562 (2009)

Choi J., Dongarra J., Pozo R., Walker D. W., ScaLAPACK: A Scalable Linear Algebra Library for Distributed Memory Concurrent Computers**,** Technical Report: UT-CS-92-181, (1992)

Choi J., Dongarra J., Ostrouchov S., Petitet A., Walker D, Whaley R, The Design and Implementation of the ScaLAPACK LU,QR and Cholesky Factorization Routines

Cook R.D., Malkus D.S., Plesha M.E., Witt R.J., “Concepts and Applications of Finite Element Analysis 4th Edition sf. 410”, (2001)

Couturier, R., Domas, S. Sparse systems solving on GPUs with GMRES. Springer Science + Business Media, LLC (2011)

CUBLAS, The NVIDIA CUDA Basic Linear Algebra Subroutines, https://developer.nvidia.com/cublas, Ekim (2012)

CUDA Programming Guide, from CUDA Developer Zone, Mart, (2012)

CULA: GPU Accelerated Linear Algebra, http://www.culatools.com/ , Ekim (2012)

Damien T., Nvidia Fermi: the GPU Computing revolution, http://www.behardware.com/art/lire/772/

Dongarra J. J., Du L. S., Sorensen D. C., and Vorst H. A. V., Numerical Linear Algebra for High Performance Computers, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, (1998)

Dongarra, J. and Walker, D., LAPACK Working Note 58, The Design of Linear Algebra Libraries for High Performance Computers

Driessche R.V., Roose D., An Improved Spectral Bisection Algorithm and Its Application to Dynamic Load Balancing, Parallel Comput. 21, 29-48 (1995)

EduShake, EduPro Civil System Inc., www.proshake.com

Escaig Y., Touzot G., Vayssade M., Parallelization of a multilevel domain decomposition method, J. Comput. Systems Engrg. Vol.5 No.3 253-263 (1994)

Farhat C., Crivelli L., and Roux F. X.. "Extending Substructure Based Iterative Solvers to Multiple Load and Repeated Analyses", Computer Methods in Applied Mechanics and Engineering, 117 195-209 (1994)

Farhat C., Pierson K., Lesoinne M., The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems, Comput. Methods Appl. Mech. Engrg. 184, 333-374 (2000)

Farhat C., Chen P.S., Risler F., Rous F. X., A Unified Framework for Accelerating the Convergence of Iterative Substructuring Methods with Lagrange Multipliers, Int. J. Numer. Methods Engrg. 42 257-288 (1998)

Farhat C., A Simple and Efficient Automatic FEM Domain Decomposer, Comput. Struct. Vol. 28 No.5 579-602 (1988)

Fulton R. E., Su P. S., Parallel substructure approach for massively parallel computers, Comput. Engng. Vol. 2 75-82 (1992)

Garey M., Johnson D., Stockmeyer L., Some Simplified NP-complete Graph Problems, Theoretical Computer Science, 1, pp. 237-267 (1976)

Guermouche A., L'Excellent J.-Y., and Utard G., Impact of reordering on the memory of a multifrontal solver, Parallel Computing 29(9), pp. 1191_1218, (2003)

Heath M. and Raghavan P. "Performance of a Fully Parallel Sparse Solver", Proceedings of Scalable High Performance Computing Conference, IEEE Computer Society Press, 334-341 (1994)

Hendrickson B. "Load Balancing Fictions, Falsehoods and Fallacies", Applied Mathematical Modelling, 25 99-108 (2000)

Hendrickson B., and Kolda T.G.. “Graph Partitioning Models for Parallel Computing”, Parallel Computing, 26 1519-1534 (2000)

Hsieh S. H., Modak S., Sotelino E. D., Object-oriented parallel programming tools for structural engineering applications, Comput. Systems Engng., Vol. 6 No. 6 533-548 (1995)

Hughes T.J.R ve Liu W.-K., “Implicit-Explicit Finite Elements in Transient Analysis: Implementation and Numerical Examples”, ASME Journal of Applied Mechanics and Engineering, (1979)

Hsieh S.H., Paulino G.H., Abel J. F., Recursive Spectral Algorithms for Automatic Domain Partitioning in Parallel Finite Element Analysis, Comput. Methods Appl. Mech. Engrg. 121, 137-162 (1995)

Jung J.H., Cholesky decomposition and linear programming on a GPU, Technical Report, (2006)

Ibrahimbegovic, A., ”Quadrilateral Finite Elements for Analysis of Thick and Thin Plates”, Comp. Meth. App. Mech. Engng., Vol.110, pp.195-209 (1993)

Ibrahimbegovic, A., Taylor, R.L. and Wilson, E.L. “A Robust Quadrilateral Membrane Finite Element with Drilling Degrees of Freedom”, Int. Jour. Num. Meth. Engng., Vol.30, pp.445-457 (1990)

Irony D., Shklarski G., and Toledo S., Parallel and fully recursive multifrontal supernodal sparse cholesky,_ Future Generation Computer Systems 20, pp. 425_440, (2004)

Karypis G., Kumar V., METIS: a Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-reducing Orderings of Sparse Matrices, version 4.0, (1998)

Krüger, J. and Westermann, R. Linear algebra operators for gpu implementation of numerical algorithms. ACM Transactions on Graphics. (TOG) 22(3) 908–916 (2003).

Krysl P. and Belytschko T., Object-oriented Parallelization of Explicit Structural Dynamics with PVM, Computer&Structures, 66, 259-273 (1998)

Kurc O., Will K.M., A substructure based parallel solution framework for structural systems having multiple loading cases, Proceedings of the International Conference on Computing in Civil Engineering, ASCE (Cancun, 2005)

Kurc O., Will K.M., An Iterative Parallel Workload Balancing Framework for Direct Condensation of Substructures, Computer Methods in Applied Mechanics and Engineering, 196, 2084-2096 (2007)

Kurc O., A Workload Distribution Framework for the Parallel Solution of Large Structural Models on Heterogeneous PC Clusters, ASCE Journal of Computing in Civil Engineering, March/April, Vol:24, No: 2, (2010) 151-160

LAPACK: Linear Algebra PACKage, http://www.netlib.org/lapack/, November, 2012

Li X. S., An overview of superlu: Algorithms, implementation, and user interface, ACM Trans. Math. Softw. 31(3), pp. 302_325, (2005)

Ltaief H., Tomov S., Nath R., Du P. and Dongarra J., A Scalable High Performant Cholesky Fatorization for Multicore with GPU Accelerators, LAPACK Working Note #223

Lucas, R. F., Wagenbreth, G., Davis, D. M. and Grimes, R. Multifrontal computations on GPUs and their multi-core hosts, VECPAR, (2010)

Lulec A., Solution of sparse systems on GPU architecture (Yüksek Lisans Tezi), ODTÜ İnşaat Mühendisliği Bölümü, (2011)

MacNeal, R.H., Harder, R. L.; "A Proposed Standard set of Problems to Test Finite Element Accuracy", Finite Elements in Analysis and Design, 1 3-20, North-Holand (1985)

MAGMA: Matrix Algebra on GPU and Multicore Architectures, http://icl.cs.utk.edu/magma/, January, (2012)

NVIDIA GeForce GTX 200 GPU Architectural Overview, Technical Document, (2008)

Oster B. , Advanced Cuda Training, Nvidia 2008 Conference, (2008)

Ozmen S., Bahcecioglu T., Kurc O., A Substructure Based Parallel Dynamic Solution of Large Systems on Homogeneous PC Clusters, Proceedings of Advances in Civil Engineering, Trabzon, Turkey, (2010)

Ozmen S., Linear analysis of large structural models on PC clusters (Yüksek Lisans Tezi), ODTÜ İnşaat Mühendisliği Bölümü, (2009)

Özmen S., Kurç Ö., Yoğun Matrislerin Grafik İşlem Birimleri Yardımıyla İndirgenmesi, Başarım 2012, Ankara, Turkey, (2012)

Paraview http://www.paraview.org/

ParMetis. Parallel Graph Partitioning and Sparse Matrix Ordering Library. http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

Pinar A., Hendrickson B., Partitioning for Complex Objectives, Proc. Intl. Parallel & Distrib. Proc. Symp. (2001)

Pellegrini F. and Roman J., Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs, in HPCN Europe (1996)

RapidXML, Marcin Kalicinski, http://rapidxml.sourceforge.net/

Reddy, J. N. , Gartling, D.K. : The Finite Element Method in Heat Transfer and Fluid Dynamics, Third Edition (Computational Mechanics and Applied Analysis) (2010)

SAP 2000, Computers and Structures Inc., www.csiberkeley.com

Schenk O. , Christen M., Burkhart H. , Algorithmic performance studies on graphics processing units, Journal of Parallel and Distributed Computing 68, pp. 1360-1369, (2008)

Schenk O. and Gärtner K., Two-level dynamic scheduling in pardiso: improved scalability on shared memory multiprocessing systems, Parallel Comput. 28(2), pp. 187_197,(2002)

Shimpi A. L. , Wilson D., "GT200 Arrives as the GeForce GTX 280 & 260", web link: "http://www.anandtech.com/video/showdoc.aspx?i=3334&p=2", (2008)

Schulze J.,Towards a tighter coupling of bottom-up and top-down sparse matrix ordering methods, BIT 41, p. 2001, (2001).

Synn S. Y. and Fulton R. E., The Concurrent Element Level Processing for Nonlinear Dynamic Analysis on a Massively Parallel Computer, Computing Systems in Engineering, 6 (3), 285-293 (1995)

Tomov S., Nath R., Ltaief H., Dongarra J., Dense Linear Algebra Solvers for Multicore with GPU Accelerators, Sym. on Parallel & Distributed Processing, Workshops and Phd Forum, 1-8, (2010)

Topping B.H.V., Ivanyi P., Partitioning of Tall Buildings using Bubble Graph Representation, J. Comput. Civil Engrg. Vol. 15 No.3, 178-183 (2001)

Volkov V., and Demmel J.W., LU, QR and Cholesky factorizations using vector capabilities of GPUs. TR:UCB/EECS-2008-49, EECS Department. University of California, Berkeley, (2008)

Vucetic M., Dobry R., “Effect of Soil Plasticity on Cyclic Response”. Journal of Geotechnical Engineering, Vol. 117, No. 1, pp. 89-107 (1991)

Yang,Y. S., Hsieh S. H., Iterative mesh partitioning optimization for parallel nonlinear dynamic finite element analysis with direct substructuring, Comput. Mech. 28 456-468 (2002)

Yang, Y., Hsieh, S., and Hsieh, T. ”Improving Parallel Substructuring Efficiency by Using a Multilevel Approach.” J. Comput. Civ. Eng., 26(4), 457–464 (2012)

Yaw, L. L., 2D Co-rotational Truss Formulation,Walla Walla University,April 23, (2009)

Zhang F., The Schur Complement and Its Applications, Springer Science, New York, 2005.

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Yrd. Doç. Dr. ÖZGÜR KURÇ
Proje No:	108M586
Proje Başlığı:	Yapı Mühendisliği İçin Genişletilebilir Paralel Sonlu Elemanlar Çözümleme Platformu
Proje Türü:	Araştırma
Proje Süresi:	36
Araştırmacılar:	MUSTAFA UĞUR POLAT
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	ORTA DOĞU TEKNİK Ü. MÜHENDİSLİK F. İNŞAAT MÜHENDİSLİĞİ B.
Projenin Başlangıç ve Bitiş Tarihleri:	01/06/2009 - 01/09/2012
Onaylanan Bütçe:	157826.0
Harcanan Bütçe:	137576.0
Öz:	Son yıllarda bilgisayar teknolojilerinde meydana gelen ilerlemeler sayesinde paralel bilgisayar sistemleri artık çok daha ucuz ve ulaşılabilir hale gelmiştir. Ancak birçok kurum ve mühendis, ellerinin altındaki paralel bilgisayar sistemlerini, kullandıkları yapısal çözümleme yazılımlarının yetersizliği nedeniyle kullanamamaktadır. Bu projenin de en önemli hedeflerinden biri, yapı mühendisliğinde sıkça kullanılan sonlu elemanlar çözümlemelerini paralelleştirerek hem hali hazırda varolan paralel hesaplama donanımlarının masraf yapılmadan kullanılmasını sağlamak, hem de çözümleme sürelerinde ciddi azalmalar elde etmektir. Bu projede sonlu eleman yönteminde sıkça kullanılan doğrusal statik, doğrusal olmayan statik, zaman alanında tanımlı doğrusal ve doğrusal olmayan dinamik çözümleme yöntemleri üzerinde çalışılmıştır. Paralel çözümleme yöntemleri olarak doğrudan çözücüler incelenmiş ve toptan ve alt-yapı temelli olmak üzere iki tip çözücü üzerinde çalışılarak çözücülerin başarımları sınanmıştır. Zaman alanında tanımlı dinamik çözümleme için örtük ve belirtik integrasyon algoritmalarının paralel uygulamaları geliştirilerek başarımları karşılaştırılmıştır. Belirtik algoritmalar kullanılarak zaman alanında doğrusal olmayan problemlerin de çözümlemesi gerçekleştirilmiştir. Mekanik ve hesaplama konularıyla ilgili yazılım geliştirmenin en önemli sıkıntılarından bir tanesi, yazılım geliştiren ekibin dışındaki araştırmacıların veya kullanıcıların yazılımı kullanmalarının ve geliştirmelerinin çok zor olması, yazılıma uyum sağlayarak anlamalarının çok mesai gerektiren ve enerji tüketen bir iş olmasıdır. Bu sebepten dolayı da geliştirilen birçok akademik yazılım geniş bir kullanıcı kitlesi bulamamaktadır. Bu projenin diğer bir önemli hedefi de genişletilebilir bir yazılım yapısı oluşturarak, ileride başka araştırmacıların da kolayca kullanabilecekleri bir platform üretmektir. Bu amaçla, platform için nesne yönelimli bir veri yapısı geliştirilmiş ve parça ekleme (plug-in) teknolojisi kullanılarak yazılım platformuna dışarıdan parçalar eklenerek platformun genişletilmesine olanak verilmiştir. Platformun genişletilebilirliği ısı transferi probleminin çözümü için geliştirilen sonlu eleman ve ilgili çözümleme yöntemlerinin platforma eklenmesiyle sınanmıştır. Son olarak yeni nesil ekran kartlarının sonlu elemanlar yönteminin kullanılabilirliği çeşitli çözümleme algoritmalarının GİB (grafik işlem birimi) uygulamalarıının geliştirilmesiyle sınanmıştır. Her ne kadar seyrek matris çözücülerde istenen başarım yakalanamadıysa, yoğun matris çözücülerinde ve belirtik integrasyon yöntemiyle zaman alanında doğrusal olmayan dinamik çözümlemelerde hız bakımından ciddi kazanımlar elde edilmiştir.
Anahtar Kelimeler:	Yapısal çözümleme, paralel hesaplama, sonlu elemanlar, alt-yapı, bilgisayar kümeleri, GİB
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır