İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MATLAB PROGRAMININ DOĞRUSAL OLMAYAN DEPREM ANALİZLERİ İÇİN PARALEL PROGRAMLAMAYA UYGUNLUĞUNUN ARAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Fatih YILDIZ

İnşaat Mühendisliği Anabilim Dalı

Yapı Mühendisliği Programı

AĞUSTOS 2017

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MATLAB PROGRAMININ DOĞRUSAL OLMAYAN DEPREM ANALİZLERİ İÇİN PARALEL PROGRAMLAMAYA UYGUNLUĞUNUN ARAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Fatih YILDIZ
(501141014)

İnşaat Mühendisliği Anabilim Dalı

Yapı Mühendisliği Programı

Tez Danışmanı : Yrd. Doç. Dr. Barış ERKUŞ

AĞUSTOS 2017

İTÜ, Fen Bilimleri Enstitüsü'nün **501141014** numaralı Yüksek Lisans öğrencisi **Fatih YILDIZ**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "**Matlab Programının Doğrusal Olmayan Deprem Analizleri İçin Paralel Programlamaya Uygunluğunun Araştırılması**" başlıklı tezini, aşağıda imzaları olan jüri önünde başarıyla sunmuştur.

Tez Danışmanı : **Yrd. Doç. Dr. Barış ERKUŞ** .....
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Doç. Dr. Ali SARI** .....
İstanbul Teknik Üniversitesi

Yrd. Doç. Dr. Devrim ÖZHENDEKÇİ .....
Yıldız Teknik Üniversitesi

Teslim Tarihi : 22 Ağustos 2017
Savunma Tarihi : 25 Ağustos 2017

Sevgili Aileme,

ÖNSÖZ

Yüksek lisans çalışmalarım esnasında, bilgi, deneyim ve yardımlarını esirgemeyen değerli danışman hocam Yrd.Doç.Dr.Barış ERKUŞ'a teşekkürlerimi sunarım.

Bu güne kadar hayatımın her aşamasında maddi ve manevi desteklerini hep yanımda hissettiğim sevgili aileme minnetimi ve teşekkürlerimi sunarım.

Bu çalışma kapsamında, bana yardımcı olan çalışma arkadaşım Barış KASAPOĞLU'na teşekkürü borç bilirim.

Ağustos 2017

Fatih YILDIZ
İnşaat Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

PEER : Pasific Earthquake Engineering Research Center

AFAD : Afet ve Acil Durum Yönetimi Başkanlığı

Sap2000 : Ticari paket programı

Bknz : Bakınız

SEMBOLLER

F : Yaya uygulanan kuvvet

k : Yay katsayısı (rijitlik)

x : Yay yerdeğiştirmesi

σ : Gerilme

E : Elastisite katsayısı

ε : Şekil değiştirme

fy : Akma kuvveti

k1 : Elastik rijitlik

k2 : İnelastik rijitlik (pekleşme)

x(*t*) : Zemine göre röletif olan yerdeğiştirmeleri ve dönmeler

M : Kütle matrisi

C : Sönümleme matrisi

K : Doğrusal elemanlardan gelen rijitlik matrisi

$\mathbf{F}_s(t)$: Doğrusal olmayan eleman kuvvetleri

P(*t*) : Dış kuvvet vektörü

α : Kütle orantı sönüm katsayısı

β : Rijitlik orantı sönüm katsayısı

ξ_n : Kritik sönümleme oranı

ω_n : Doğal frekans

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MATLAB PROGRAMININ DOĞRUSAL OLMAYAN DEPREM ANALİZLERİ İÇİN PARALEL PROGRAMLAMAYA UYGUNLUĞUNUN ARAŞTIRILMASI

ÖZET

İnşaat mühendisliğindeki gelişmeler sayesinde yapısal modelleme ve çözümleme farklı şekillerde yapılabilmektedir. Yapısal modelleme sürecinde, yapıyı oluşturan kolon, kiriş, döşeme, perde gibi fiziksel elemanların davranışlarını gözönünde bulundurarak ve davranışları ile ilgili bazı kabuller yaparak, bu elemanlar için matematiksel modeller kullanılır. Fiziksel elemanların yanı sıra, doğrusal olmayan davranışı temsil edecek elemanların matematiksel modelleri de oluşturulur. Bu eleman bazındaki modeller kullanılarak yapısal sistem (çerçeve sistem, perdeli-çerçeve sistem, kabuklar, kablolu sistemler vb.) ve genel matematiksel model oluşturulur. Modellemesi biten yapıların analiz sürecinde ise toprak, rüzgar ve deprem gibi farklı yükleme durumları için çözümler yapılır.

Bu tez kapsamında, kolon ve kiriş elemanlar için çubuk eleman basitleştirmesi ve doğrusal olmayan elemanlar için de yay kabulü yapılmıştır. Bu çubuk elemanlar ve yaylar, düğüm noktaları yardımıyla birleştirilip çerçeve sistemler oluşturulur. Çubuk elemanların doğrusal, yay elemanların ise doğrusal olmayan davranış sergilediği düşünülerek sistemin doğrusalsızlığı yay elemanlara yoğunlaştırılmıştır. Aynı zamanda Matlab programlama dili yardımıyla doğrusal olmayan deprem analiz programı yazılmıştır. Bu bilgisayar programlarının bize sağladığı işlem kolaylığı ile yay ve çubuk elemanlardan teşkil edilen, istenilen açıklık ve istenilen kat sayısında çerçeve sistemler oluşturulmuş ve bu çerçeve sisteme dinamik yük uygulanarak zaman tanım aralığında analiz yapılmıştır. Dinamik yük olarak çeşitli deprem yer hareket kayıtları çerçeveye yüklenerek artırımlı zaman aralığında, deprem analizi için geliştirilen program çalıştırılmış ve analiz sonucu oluşan ivme, hız ve yerdeğiştirmeler belirlenerek yapının sismik davranışı saptanmıştır.

Günümüzde yaygın olarak kullanılmaya başlanan yüksek katlı ve çok sayıda açıklığı bulunan çerçeve sistemlerin modellenmesi esnasında kolon ve kirişlerin sayısı hatırı sayılır biçimde arttığı için fazlasıyla çubuk eleman göz önünde bulundurulmalıdır. Buna bağlı olarak her çubuk elemanda, doğrusal olmayan davranışı temsil eden yaylar da matematiksel modele entegre edilmelidir. Bu çerçeve sistemin oluşturulması ve özellikle çözülmesi esnasında Matlab programı ile geliştirilen

deprem analiz programı fazlasıyla işlem yapmaktadır. Bu işlemlerin fazlalığı analiz için gerekli olan süreyi uzatmakta dolayısıyla vakit kaybı yaşanmakta ve projelerde gecikmeler yaşanabilmektedir.

Analiz sürelerinin uzun olmasının yol açtığı problemleri çözüme kavuşturabilmek için geleneksel yöntem olan seri hesaplamalardan uzaklaşıp paralel programlama konseptinin kullanılması gerektiği ortaya çıkmıştır. Bilgisayar mimarisindeki değişimler ile ön plana çıkan paralel programlama, farklı avantajları sayesinde bir çok alanda kullanılabilmektedir. Zaman tasarrufu, bilgisayar çekirdeklerini aynı anda kullanım imkanı ve büyük verileri çözme kabiliyeti paralel programlamanın kullanım alanlarını her geçen gün artırmaktadır. Bu tez kapsamında, paralel programlama konsepti ile Matlab programında geliştirilen deprem analiz programının içerisindeki iş yükü küçük parçalara ayrılarak bilgisayarın çekirdeklerine (CPU) gönderilmiş ve eş zamanlı olarak çözümün elde edilmesi beklenmiştir. Bu sayede programların hızlandırılması ve analiz süresinin kısalması hedeflenmiştir.

APPLICABILITY OF MATLAB LANGUAGE TO PARALLEL PROGRAMMING FOR NONLINEAR SEISMIC ANALYSIS OF STRUCTURE

SUMMARY

The development of civil engineering has facilitated the modeling and analysis of constructions. In the modeling process of constructions, mathematical models are used for physical elements with considering its behaviors and making some assumptions about its behaviors. Some samples of the physical elements forming the building are columns, beams, slabs, shear walls. In addition to physical elements, mathematical model of elements representing nonlinear behavior is created. Using this element-based models, the structural system (frame system, shear walled frame system, shells, cable system etc.) and general mathematical model are generated. In the analysis process of modeled structures, solutions are made for different loading conditions such as soil, wind and earthquake.

Within the scope of this thesis, simplification of the rod element for column and beam elements and acceptance of the spring for non-linear elements are made. These rod elements and springs are connected with the node points to form frame systems. The nonlinearity of the system is concentrated on the spring elements, considering that the rod elements exhibit linear behavior and the spring elements exhibit nonlinear behavior. At the same time nonlinear seismic analysis programs were written with the aid of Matlab software language. With the ease of operation provided by the computer program, frame systems with desired number of storey and bay and that constituted by the spring and rod elements are created and the time history analysis are made by applying dynamic load to this frame system. Various earthquake ground motion records were loaded as a dynamic load on the frame, seismic analysis programs were run in the incremental time interval and the seismic behavior was discovered by determining the acceleration, speed and displacements that occurred after the analysis.

The number of columns and beams is considerably increased during the modeling of frame systems which have a lot of storeys and bays . For this, too much rod element should be considered in the analysis. Accordingly, in each rod element, springs representing nonlinear behavior must be integrated into the mathematical model. The analysis programs written in Matlab software language intensively calculate data in

the creation of this frame system and in particular its analysis. The surplus of these transactions prolongs the time required for the analysis, which leads to waste of time and delays in the projects.

In contrast to the conventional method of serial calculations, parallel programming concept has to be used in order to be able to solve the problems caused by long analysis times. Parallel programming, which comes to the forefront with the changes in the computer architecture, can be used in many areas thanks to its different advantages. The time savings, the ability to use computer cores at the same time and the ability to solve large amounts of data are increasing the usage areas of parallel programming day by day. In this thesis, considering the parallel programming concept, the workload of the analysis programs written in Matlab program is divided into small pieces and sent to the cores of the computer (CPU) and it is expected to obtain the solution simultaneously. This is aimed at accelerating the programs and shortening the analysis time.

This thesis investigates the effectiveness of parallelization of material constitutive model functions in nonlinear seismic time-history analysis of structures coded with script languages.Traditionally, nonlinear time history analysis programs are written in conventional programming languages such as FORTRAN or C. However, recently, script language such as MATLAB is also being used mostly for research purposes as they provide many useful libraries and packages

There are two types of nonlinear element models that are frequently used in nonlinear time history analysis: concentrated plasticity models and distributed plasticity models. Concentrated plasticity models are phenomenological models with single nonlinear element. Distributed plasticity models such fiber models are macro models with many nonlinear elements.

The most time-consuming stage of a nonlinear time history analysis is generally solution of a linear system of equations with a size equal to the structure's degrees of freedom. Furthermore, if fiber models are used, a system of linear equation for each fiber macro element is solved to achieve the element equilibrium.

Simulation of constitutive models may also have significant time-cost, especially if fiber elements are formed by many fibers with complicated and time-consuming constitutive models.For the solution of linear system equations parallel programming algorithms such as domain decomposition method has been utilized, which significantly reduces the analysis durations. To evaluate and improve the time-cost simulations of constitutive models using parallel programing a program is written for nonlinear analysis of two-dimensional frame type structures.

Frame structure is comprised of columns and beams that are modelled as linear frame elements. At both ends of each frame element, there are nonlinear rotational springs. These springs can have three types of nonlinearities: a bilinear behavior modelled as piecewise linear, a trilinear Takeda model and a Bouc-Wen model. Newmark's β-method is used for the solution of differential equations, and a single-step method is

used for minimization of unbalanced forces. Five frames with 10, 25, 50, 75 and 100 stories, each with four bays are generated as sample structures. These structures have 180, 450, 900, 1350 and 1800 nonlinear elements respectively.

Call to nonlinear constitutive models are programmed parallel using the tools available in the scripting languages. A version of the program with sequential loop is also written for comparison purposes.

Results show that parallel programming reduces the time duration of the nonlinear function calls significantly as the number of nonlinear elements and call duration of single element increases. It is identified that prescriptive languages provide several parallelization tools but not all tools provide efficient results. Also, overhead time in parallel simulations is observed, and a formulation is developed to find the minimum number of elements that is required to overcome the overhead.

1. GİRİŞ

1.1. Konu

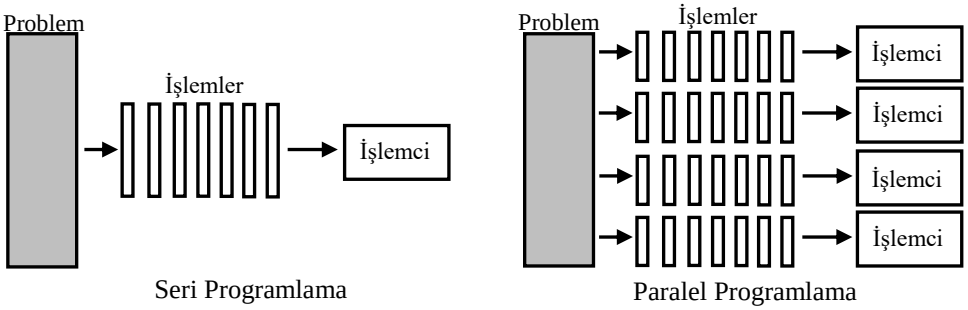
İnşaat mühendisliği, kendi anabilim dalları ve farklı disiplinlerin katkılarıyla sorunlara hızlı ve etkili cevap verebilme kabiliyetini her geçen gün artırmaktadır. İnşaat mühendisliğindeki gelişmelere katkıda bulunan önemli alanlardan biri de bilgisayar ve bilgi teknolojileridir. Hayatımızın her aşamasında karşımıza çıkan bilgisayarın inşaat sektöründe kullanımı yaygınlaşmış olsa da inşaat sektörü hala bilgisayar teknolojilerinden yeterli seviyede faydalanamamaktadır. Bilgi teknolojileri etkili şekilde kullanılmaları halinde, projelendirme ve imalat süreçlerine büyük imkanlar sunarlar. Bilgisayar destekli tasarım ve mühendislik yazılımları projelendirmede ön plana çıkarken proje yönetimi ve yapı işletmesi için geliştirilen yazılımlarda imalatta önemli rol oynamaktadır. Geçmiş senelerde iki boyutlu çizimler, teknik hesaplar elle yapılırken günümüzde yoğun verili analizler ve sonrasında üretilen üç boyutlu çizimler bilgisayar programları tarafından yapılabilmektedir. Bilgisayar ve bilgi teknolojileri sayesinde iş yükü önemli ölçüde azaltılır ve istenilen ürünler kolaylıkla elde edilebilir.

İnşa edilecek yapıların mimarı, statik, mekanik vb. projelerini oluşturmak için yerine getirilen bazı işlemler vardır. İnşaat mühendislerinin sorumluluğunda olan statik hesaplarda, hayata geçirilmesi düşünülen yapının modellemesi ve analizinin yapılması gereklidir. Öncelikle yapısal sistemin modellemesi göz önünde bulundurulur. Bu tez çalışması kapsamında, yapısal sistemin çerçeve olduğu durum üzerine yoğunlaşılmıştır. Çerçeve sistemdeki kolon ve kirişler için çubuk eleman kabulü yapılmıştır. Buna ek olarak her kolon ve kirişin doğrusal olmayan davranışını matematiksel olarak modelleyebilmek için doğrusal olmayan davranışın bir noktaya yığıldığı kabulü yapılarak çubuk elemanların uç bölgelerinde çift doğrusal dönme yayı elemanı teşkil edilmiştir. Sonlu elemanlar yönteminde vurgulanan düğüm noktaları sayesinde bu çubuk ve yay elemanların birbirine bağlantısı sağlanmıştır. Elemanların bağlantıları sonrasında, doğrusal olmayan yay elemanları ile teşkil edilen çerçeve sistem elde edilmiştir.

Çerçeve sistemlerin hesabını yapabilmek için Matlab programlama dili kullanılarak doğrusal olmayan deprem analizi programı yazılmıştır. Bu program sayesinde çerçeve sistem istenilen açıklık ve kat sayısına göre oluşturulabilmekte, kat yüksekliği ve açıklık mesafesi belirlenebilmektedir. Bu yapısal sistemi zorlayan yükleme ise dinamik yük olan depremdir. Dinamik yükleri statik yüklerden ayıran temel özellik, dinamik yüklerin zamana bağlı değişimleridir. Bu farklılık analizde izlenen çözüm yöntemlerinin de değişimine sebeb olmakta ve diferansiyel denklemlerin çözümü gerekmektedir. Dinamik dengeyi temsil eden bu diferansiyel denklemi zaman tanım aralığında çözmek bazı kabuller yaparak mümkündür. Bu kabullerden biri ivmenin çok küçük zaman aralıklarında doğrusal değiştimi, diğeri ise ivmenin ortalama değerinin hesaplanması şeklindedir. Newmark'ın bu alanda yaptığı çalışmalar sonucunda diferansiyel denklem çok küçük zaman aralığında cebirsel olarak adım adım çözülerek yapıda oluşan ivme, hız, yerdeğiştirme değerleri bulunabilir. Bu değişkenler yapının deprem esnasındaki davranışını gösterir.

Çerçeve sistemlerin büyük olması, deprem analizlerinin hassas hesaplanma isteği sonucu zaman aralığının küçük tutulması, depremin birden fazla yöndeki bileşenin gözönünde bulundurulması ve birden fazla depremle analiz yapılması gibi sebeblerden dolayı analizler karmaşık bir hal almaktadır. Bununla birlikte doğrusal olmayan yay elemanların analiz esnasında etkili bir şekilde işleme katılmaları analizi zorlamaktadır. Tüm bunlar doğrusal olmayan deprem analizi için gerekli olan süreyi fazlasıyla uzatmaktadır. Bu sürenin kısaltılması vaktin etkili kullanılması için önemlidir.

Matlab ile yazılan doğrusal olmayan deprem analiz programının paralel programlama bakış açısıyla güncellenmesi analizleri hızlandıracaktır. Paralel programlama ile iş yükü küçük parçalara bölünerek bilgisayarın farklı çekirdeklerine gönderilebilir. Bu sayede her çekirdek aynı anda çalışarak kendisine gelen işi çözümler ve ana programa geri gönderir (bknz Şekil 1.1). Bu işlemin aynı anda gerçekleşmesi sürenin kısalmasına sebeb olacaktır. Paralel programlama ile güncellenen deprem analiz programı sayesinde veriler daha hızlı şekilde işlenebilir ve sonuçlar vakit kaybı olmadan elde edilebilir. Bu sayede yüksek katlı binalarda çözüm yapılırken bütün taşıyıcı sistem elemanları modellenip hesaba dahil edilebilir. Çünkü paralel programlama ile yoğun verili problemler çekirdeklere bölünerek hızlı bir şekilde işlenebilmektedir.



Şekil 1.1 : Seri ve paralel programlama yöntemlerinin şematik gösterimi.

1.2. Tezin Amacı ve Kapsamı

Bu tez kapsamında yapılan çalışmada, çubuk elemanlar ve her bir çubuk elemanın uçlarında doğrusal olmayan yay elemanı olduğu düşünülmüştür. Bu çubuk elemanlar ile istenilen açıklık ve kat sayılı çerçeve sistemler oluşturulmuştur. Bu çerçeve sistemlerin doğrusal olmayan deprem analizleri için Matlab programlama dilinde program geliştirilmiştir. Çerçeve sistem büyüklüğüne bağlı olarak artan bünye fonksiyonu denklemlerinin paralel programlama ile hızlandırılması hedeflenmiştir.

1.3. Çalışmada İzlenen Yöntem

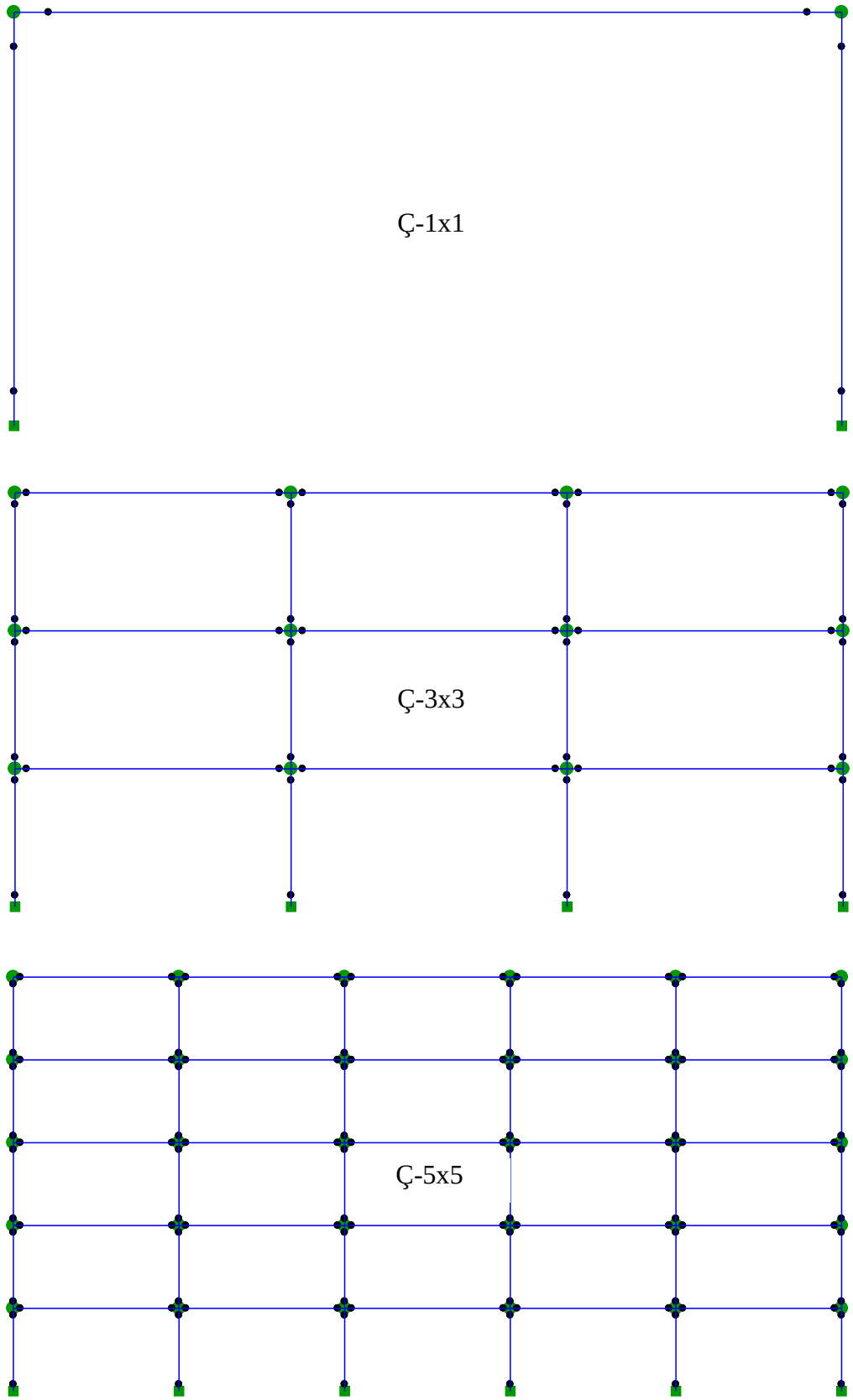
Tez kapsamında gözetilen hedefe ulaşabilmek için Matlab programlama diliyle yazılan doğrusal olmayan deprem analiz programıyla farklı büyüklükteki çerçeve sistemler oluşturulmuştur. Oluşturulan sekiz farklı büyüklükteki çerçeve sistemin açıklık ve kat sayıları Çizelge 1.1' de ve bazı çerçeve sistemler için örnek figürler Şekil 1.2'de verilmiştir. Bu numuneler için doğrusal olmayan deprem analizi tekrarlanarak çerçeve sistemlerin tepe yerdeğiştirme, hız ve ivme değerleri bulunmuş ve paket programlar ile kıyaslanarak doğruluğu teyit edilmiştir. Bununla birlikte bazı düğüm noktalarındaki çevrimsel davranış ticari programlar ile kıyaslanmıştır.

Çerçeve sistemlerin herbiri için doğrusal olmayan bünye fonksiyonu sayısı tespit edilip hızlanmaları üzerine araştırma yapılmıştır. Matlab programlama dilinde, paralel programlama araç kutusunda sunulan 'parfor' ve 'spmd' ifadeleri doğrusal olmayan deprem analizlerinde gözönünde bulundurulan doğrusal olmayan elemanlara uygulanmıştır. Çift doğrusal ve takeda doğrusal olmayan eleman bünye fonksiyonları için öncelikle 'parfor' ile paralelleştirme çalışması yapılmıştır. 'Parfor' da ortaya çıkan doğal ek süreleri tespit edebilmek için çalışma süresi ayarlanabilen bir fonksiyon oluşturulmuş ve farklı çalışma sürelerine bağlı olarak doğal ek süreyi

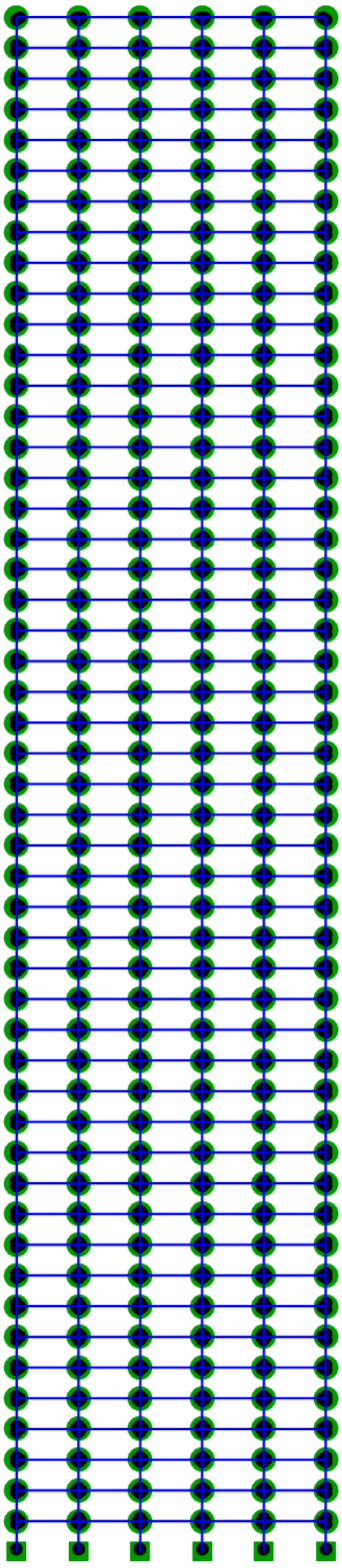
yenebilecek eleman sayısı tespit edilmiştir. Çift doğrusal ve takeda doğrusal olmayan bünye fonksiyonlarının çalışma süreleri bulunup bu fonksiyonların ‘parfor’ ile hızlanabilmesi için gerekli olan eleman sayısı belirlenmiştir. ‘Spmd’ ifadesiyle paralelleştirme çalışması esnasında ortaya çıkan doğal ek süre ‘parfor’ fonksiyonuna benzer şekilde tespit edilerek gerekli olan en az eleman sayısı ortaya konmuştur. ‘Spmd’ ile hem sadece fonksiyon çağırımı hem de statik simülasyon yapılmıştır.

Çizelge 1.1 : Çerçeve sistemlerin kat ve açıklık sayıları.

No	Kat ve Açıklık sayısı	Gösterim
1	Tek katlı tek açıklılı çerçeve sistem	Ç-1x1
2	Üç katlı üç açıklıklı çerçeve sistem	Ç-3x3
3	Beş katlı beş açıklıklı çerçeve sistem	Ç-5x5
4	On katlı beş açıklıklı çerçeve sistem	Ç-10x5
5	Yirmibeş katlı beş açıklıklı çerçeve sistem	Ç-25x5
6	Elli katlı beş açıklıklı çerçeve sistem	Ç-50x5
7	Yetmişbeş katlı beş açıklıklı çerçeve sistem	Ç-75x5
8	Yüz katlı beş açıklıklı çerçeve sistem	Ç-100x5



Şekil 1.2 : Bazı çerçeve sistemlerin görünümleri.



Ç-50x5

Şekil 1.2 (devam): Bazı çerçeve sistemlerin görünümleri.

2. LİTERATÜR ARAŞTIRMASI

2.1. Yapısal Elemanların Davranışları

Temel, kolon, kiriş, perde duvar, döşemeler ve bağ kirişleri gibi binayı oluşturan taşıyıcı elemanlar yapısal eleman olarak düşünülür. Binanın taşıyıcı elemanları dışında kalan bütün elemanlar yapısal olmayan eleman olarak tanımlanabilir. Bu tez kapsamında sadece yapısal elemanlar üzerinde araştırma yapılmıştır.

2.1.1. Doğrusal davranış sergileyen elemanlar

Elastik şekil değiştirme üzerine 17. yy da çalışmalar yapan fizikçi Robert Hooke, yayların davranışını gözlemlemiştir. Yaya uygulanan kuvvet ile yayın esnemesi arasında yayın fiziksel özelliklerine bağlı olarak sabit bir orantı olduğunu farketmiştir. Denklem 2.1'de bu bağıntının matemaktisel formülü yazılmıştır.

$$F=kx \tag{2.1}$$

Burada

F : Yaya uygulanan kuvvet

k : Yay katsayısı (rijitlik)

x : Yay yerdeğiştirmesi

Yaylar üzerinde tatbik edilen bu yükleme durumu silindir bir çubuk üzerine uygulanırsa ve kuvvet yerine gerilme, esneme yerine birim şekil değiştirme kullanılırsa karşılaşacağımız durum denklem 2.2'de verilmiştir.

$$\sigma = E\varepsilon \tag{2.2}$$

Burada

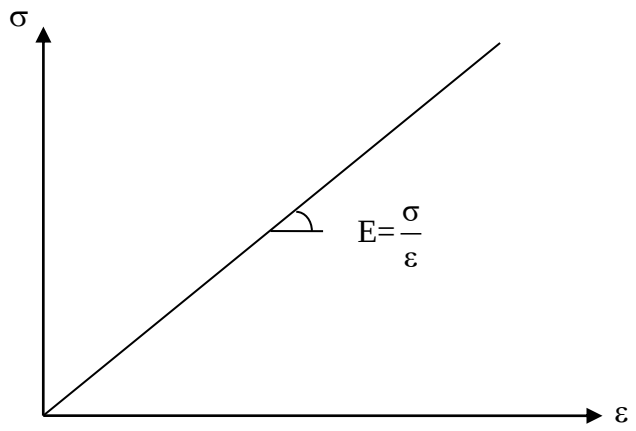
σ : Gerilme

E : Elastisite katsayısı

ε : Şekil değiştirme

Malzeme bazında gerilme ve şekil-değiştirme ilişkisi denklem 2.2'de açıklandığı gibidir. Burada E ile ifade edilen parametre elastisite katsayısı olup konu üzerine çalışan Thomas Young'a ithafen young modülü olarakta isimlendirilmiştir (bknz Şekil 2.1).

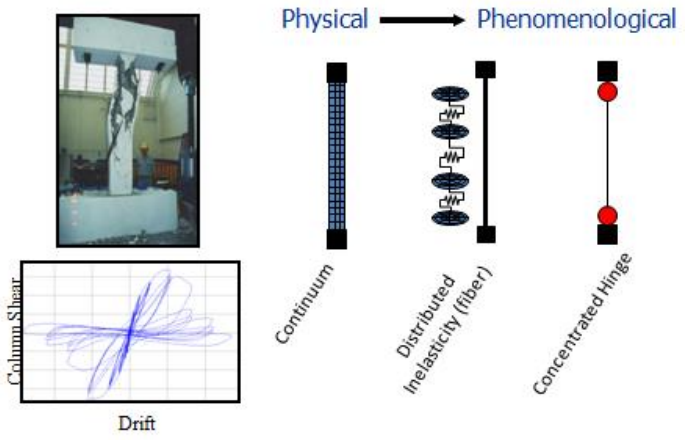
Bu çalışmalar gözönünde bulundurulduğunda elemanların doğrusal davranışları rahatlıkla anlaşılabilmektedir.



Şekil 2.1 : Elastisite modulü (Young sabiti).

2.1.2. Doğrusal olmayan davranış sergileyen elemanlar

Doğrusal olmayan davranış gözönünde bulundurulan elemanın cinsine göre değişkenlik göstermektedir. Çelik ve betonarme elemanların akma değerleri farklı olduğu için doğrusal olmayan davranışları da birbiriyle aynı modellenememektedir. Doğrusal davranış sergilemeyecek kolon elemanlar için tercih edilecek modeller Şekil 2.2'de gösterilmiştir. Bunlar sürekli modeller, fiber modeller ve bir noktaya yoğunlaştırılmış konsatre modeller olarak düşünülebilir. Bu bilgiler Pasifik Deprem Mühendisliği Araştırma Merkezinden alınmıştır (PEER, 2010).



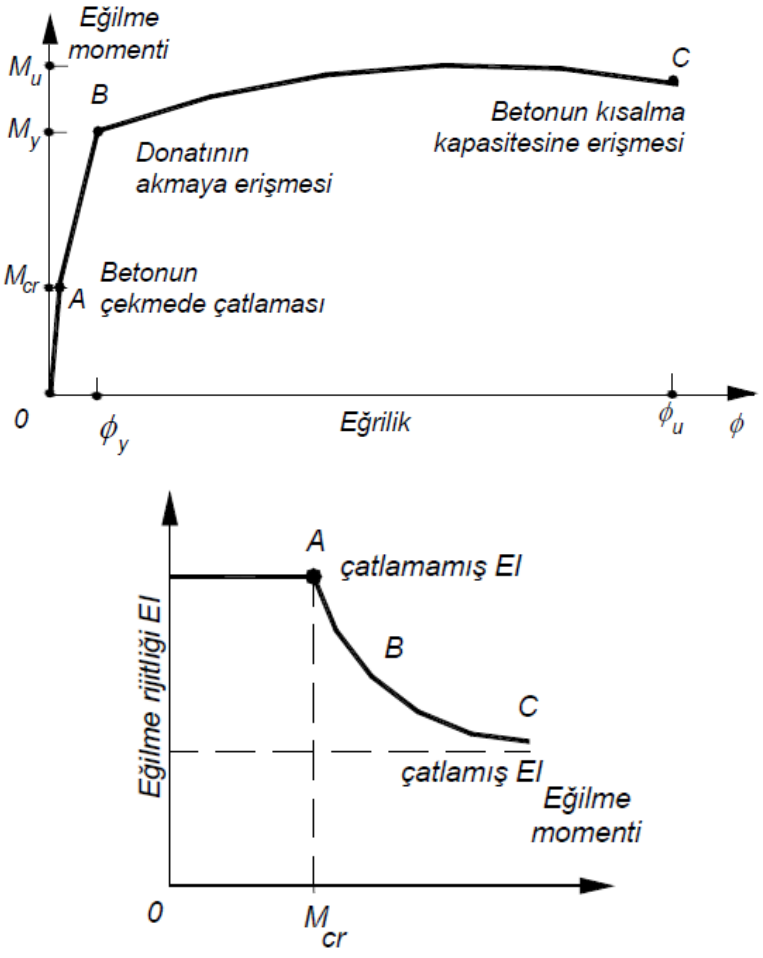
Şekil 2.2 : Doğrusal olmayan model çeşitleri.

2.2. Betonarme Yapılarda Doğrusal Olmayan Davranış

Betonarme elemanların modellenmesi için doğrusal olmayan analiz yöntemleri eski yıllardan beri akademik çevrelerce kullanılmasına rağmen pratikte çalışan mühendisler tarafından ancak son yıllarda kullanılmaya başlanmıştır. Doğrusal olmayan analizler deprem yüklerine maruz kalan yapıların davranışını en iyi yansıtan yöntem olarak kabul görmektedir.

2.2.1. Betonarmede eğilme momenti – eğrilik ilişkisi

Eğilme momentinin küçük olduğu durumlarda donatı ve beton kesit doğrusal davranmaya devam etmektedir. Eğilme momenti arttığında betonun belirli bölgelerinde çatlaklar ve kırılmalar oluşur. Bu çatlaklar kesitte eğilme momenti – eğrilik ilişkisinde doğrusalsızlığı başlatmaktadır. Betonun doğrusal olmayan davranışı gerilmelerin artmasıyla artış göstermektedir. Eğilme momenti artarken, beton kesitte eğrilik doğrusal olmayan şekilde değişmekte ve donatı akma gerilmesine ulaşmaktadır (Celep, 2007).



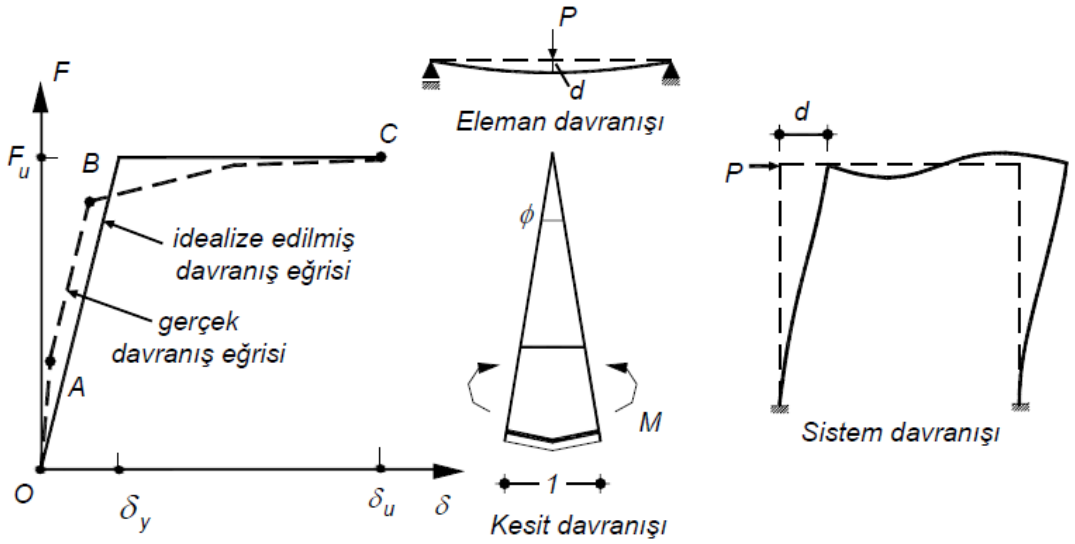
Şekil 2.3 : Betonarmede eğilme momenti-eğrilik ilişkisi.

2.2.2. Süneklik

Süneklik, elastik sınırın ötesinde şekil değiştirme veya yerdeğiştirme yapma yeteneği olarak tanımlanabilmektedir. Belirli aralıklarla meydana gelen depremin oluşturduğu kuvvetleri yapının doğrusal olmayan davranış ile karşılaması beklenir. Yapının doğrusal hareket sınırlarını geçtikten sonra içsel kuvvet üretebilme kabiliyeti azalır ve kesit zorları fazla artmasa bile yüksek deplasmanlar yapmaktadır. Bu şekilde deprem hareketinin ivmeli ve çevrimsel etkisi enerji yutularak sönümlenebilmektedir.

Sünek bir yapı deprem esnasında büyük salınımlar yaparak zeminden gelen enerjiyi soğurabilir. Süneklikle akmaya ulaşan kesitlerde plastik şekil değiştirme gözlemlenir. Performansa göre tasarım ilkeleriyle oluşturulan binanın deprem esnasında oluşturacağı hasar belirlenebilir.

Betonarme bir elemanın plastikleşen uçlarındaki gerçek doğrusal olmayan davranış lineer parçalı elemanlar ile idealleştirilebilir. Süneklik kesit, eleman ve sistem davranışlarında önemli bir faktör olarak karşımıza çıkmaktadır (bknz Şekil 2.4).



Şekil 2.4 : Şekil değiştirme ilişkileri.

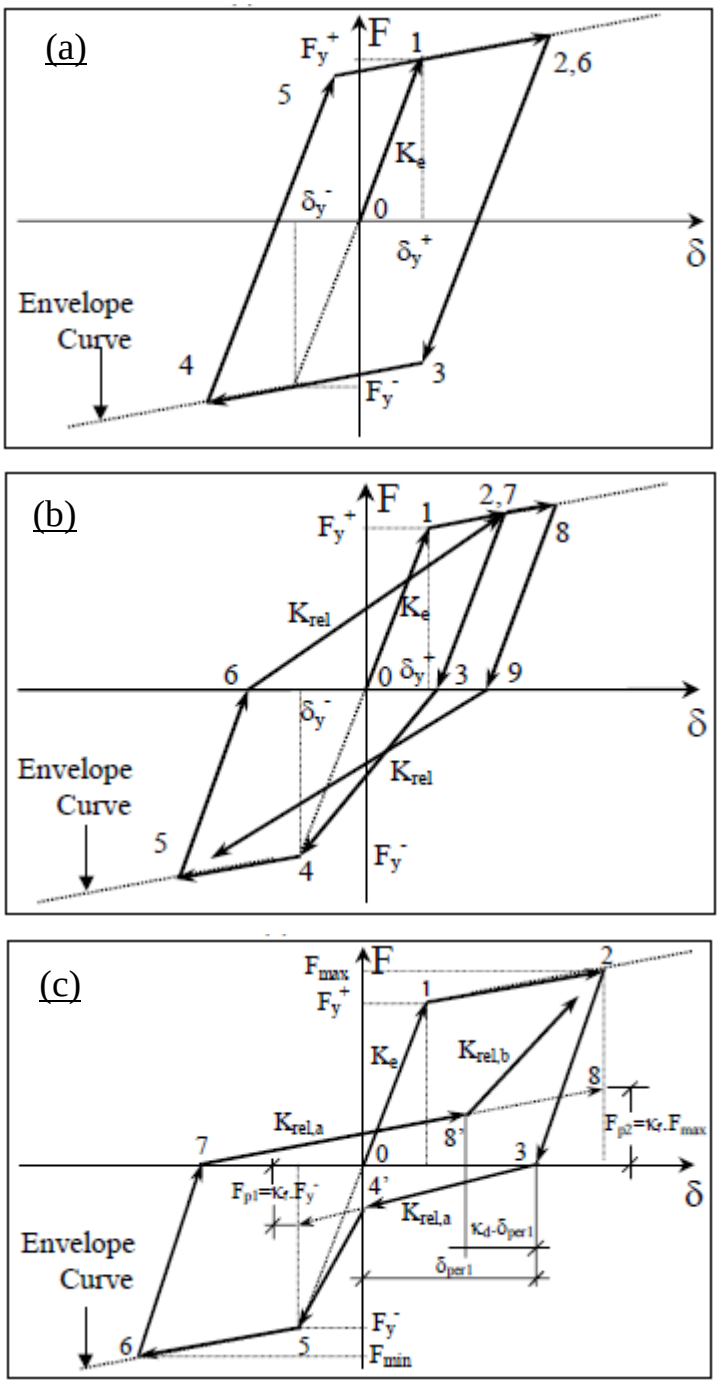
2.2.3. Histeritik modeller

Doğrusal olmayan davranışı temsil edebilmek için kullanılan bazı matematiksel modeller mevcuttur. Çift doğrusal (bilineer), takeda, bouc-wen bunlardan bazılarıdır.

Elemanların kuvvet – yerdeğiştirme ilişkileri deneysel çalışmalar sonucu elde edildikten sonra bunları matematiksel olarak ifade edebilecek modeller geliştirilir. Bu modeller sürekli bünye fonksiyonları olabilir. Sürekli bünye fonksiyonlarını hızla

çözebilmek için bazı basitleştirmeler ve idealleştirmeler gerçekleştirilerek lineer parçalar olarak ifade edilebilir.

Bazı temel histeritik davranış kuralları, bilineer, peak oriented ve pinching benzeri oluşturulan maatematiksel modeller olarak bilinir (Medina ve Krawinkler, 2003).



Şekil 2.5 : Histeritik davranışlar : (a) bilineer (b) peak oriented (c) pinching (Medina ve Krawinkler, 2003).

2.3. Doğrusal Olmayan Davranışın Modellenmesi

Doğrusal olmayan davranışın uzun yıllara dayanan bir serüveni olmakla birlikte son yıllarda bilgisayar mimarisindeki gelişmeler ile analiz programlarına entegre edilmeye başlamıştır. Doğrusal olmayan modelleme sayesinde yapısal sistemde ki davranış ayrıntılı ve gerçekçi bir şekilde ortaya konabilmektedir. Elastik ötesi yerdeğiştirme kuvvet ilişkisini inceleyip idealize eden bir çok model vardır.

2.3.1. Plastik mafsal modeli

Doğrusal olmayan analizlerde kesitlerin ulaşabileceği nihai bir dayanım değeri vardır. Elemanlara gelen yüklerin artması sonucu rijitliklerinde azalmalar oluşur ve bu nihai dayanım değerlerine ulaşılır. Bu yükleme altında kalıcı şekildeğiştirmeler başlar ve yükü taşıyamayacak hale gelene kadar bu plastik şekildeğiştirmeler artarak devam eder. Literatürde plastik mafsalla ilgili bazı çalışmalar şunlardır: Clough ve diğ. (1965), Giberson (1967), Nigam (1970), Powel ve Chen (1986).

Yapıların deprem davranışlarını incelerken lineer statik analiz ile elde edilemeyecek yapının plastik hasar alma esnasında nasıl davranış sergileyeceği, elemanlarda kuvvet dağılımının nasıl olacağını gibi konuları hakkında bilgi edilebilir (Krawinkler ve Seneviratne, 1998).

Doğrusal olmayan bölgenin yayılı veya yoğunlaşmış bir bölge olduğu kabulüne göre yayılı ve yığılı plastik davranış modelleri oluşturulmuştur.

2.3.1.1. Yığılı plastik mafsal modeli

Yığılı plastik mafsal modeli, doğrusal olmayan davranışın elemanda belirli bir bölgesinde oluştuğu, sadece bu bölgede plastik şekildeğiştirmelerin olduğunu kabul eder. Elemanın diğer kısımlarında doğrusal davranışın devam ettiğini öngörüsünde bulunur. Genellikle kolon ve kirişlerin uç kısımlarında elastik ötesi davranışın oluştuğu düşünülerek bu kısımlara yığılı plastik mafsal için matematiksel modeller konulmaktadır.

Plastik mafsal doğrusal olmayan analizlerde rijit plastik mafsal veya eğrilik mafsalı olarak tanımlanabilir. Rijit plastik mafsal, malzeme akma kuvvetine gelinceye kadar herhangibir deplasman yapmaz yani sonsuz rijitliği vardır. Akma değerine ulaştıktan sonra ise teorik olarak sonsuz deplasman yapabilme kabiliyetine sahiptir. Deneysel olarak kırılma noktasına kadar deplasman yapabilmektedir. Eğrilik mafsalı, kuvvet

deplasman ilişkisi eğilme değerine göre hesaplanır. İlk rijitlik sonsuz rijitlik yerine belirli bir değer ile ilerler.

2.3.1.2. Yayılı plastik mafsal modeli

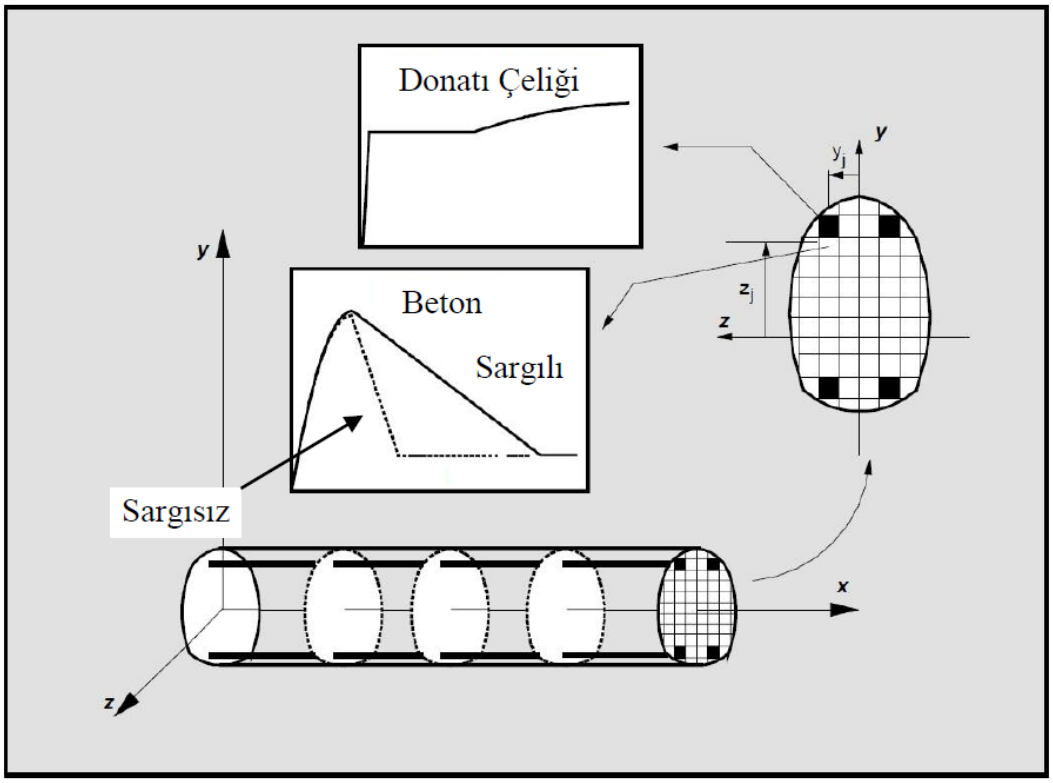
Yayılı plastik mafsal modeli, yapısal elemanlarda elastik ötesi şekil değiştirmenin belirli bölgelerde sürekli olarak devam etmesini temsil etmektedir. Kiriş ve kolonlarda bu sürekliliğin karakterini belirleyebilmenin zorluğundan dolayı araştırmacılar yaptıkları çalışmalarla bu davranışı idealleştirmek istemişler ve farklı formüller oluşturmuşlardır. Literatürde en yaygın olarak bilinen Park ve Pauley (1975), Pauley ve Pristley (1992), Fardis ve Biskinis (2003)'in geliştirmiş oldukları yaklaşımlardır.

2.3.2. Fiber model

Doğrusal olmayan davranış yığılı modeller ile bir noktaya toplanır ve bütün doğrusalsızlık burada tek bir elemanla ifade edildiği için doğası gereği çok kaba bir yaklaşım ortaya çıkar. Bunun yerine doğrusal olmayan davranış sergileyeceği düşünülen bölgelere detaylı analiz yapabilmek için fiber modeller kullanılabilir.

Fiber modeller bulundukları bölgede birden fazla liflerden oluşur ve her lifin kendine ait bir bünye fonksiyonu vardır. Elemanın doğrusalsızlığı bu lifler sayesinde mikro boyutta modellenebilir. Kolon, kiriş elemanlarda da kullanılabilen fiberler perde duvar modellemesi için oldukça elverişlidir. Bir betonarme duvar kesiti beton ve çelik malzemesinin lifleri olarak tanımlanabilir (PEER, 2010) (bknz Şekil 2.6). Betonarme duvarın belirlenen bir malzeme modeli ile oluşturulması için perde duvar istenen sayıda parçalara bölünmelidir. Her bir parçaya bir fiber tanımlanarak perde duvar analizi yapıldıktan sonra elemanın davranışı tespit edilebilmektedir.

Fiber modellerde, malzeme modeli ne kadar gerçekçi ve hassas modellenirse elemanın davranışı o kadar hassas ve gerçeğe yakın olur. Bu modelin, eksenel kuvvet ve eğilme davranışı arasındaki ilişkiyi elde etmesi, uzama ve kısalma davranışını detaylı bir şekilde belirlemesi gibi faydaları vardır.



Şekil 2.6 : Fiber model (ATC 72).

2.4. Zaman Tanım Aralığında Doğrusal Olmayan Analiz

Deprem yer hareketi esnasında yer kabuğunda oluşan ivme değerleri belirli konumlara yerleştirilen sensörler sayesinde üç boyutlu olarak kayıt altına alınır. Bu ivme değerleri saniyenin binde biri mertebesindeki zaman aralıklarında 'g' cinsinden kaydedilebilmektedir. Farklı veri tabanları tarafından bu kayıtlar depolanmaktadır (AFAD, 2017; PEER, 2017). Güçlü deprem yer hareketleri yüksek yıkıcı etkiye sahip oldukları için zaman tanım aralığındaki deprem analizlerinde tercih edilebilmektedir.

Yapılara etkiyen deprem ivmesinin matematiksel olarak modellenmesi sonucu ortaya diferansiyel bir hareket denklemi çıkmaktadır. Bu diferansiyel denklemi çözüme kavuşturabilmek için araştırmacılar farklı sayısal yöntemler üzerine çalışmıştır. En yaygın kullanılan sayısal yöntem Newmark-β yöntemidir

2.4.1. Newmark-β yöntemi

Yapısal sistemlerin dinamik analizi için doğrudan integrasyon yöntemi olarak Newmark-β kullanılabilmektedir (Newmark, 1959). Bu yöntemle ele alınan yapısal sistem seçilen küçük zaman aralıklarında zamana bağlı olarak diferansiyel denklem

integre edilmektedir. Newmark yöntemi başka araştırmacılar tarafından irdelenmiş ve farklı formlarda ifade edilmiştir. Erkus (2004) buna örnek gösterilebilir.

Çözümü yapılacak yapının tüm düğüm noktaları oluşturulup bu düğüm noktalarındaki serbestlik dereceleri tanımlandıktan sonra dinamik diferansiyel denklem zaman tanım aralığında çözülür.

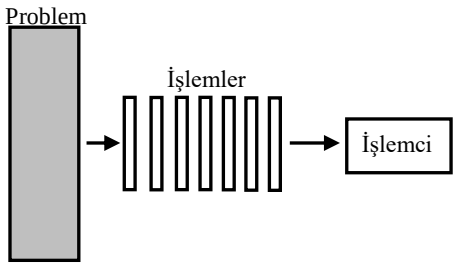
2.4.2. Dengelenmemiş kuvvet ve düzeltme yöntemi

Doğrusal olmayan analizde kabul edilen içsel kuvvetler ile malzeme modelinden gelen içsel kuvvetin farkı dengelenmemiş kuvvet olarak karşımıza çıkar. Bu dengelenmemiş kuvveti düzeltmek için Ben-Israel (1966) çalışmasında bahsedilen Newton Raphson yöntemi veya Powell ve diğ. (1973) çalışmasında geçen dengesiz kuvvet düzeltme yöntemi kullanılabilir.

2.5. Paralel Programlama

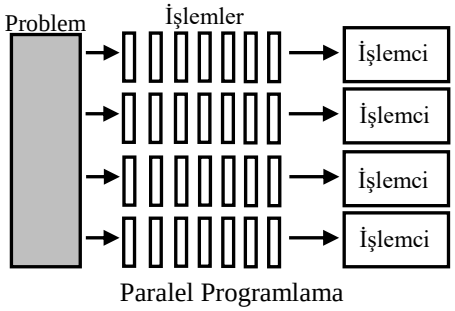
Paralel programlama, bir problemin parçalara ayrılarak her bir işlemci veya işlemci çekirdeğinde paralel bir şekilde eş zamanlı olarak çalıştırılmasına denir. Son yıllarda yapay zeka ve makine öğrenmesi gibi birçok bilimsel araştırma alanında da kullanılmaya başlanan paralel programlama ile birden fazla işlemci ve/veya işlemcilerin birden fazla çekirdeği eş zamanlı bir şekilde kullanılarak programların performansı arttırılabilmekte ve yapılacak işlemlerin süresi kısaltılabilmektedir.

Paralel programlamanın anlaşılabilmesi için öncelikle seri programla hakkında bazı bilgilere sahip olmak gerekmektedir. Seri bağlı bilgisayarlarda yazılan programlar, bir işlemci tarafından hesaplanır. Program birbirini takip eden farklı komutların bütünleşmesi ile oluşur ve ard arda gelir. Bir komut işlendikten sonra diğer komuta geçilir. İşlemler öncelik sırasına göre yapılır. Bir anda sadece bir komut işlem görür (bknz Şekil 2.7).



Şekil 2.7 : Geleneksel bilgisayarlarda seri hesaplama.

Paralel programlamadan bahsedecek olursak karşımıza çıkan ilk farklılık birden çok komutu aynı anda farklı çekirdeklerde işleyebiliyor olmamızdır. Birden fazla çekirdeği olan bilgisayarlarda paralel programlama yapılabilmektedir. Bunun yanı sıra ekran kartıları ve bilgisayar kümeleri de paralel programa için kullanılan argümanlardır. Temel olarak paralel programlamada dikkat çeken özellik birden çok işlemcinin kullanılmasıdır. Programda işlenen veriler, parçalara bölünerek eşzamanlı olarak çözülür. Veriler aynı anda farklı çekirdeklerde işleme tabi tutularak hesaplanırlar (bknz Şekil 2.8).



Şekil 2.8 : Birden çok işlemcili bilgisayarlarda paralel programlama.

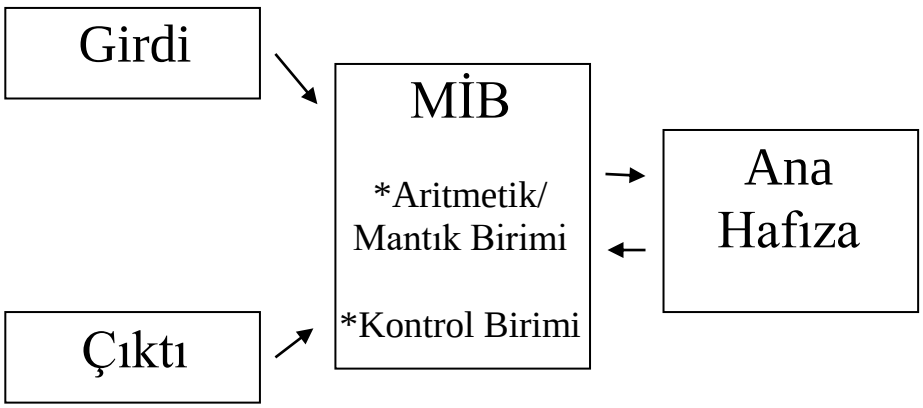
2.5.1. Paralel programlama tarihçesi

Paralel programlama ile ilgili ilk fikri Gill (1958) ortaya atmıştır. Aynı yıl IBM araştırmacıları John Cocke ve Daniel Slotnick paralel programlamanın nümerik hesaplamalarda kullanılabileceğini tartışmaya açmıştır (Wilson, 1994). Burroughs Corporation firması tarafından 1962 yılında 4 bilgisayar 16 bellek ile ilk çoklu bilgisayar sistemi geliştirmiştir (Anderson ve diğ., 1962). 1967 yılında Amdahl ve Slotnick AFIPS konferansında yayınladıkları makale ile paralel hesaplamanın fizibilitesini tartışmaya açmıştır ve paralelleştirmenin belli bir hızlanma limiti olduğunu öne sürmüştür (Wilson, 1994). Bu makalede ortaya atılan fikir sonraki yıllarda "Amdahl Kanunu" olarak literatürde yerini almıştır. Carnegie Mellon Üniversitesinde 1970'li yıllarda çok işlemcili bilgisayarı geliştirmiştir. 1983 yılında NASA Goddard Space Flight Center içinGoodyeat Massively Parallel Processor (MPP) süper bilgisayarı geliştirilmiştir (Fung, 1983). Dağınık bellekli sistemlerde bilgisayar iletişim protokolü olan MPI 1992 yılında yayınlanmıştır. 1993 yılında IBM ilk SP1 Powerparallel sistemi piyasaya sürmüştür (Punzo ve diğ., 1994). Çoklu işlemeyi destekleyen bir uygulama olan OpenMP'nin ilk arayüz programı Fortran 1.0 için 1997 yılında yayınlandı. IBM 2000 yılında POWER4 işlemcileri ile ilk çift

çekirdekli işlemciyi üretmiştir. 2007 yılında Intel ilk dört çekirdekli işlemciyi üretmiştir. Bugün kullanılan i7 işlemcilerinde 6 veya 8 çekirdekli olabilmektedir.

2.5.2. Von Neuman mimarisi

Bilgisayar teknolojisinin ilk çıktığı yıllarda sabit programlı makinalar kullanılmaktaydı. Bu makinaların işlem yapabilmeleri için gerekli olan veriler delikli kağıtlar veya kablolar yardımıyla verilmekteydi. Verilerinde depolanabilme fikri 1930'lu yıllarda gündeme geldi. Veriyileri tutabilen bellekler tasarlanmaya başlandı. 1950'li yıllarda John von Neumann, komut ve verileri belleğinde tutabilen bir bilgisayar tasarladı. 'Von Neumann mimarisi' terimi bu çalışmalar esnasında ortaya çıkmıştır.



MİB : Merkezi işlem birimi

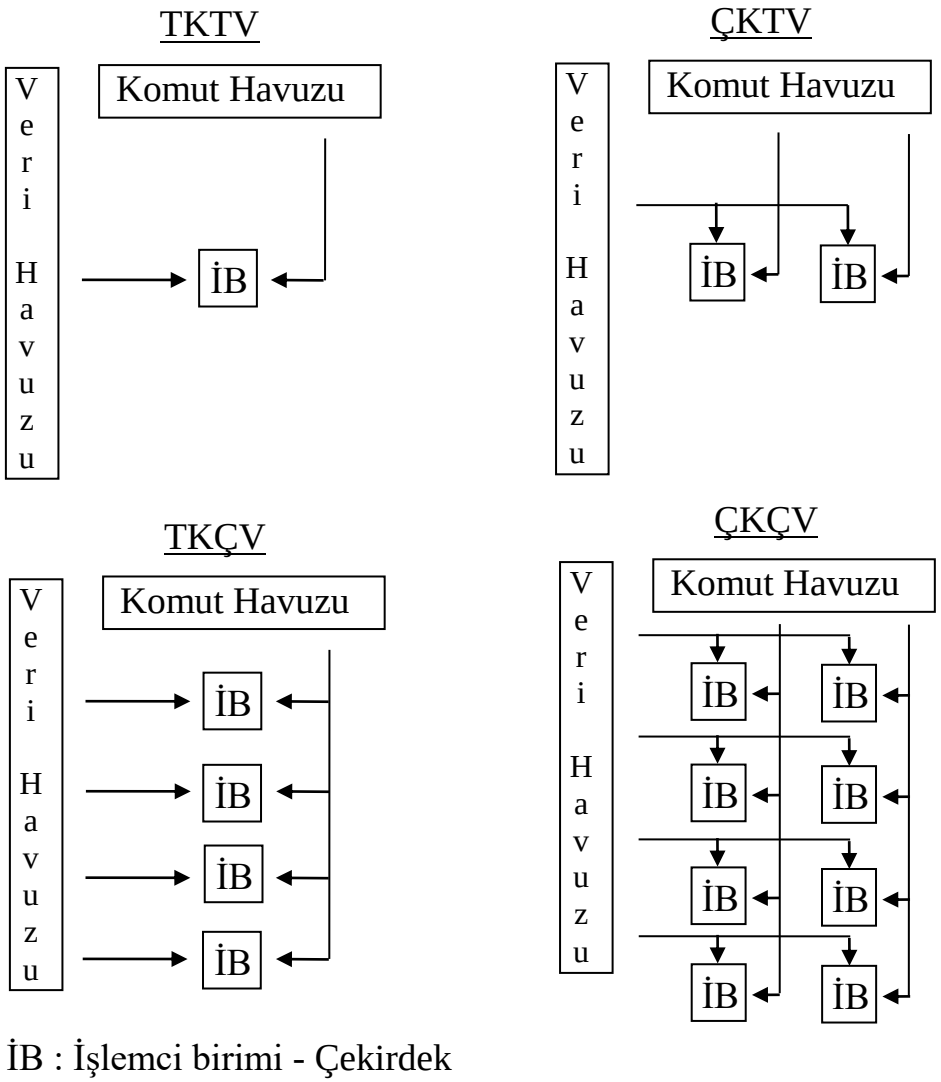
Şekil 2.9 : Von Neumann mimari.

Von Neumann mimari 4 ana bölümden oluşmaktadır (bknz Şekil 2.9). Bunlar hafıza, kontrol ünitesi, aritmetik/mantık birimi ve girdi/çıktı bilgileridir. Programlanan görevi başarıyla bitirebilmek için kontrol ünitesi veri ve komutları hafızadan getirir, komutları çözümler daha sonra işlemleri sıralı bir şekilde gerçekleştirir. Aritmetik işlem birimi temel matematiksel işlemleri icra eder.

2.5.3. Paralel programlama sınıflandırması

Paralel bilgisayarlar birçok şekilde sınıflandırılabilmektedir. Handler'ın sınıflandırması, yapısal sınıflandırma, tane boyutuna göre sınıflandırma ve Flynn'nin sınıflandırması bunlara örnek verilebilir. Bunlardan biri olan komut ve veri akışına dayalı sınıflandırma, veri ve komut arasındaki ilişkiyi gözönünde bulundurur. Flynn

(1966) tarafından yapılan klasik sınıflandırma ile dört farklı grup oluşmaktadır (bknz Şekil 2.10).



Şekil 2.10 : Flynn klasik sınıflandırması.

Tek komut tek veri (TKTV) : Bu bilgisayar yapısı tek işlemcilidir ve tek bir veri akışı üzerinde işlem yapan tek bir komutu sıralı biçimde çalıştırır. Klasik Von Neumann mimarisidir. Geleneksel tek işlemcili bilgisayarlar buna örnek verilebilir.

Tek komut çoklu veri (TKÇV) : Bu bilgisayarlar birden çok özdeş işlemciden oluşur ve bu işlemcilerin kendi yerel hafızası bulunur. Her bir işlemciye bir tane gelecek şekilde birden fazla veri vardır. Eş zamanlı olarak, bütün işlemciler tek bir komut ile farklı veri grupları üzerinde çalışırlar. Bu, veri seviyesindeki paralelleştirmeye örnektir.

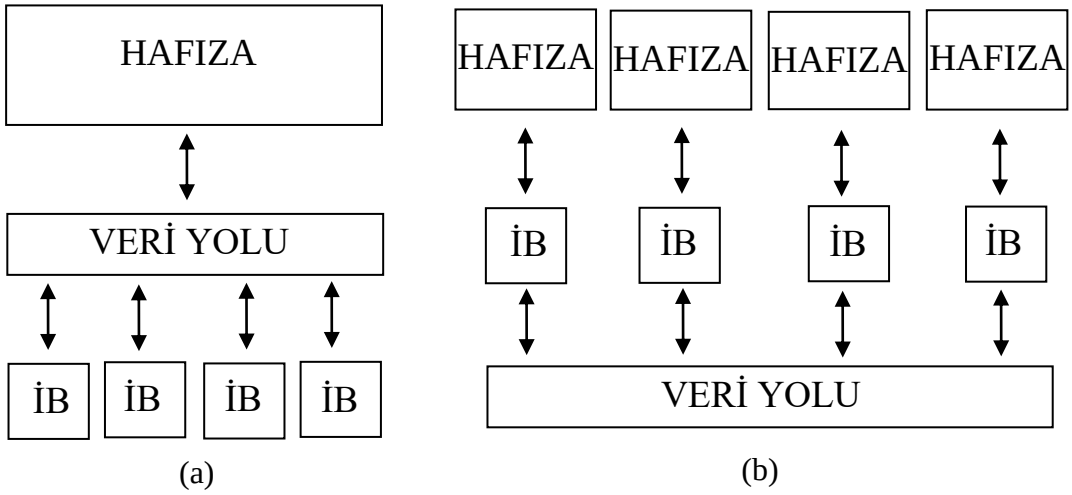
Çoklu komut tek veri (ÇKTV) : Bu modeller birden çok işlemciden oluşur. Her işlemcinin kendine ait yerel bir kontrol birimi vardır ve işlemciler tek bir hafıza birimini paylaşırlar. Bütün işlemciler hafızadan aldıkları aynı veriyi eş zamanlı olarak kontrol birimlerinden aldıkları farklı komutlar ile işlerler. Bu, komut seviyesindeki paralelleştirmeye örnektir.

Çoklu komut çoklu veri (ÇKÇV) : Bu bilgisayar yapısı en genel ve diğerlerine göre daha güçlü bir yapıdır. Birden çok işlemci, birden çok komutu birden çok veri üzerinde çalıştırabilir. Her işlemcinin kendine ait kontrol ve hafıza biriminin bulunması bu bilgisayar modelini kuvvetli hale getirir. Bu bilgisayar mimarisi kişisel bilgisayarlara, süper bilgisayarlara ve bilgisayar çalışma ağlarına uygulanabilir.

2.5.4. Paralel bilgisayar hafıza mimarisi

Paralel programlamada bir diğer önemli konu hafıza yapısı ve kullanımıdır. Paylaşımlı hafıza, dağıtımlı hafıza ve karma hafıza olmak üzere başlıca üç tür bellek yapısından bahsedilebilir. Paylaşımlı hafıza yapısında işlemciler ortak bir belleği kullanırlar (Şekil 2.11 :a). Dağıtımlı hafıza sistemlerinde her işlemci kendi belleğini kullanır. Karma hafıza da ise bu iki yapının beraber kullanılması durumudur. Örnek vererek açıklamak istenirse, çok çekirdekli bir bilgisayarda işlemci çekirdekleri ortak bir hafızayı paylaşır. Böyle bir sistemde çekirdekler paylaşımlı hafıza kullanmış olur. Tek çekirdeği olan birden fazla bilgisayar ile paralel programlama yapılmak istenirse burada dağıtımlı hafıza kullanımı söz konusudur (Şekil 2.11 :b). Karma hafıza yapısı ise birden fazla çekirdeği olan birden fazla bilgisayarın kullanılması durumudur.

Dağıtımlı hafıza yapılarında bir işlemci başka bir işlemcinin belleğindeki veriye mesaj yolu ile ulaşır. Bu noktada mesajlaşmak için kullanılabilecek hali hazırda birçok protokol ve araç vardır. Bunların içinde en çok kullanılanlardan ikisi MPI (Message Passing Interface) ve PVC (Parallel Virtual Machine)'dir. Paylaşımlı hafıza yapılarında da paralel programlama için geliştirilmiş OpenMP gibi uygulama arayüzleri mevcuttur.

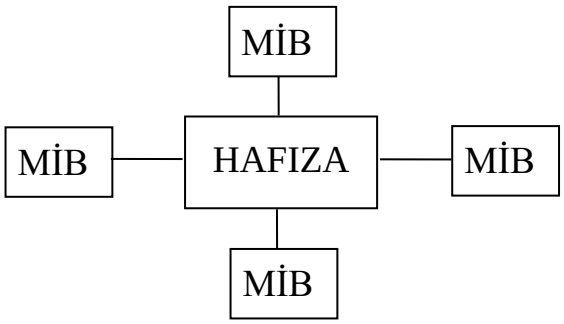


Şekil 2.11 : Bellek yapısı: (a) Paylaşımlı bellek; (b) Dağınık bellek.

2.5.4.1. Paylaşımlı hafıza

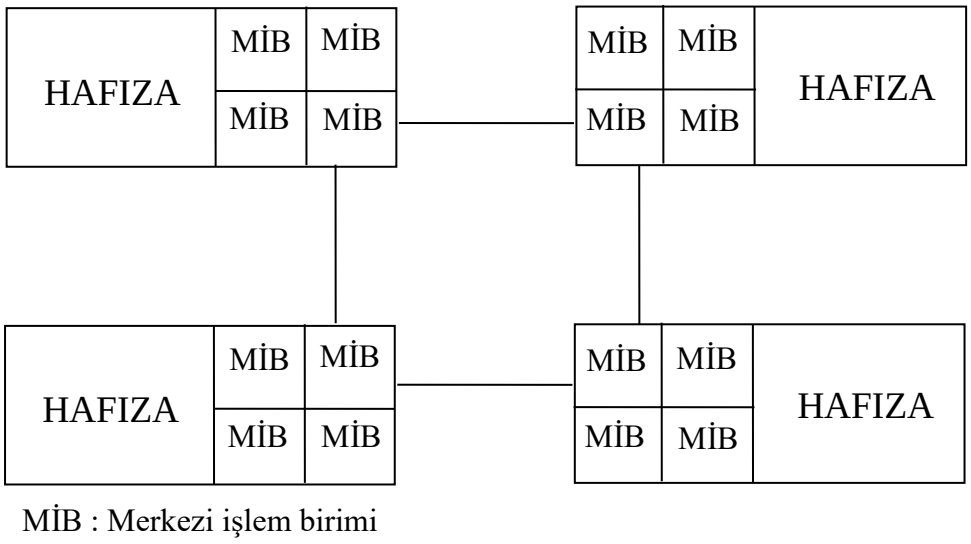
Paylaşımlı bellek türleri paralel bilgisayarlarda mevcuttur. Bütün çekirdeklerin hafızaya ulaşım imkanları vardır. Her işlemci kendi işlemini yapar ve diğer işlemcilerinde erişebileceği belleği kullanarak girdi/çıktı verilerini depolar. Bir çekirdeğin hafıza üzerinde yaptığı değişikliği diğer çekirdeklerde görebilir. Bilgisayarlar hafızaya ulaşım hızlarına göre düzenli hafıza erişim ve düzensiz hafıza erişimi olarak 2 ye ayrılırlar.

Düzenli hafıza erişiminin özelliği, belleğe erişim hızları eşittir ve aynı miktarda erişirler (bknz Şekil 2.12).



Şekil 2.12 : Düzenli paylaşımlı hafıza.

Düzensiz hafıza erişiminin özelliği ise, bütün çekirdekler her belleğe aynı sürede erişemez (bknz Şekil 2.13).

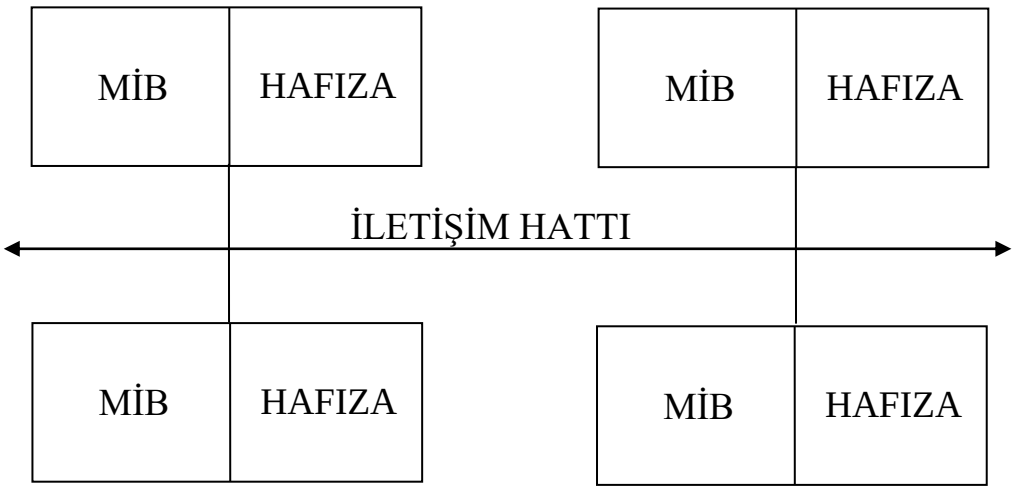


Şekil 2.13 : Düzensiz paylaşımlı hafıza.

2.5.4.2. Dağıtımlı hafıza

Dağıtımlı hafızaların diğer hafıza türünden farkı bellekler arasında bir iletişim hattının olmasıdır (bknz Şekil 2.14). Her çekirdeğin kendi hafızası vardır, bu hafızalarda değişiklik yapar ve bu değişiklik diğer çekirdekler tarafından farkedilmez.

Bu hafıza çeşitinin bazı avantajları vardır. Çekirdek sayısının artması hafıza miktarınını artması anlamına gelir. Her çekirdek kendi lokal hafızasına hızlı bir şekilde erişir dolayısıyla ağ iletişim için vakit kaybı oluşmaz.



Şekil 2.14 : Dağıtımlı hafıza.

2.6. Paralel Programlamanın İnşaat Mühendisliğinde Uygulamaları

Paralel programlama inşaat mühendisliği problemlerinde genellikle yapı sistemi analizinde doğrusal denklem takımı çözümü için kullanılmaktadır. Doğrusal denklem takımı çözümü birçok mühendislik alanında karşılaşılan ortak bir problem olduğundan dolayı özellikle matematik ve bilgisayar alanında çalışan araştırmacılar bu ortak problemin çözümüne yönelik, doğrusal denklem takımı çözüm sürelerini paralel programlama ile kısaltmışlardır (Intel, 2017; NAG, 2017). Bu yöntemlerde ana amaç büyük denklem takımını daha küçük takımlar ile ifade edip işlemcileri eş zamanlı paralel bir biçimde kullanarak toplam denklem takımının çözüm süresini kısaltmaktır.

İnşaat mühendisliğinde uygulamalar için bazı örnekler verilebilir. Abrahamson ve Shedlock (1997) tarafından yapılan çalışmada nesne tabanlı programlama ile doğrusal olmayan analiz, alan ayrıştırması (domain decomposition) yöntemi ile paralelleştirilmiştir. Bu çalışma OpenSees programının Mckenna (1997) paralel işleme alt yapısını oluşturmuştur "OpenSeesSP" ve "OpenSeesMP" olmak üzere iki farklı paralel derleyici vardır. "OpenseesSP"' çok büyük modellerin analizlerinde, "OpenseesMP" birden fazla küçük ölçekte birçok modelin paralel analizinde kullanılmaktadır (McKenna ve Fenves, 2008). Diğer bir çalışmada Dizon (2016) tarafından geliştirilen "FRAME3D" çelik yapı doğrusal olmayan analiz programının 2016 sürümünde hibrit paralel programlama çözümü uygulanmıştır. Bu yöntemlerde ana amaç büyük denklem takımını daha küçük takımlar ile ifade edip işlemcileri eş zamanlı kullanarak toplam denklem takımının çözüm süresini kısaltmaktır.

3. MATEMATİKSEL İFADELER VE TEORİLER

3.1. Bünye Fonksiyonları

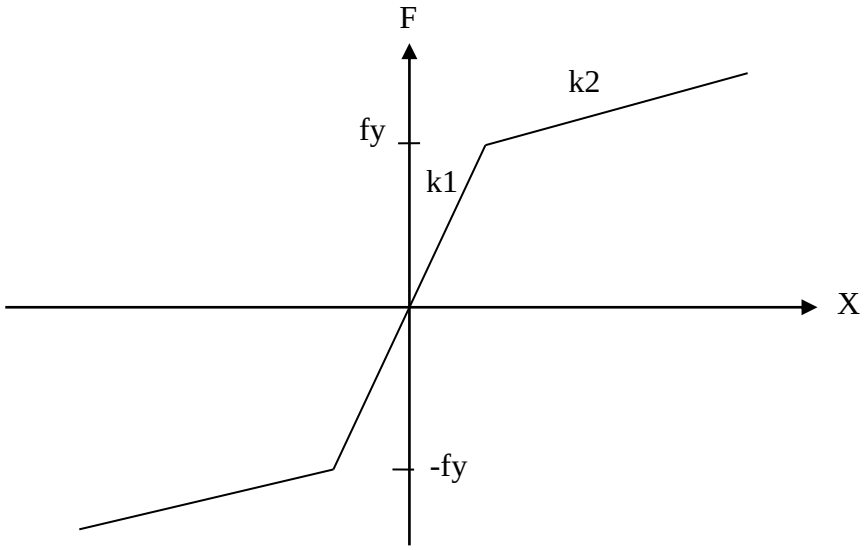
Doğrusal elastik modeller literatürde mevcut bulunan en basit bünye fonksiyonlarıdır. Doğrusal elastik modellerde, beton nihai dayanıma ulaşana kadar doğrusal elastik olarak davranır ve sonradan gevrek davranışla kırılma gösterir. Malzemelerin yükleme altında yaptıkları buna benzer deformasyonu gösteren bağıntılar malzeme bünye fonksiyonu olarak bilinir

Doğrusal olmayan davranışı temsil edebilmek için kullanılan bazı matematiksel modeller mevcuttur. Çift doğrusal (bilineer), takeda, bouc-wen bunlardan bazılarıdır.

Bu modeller, elemanın bulunduğu konumdan belirli bir deplasman yapması sonucu oluşabilecek kuvvet farkını hesaplayabilme kabiliyetine sahiptirler. Malzemenin gerçek davranışını yansıtabilecek şekilde idealleştirildikten sonra kullanılabilmektedirler. Bunlardan en yaygın ve basit seviyede olan model çift doğrusaldır.

3.1.1. Çift doğrusal model

Çift doğrusal eleman modelinde, malzemenin akma değerine kadar sabit bir rijitlikle ilerlediği aktıktan sonra ise pekleşme yaptığnı ifade eden ilk rijitlikten daha düşük sabit bir rijitlikle davranışına devam ettiği düşünülmektedir. Deplasmanlar pozitif ve negatif yönde olduğu takdirde çevrimsel davranışta sergileyebilmektedir (bknz Şekil 3.1).



Şekil 3.1 : Çift doğrusal model.

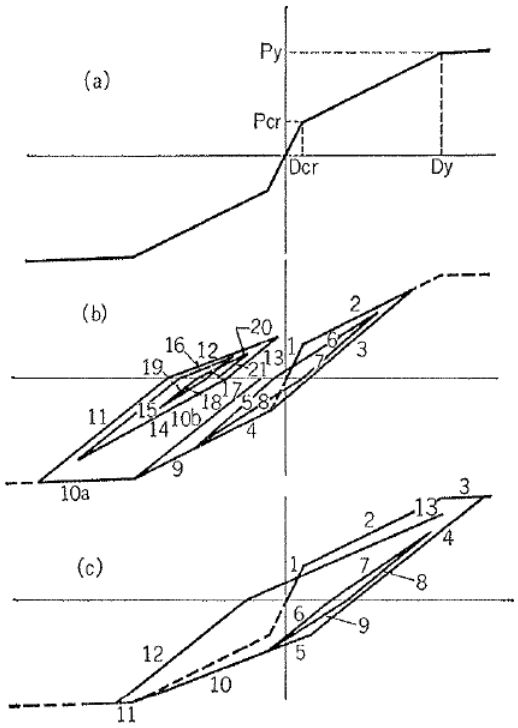
fy : akma kuvveti

k1 : elastik rijitlik

k2 : inelastik rijitlik (pekleşme)

3.1.2. Takeda model

1970 yılında, Takeda sistemin maksimum yerdeğiştirmesine bağlı olarak üç parçalı lineer iskelet eğrisine sahip bir model geliştirdi (Takeda ve diğ., 1970).



Şekil 3.2 : Kuvvet – Yerdeğiştirme eğrisi (Takeda ve diğ., 1970).

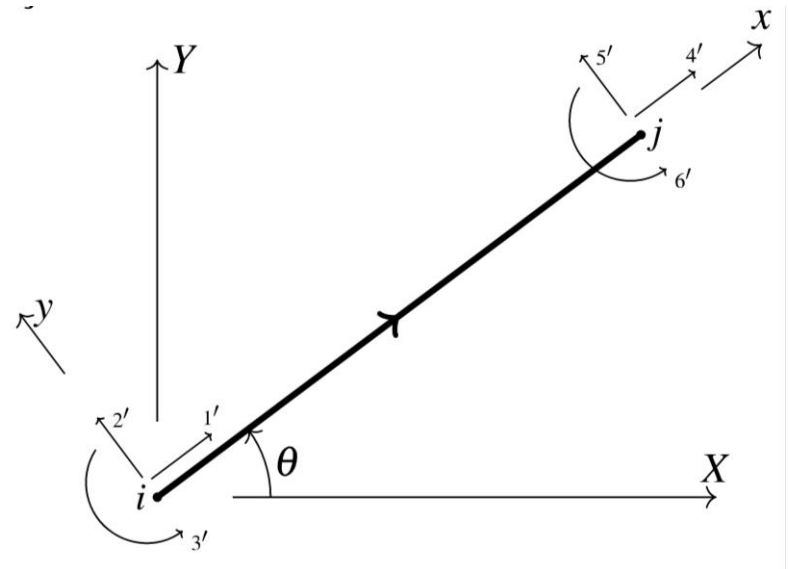
3.2. Yapısal Sistem Modeli

Gerçek hayatta yapılar, kendi ağılıklarına, hareketli yüklere ve farklı düzlemlerde ki karmaşık depremsel zorlamalara maruz kalan üç boyutlu karmaşık sistemlerdir. Ciddi düzensizlikleri olan yapısal sistemler için üç boyutlu modeller kritik bir konu olmaktadır. Bu tip yapısal sistemlerin basitleştirilmiş iki boyutlu modelleri yapıda oluşacak depremsel davranışları yakalamakta yeterli olmamaktadır. Düzenli yapılarda üç boyutlu yapısal sistemin özellikleri, gerçek yapının yerdeğiştirme, dayanım gibi genel davranışlarını yeterince yakalayabilecek basitleştirilmiş iki boyutlu sisteme yoğunlaştırılabilir.

Bu tez çalışması kapsamında gözönünde bulundurulacak yapısal sistem çubuk elemanlardan ve dönme yaylarından oluşan bir çerçeve sistemdir.

3.2.1. Çubuk elemanlar

Yapısal sistem modellemesi esnasında; kolon, kiriş gibi yapısal elemanlar, kenarları ve yüzeyleri birleştiren düğüm noktaları ve mesnetleme olarak karşımıza çıkan sınır koşulları gibi konular gözönünde bulundurulurlar. Yapısal sistemi ifade eden kütle, rijitlik ve sönümleme terimleri, düğüm noktalarına toplanır ve yığılı oldukları düşünülür. Kütle, atalet ve yükler yapısal elemanlara uygulanır ve sonrasında yapısal elemanları diğer yapısal elemanlara bağlayan düğüm noktalarına nakledilir. Her düğüm noktasının ötelemelerini ve dönmelerini gösterin serbestlik dereceleri vardır (bknz Şekil 3.3). Şekil 3.3'de global eksene göre saat yönünün tersinde θ açısı kadar dönmüş bir çubuk eleman görünmektedir.



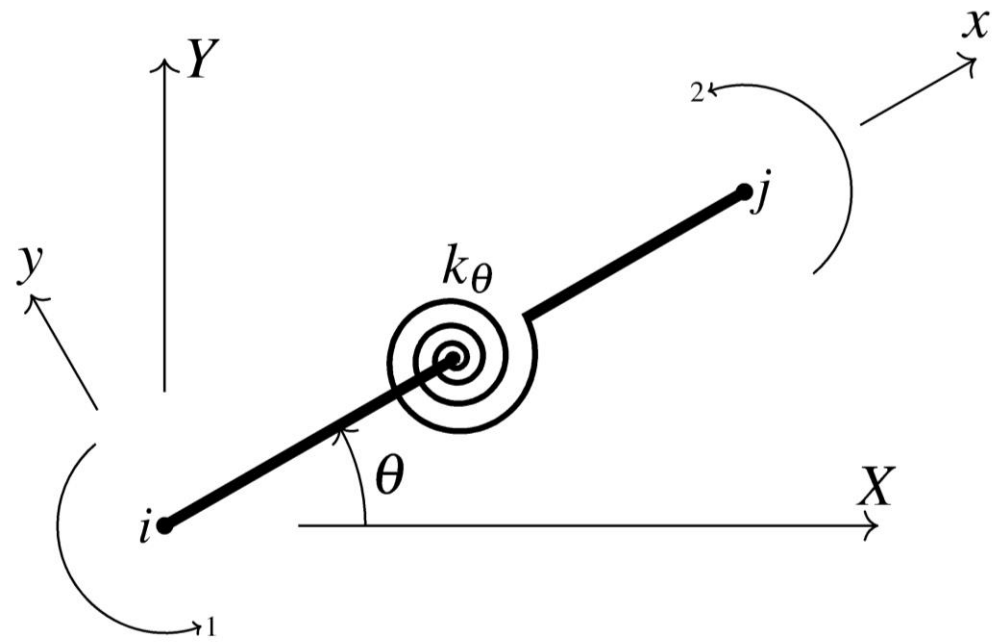
Şekil 3.3 : Çubuk eleman serbestlik dereceleri.

3.2.2. Yay elemanlar

Betonarme kolon ve kirişin doğrusal olmayan davranışını yakalayabilmek için yay modeller kullanılır. Elemanların doğrusalsızlığı tek bir noktaya yoğunlaştırılmış ve bu noktaya yay eleman koyarak oradaki doğrusal olmayan davranış temsil edilmiş olur. Yığılı mafsal modeller, basit yapılı olmalarına rağmen dayanım bozulması etkilerini yansıtabilirler. Yığılı mafsal modeller halihazırda kullanılan yönetmelik ve standartlardaki limit durumlara oldukça uygundur.

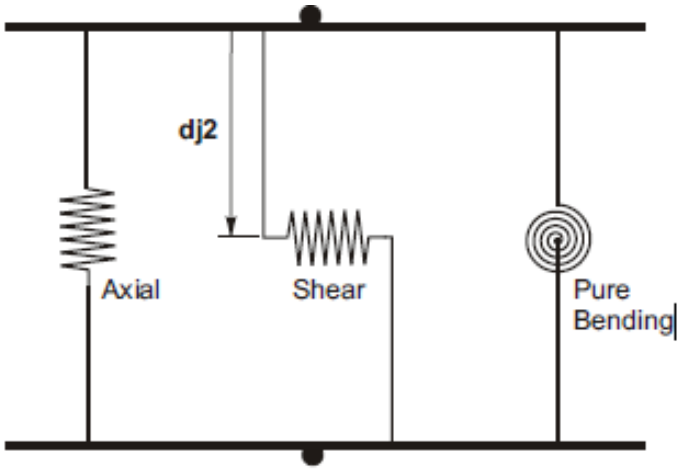
Yay elemanlar rijit-plastik ve elasto-plastik olabilirler. Bu çalışma kapsamında elasto-plastik yay tercih edilmiştir. Bu elasto-plastik model içinde iki parçalı doğrusal yay kabulü yapılmıştır. Yaylar ilk rijitlikle devam ederken akma kuvvetine kadar zorlanırlarsa ikinci rijitliğe geçerler ve pekleşme devreye girer.

Yay elemanlar, çubuk elemanlar ile düğüm noktaları sayesinde bağlanırlar ve kuvvet aktarımı bu düğüm noktalarında gerçekleşir. Yay elemanın serbestlik derecesi yansıtması istenen davranışa göre ayarlanabilir. Eksenel yaylar için eksenel serbestlik dereceleri, kesme yayları için kesme serbestlik dereceleri doğrusal olmayan davranışı yansıtır. Şekil 3.4 ‘ de dönme yayı için sadece dönme serbestlik derecesi tanımlanmıştır.



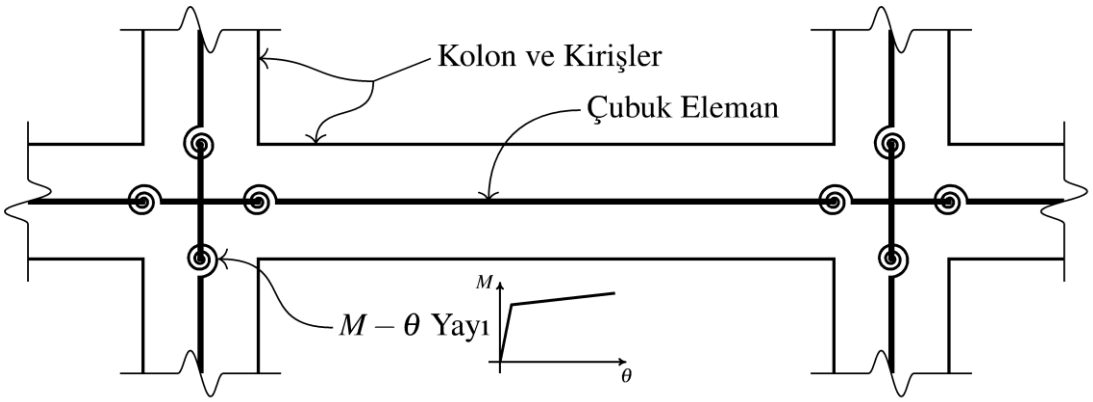
Şekil 3.4 : Dönme yayı elemanı serbestlik dereceleri.

Aynı zamanda eksenel, kesme ve dönme doğrusalsızlığının hepsini aynı anda inceleyen yaylarda tanımlanabilir (örnek sap2000 yay modeli).



Şekil 3.5 : Sap2000 programı yay modeli.

Yay modeller, genellikle kolon ve kirişlerde doğrusal olmayan davranışın en yoğun olacağı bölgelere yerleştirilirler. Dönme yayları kolon ve kiriş yüzeylerine eklenmiştir (bknz Şekil 3.6).Tanımlanan bünye fonksiyonuna göre yapının belirli deplasmanlarda üreteceği içsel kuvvetleri hesaplarlar.



Şekil 3.6 : Dönme yayı konumları.

3.2.3. Kütle Rijitlik ve Sönüm matrislerinin elde edilmesi

Yapısal sistem çubuk ve yay elemanlar ile oluşturulduktan sonra analiz için yapıya ait fiziksel özellikler karakterize edilir. Yapının önde gelen fiziksel özelliği rijitliğidir. Yapıya kuvvetlere karşı koyabilme yeteneği kazandırır. Bir diğer özellik ise yapının kütle bilgisidir. Yapının sönümleme oranı da davranışı etkileyen başka bir faktördür.

Dinamik analizlerde, yapı kütlesi atalet kuvvetlerini hesaplamak için kullanılır. Malzemenin kütle yoğunluğundan ve hacminden faydalanarak kütle elde edilir. Herbir düğüm noktasında kendisine bağlanan elemanların kütleleri toplanır ve o düğüm noktası için yığılı bir kütle elde edilmiş olur. Bu kütle öteleme serbestlik

dereceleri için kullanılır. Dönme serbestlik derecesi için ise dönme ataleti gözönünde bulundurulur. Kütle matrisi ortaganal bir şekilde oluşturulup bütün serbestlik dereceleri için hesaplanır (Şekil 3.7).

$$M=\begin{bmatrix} m_1 & & & & & \\ & m_2 & & & & \\ & & m_3 & & & \\ & & & \bullet & & \\ & & & & \bullet & \\ & & & & & \bullet \\ & & & & & & m_{n-1} & \\ & & & & & & & m_n \end{bmatrix}$$

Şekil 3.7 : Tipik kütle matrisi.

Rijitlik matrisi, elemanların kuvvet ve yerdeğiştirme ilişkilerini hesaplamak için kullanılır. Rijitlik katsayıları ile her çubuk eleman için 6 x 6 lık lokal rijitlik matrisi elde edilir. Bu rijitlik matrisi geçiş matrisleri yardımıyla global koordinat sistemine çevrilebilir. Çubuk uç kuvvetleri, uç deplasmanları, rijitlik matrisi ve dönüşüm matrisleri verilmiştir (Şekil 3.8).

$$\mathbf{d}'=\begin{Bmatrix} d_1' \\ d_2' \\ d_3' \\ d_4' \\ d_5' \\ d_6' \end{Bmatrix} \qquad \mathbf{f}'=\begin{Bmatrix} f_1' \\ f_2' \\ f_3' \\ f_4' \\ f_5' \\ f_6' \end{Bmatrix} \qquad \mathbf{k}'=\begin{bmatrix} \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}$$

$$\mathbf{f}'=\mathbf{k}'\mathbf{d}' \qquad \mathbf{d}'=\mathbf{T}\mathbf{d} \qquad \mathbf{k}=\mathbf{T}^{\mathrm{T}}\mathbf{k}'\mathbf{T} \qquad \mathbf{f}=\mathbf{T}^{\mathrm{T}}\mathbf{f}' \qquad \mathbf{f}=\mathbf{k}\mathbf{d}$$

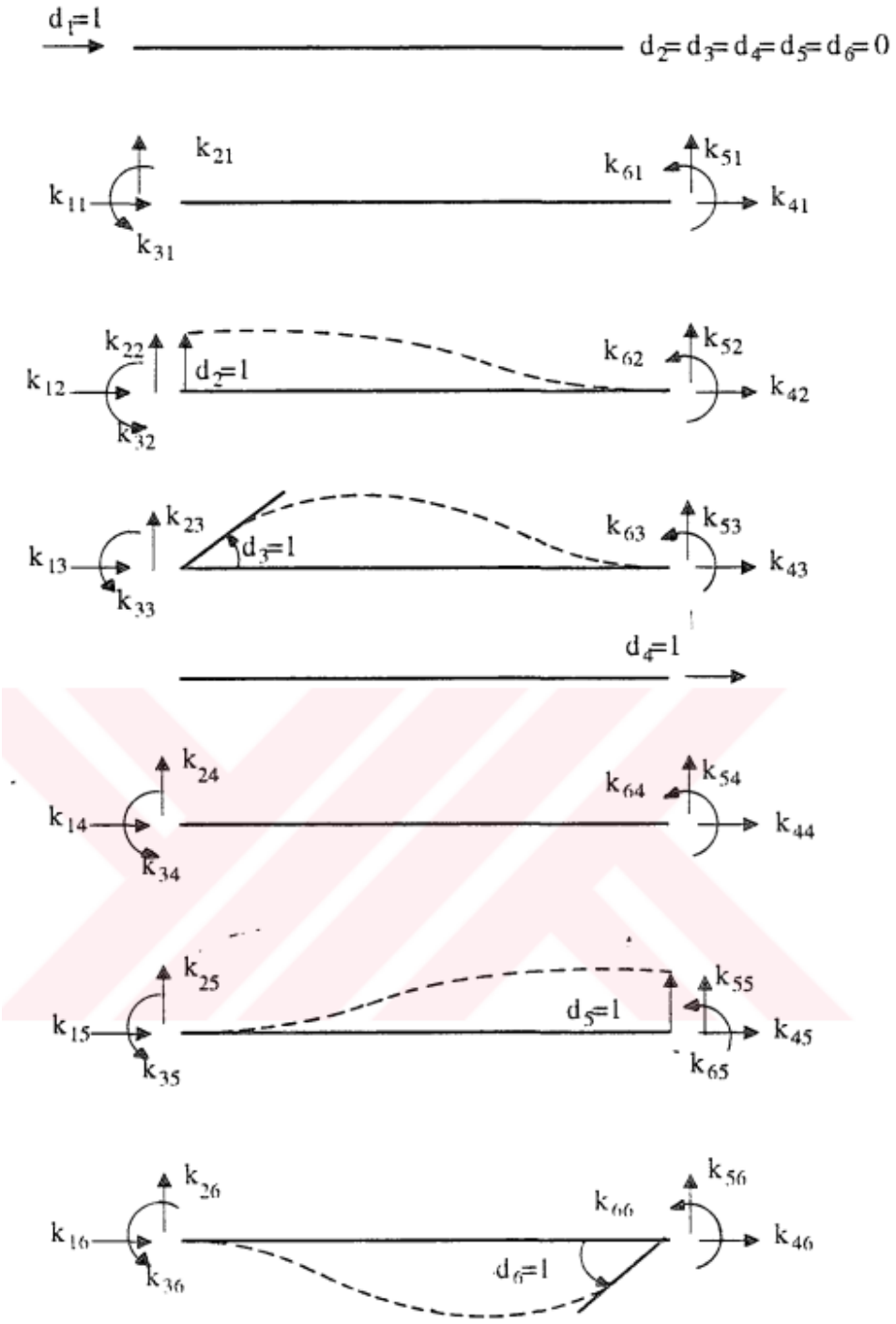
$$\mathbf{T}=\begin{bmatrix} \overline{\mathbf{T}} & \mathbf{0} \\ \mathbf{0} & \overline{\mathbf{T}} \end{bmatrix} \qquad \overline{\mathbf{T}}=\begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{d}',\mathbf{f}',\mathbf{k}'$: Lokal koordinat sistemindeki değerler

$\mathbf{d},\mathbf{f},\mathbf{k}$: Global koordinat sistemindeki değerler

Şekil 3.8 : Çubuk eleman matrisleri.

Düğüm noktasındaki birim yerdeğiştirme için gerekli olan o doğrultudaki kuvvete rijitlik denir. Şekil 3.3'de tarif edilen her bir serbestlik derecesi yönünde birim yerdeğiştirme uygulanarak ve diğer bütün yerdeğiştirmelerin olduğu bu durama ait elastik eğrinin formunu koruyabilecek kuvvetler hesap edilir (bknz Şekil 3.9). Bu sayede rijitlik katsayıları her serbestlik derecesi için elde edilir (Karaduman, 1993).



Şekil 3.9 : Rijitlik katsayıları (Karaduman, 1993).

Rijitlik katsayıları çubuk elemanlar gibi yay elemanları içinde elde edilirler. Üç serbestlik derecesi içinde hesaplanabileceği gibi sadece bir serbestlik derecesi içinde hesaplanıp matris formuna getirilebilir. Dönme yayına ait kuvvet, deplasman ve rijitlik matrisi verilmiştir (Şekil 3.10)

$$\mathbf{d} = \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} \qquad \mathbf{f} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \qquad \mathbf{k} = \begin{bmatrix} k_\theta & -k_\theta \\ -k_\theta & k_\theta \end{bmatrix}$$

$$\mathbf{f} = \mathbf{kd} \qquad \mathbf{d}, \mathbf{f}, \mathbf{k} : \text{ Global koordinat sistemindeki değerler}$$

Şekil 3.10 : Dönme yayı eleman matrisleri.

Sönümleme, malzeme ve yapı özelliklerine göre dinamik davranışı etkiler. Sönümleme farklı yükleme durumları için değişiklik gösterebilir. Response spectrum ve modal zaman tanım aralığı analizleri için modal sönümleme kullanılırken direk zaman aralığı integrasyonu analizinde viskoz oran sönümü kullanılır.

Rayleigh sönümleme (Rayleigh, 1896), kütle ve rijitlikle orantılı sönümleme olarak bilinir ve genellikle doğrusal olmayan dinamik analizlerde kullanılır. Sönümlemenin kütle ve rijitlik matrisleri ile orantılı bir şekilde oluştuğu varsayılır (bknz Denklem 3.1).

$$C=\alpha M+\beta K \tag{3.1}$$

Burada

α : kütle orantı sönüm katsayısı

β : rijitlik orantı sönüm katsayısı

Modal denklemlerin ortogonalite özelliği sayesin kritik sönüm oranı; doğal frekans,kütle ve rijitlik orantı sönüm katsayısına bağlı olarak hesaplanır (bknz Denklem 3.2).

$$\xi_n = \frac{1}{2\omega_n}\alpha + \frac{\omega_n}{2}\beta \tag{3.2}$$

Burada

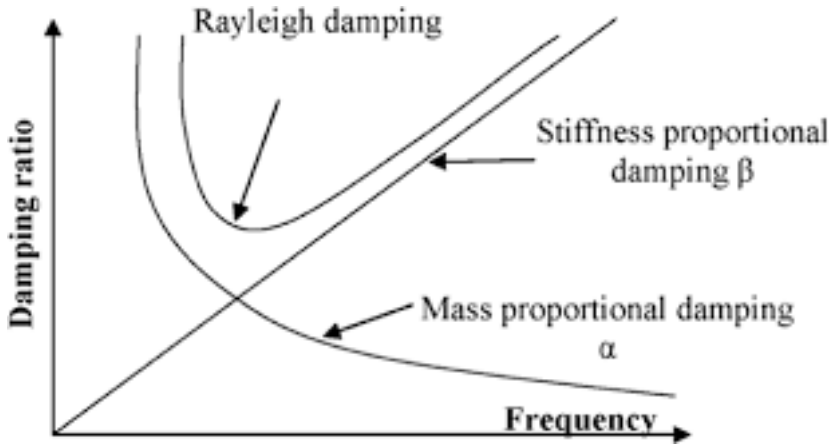
ξ_n : kritik sönümleme oranı

ω_n : doğal frekans

Belirlenmiş iki frekans ve rayleigh katsayıları biliniyor ise onlara karşılık gelen sönüm oranları elde edilebilir (bknz Denklem 3.3).

$$\begin{bmatrix} \xi_i \\ \xi_j \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \frac{1}{\omega_i} & \omega_i \\ \frac{1}{\omega_j} & \omega_j \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{3.3}$$

Rayleigh sönüm oranı, rijitliğin frekansla doğrusal artması ve kütlenin ters orantı ile değişmesi ile grafiksel olarak elde edilir (Şekil 3.11). Bulunan rijitlik ve kütle orantısı birleştirildiğinde rayleigh sönümü ortaya çıkar.



Şekil 3.11 : Rayleigh sönümleme grafiği.

3.2.4. Çerçeve Sistem

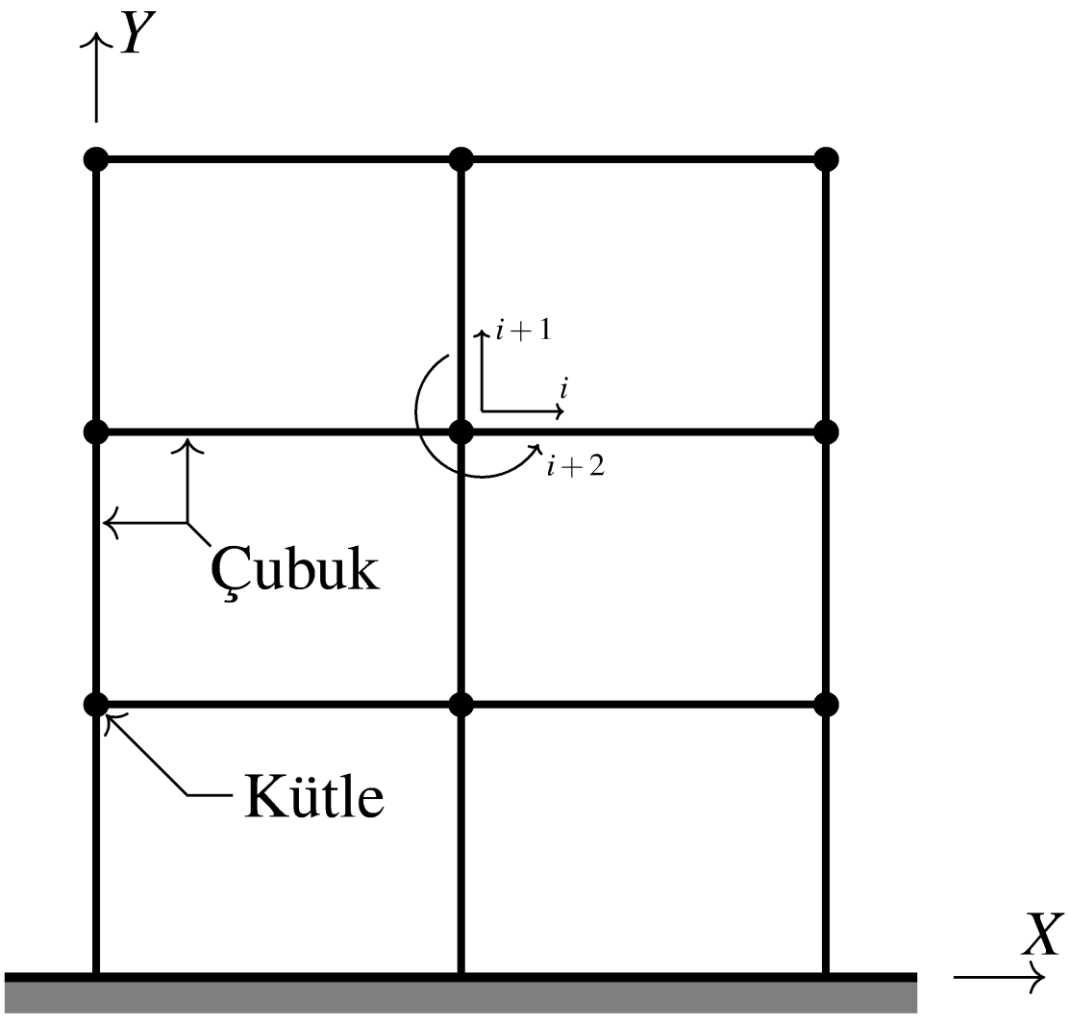
Çerçeve sistemler en genel, belki de en çok kullanılan bina çeşitidir. Düşey elemanlar kolon, yatay elemanlar kiriş olarak adlandırılırlar. İnsanların yürüdüğü düzlem ise döşeme olarak isimlendirilir. Kolonlar birincil yük taşıyıcı elemanlar olduğu için hayati önem taşımaktadırlar. Eğer kiriş veya döşeme zarar görürse çoğunlukla bulunduğu kat zarara uğrar fakat kolonun hasar alması bütün binayı etkileyecek bir durumdur.

Çerçeve sistemler, yanal ve yerçekimi kuvvetlerine karşı koyabilen kolon, kiriş, döşeme ve perde duvar gibi elemanların birleşmesiyle oluşan yapılardır. Bu sistemler genellikle uygulanan yüklerin oluşturduğu büyük momentlere karşı koyabildikleri için tercih edilirler. Rijit çerçeve sistem ve gergili çerçeve sistem olarak ikiye

ayrılabilirler. Rijitlik yerdeğiştirmelere karşı koyabilme yeteneği olarak düşünülebilir ki rijit çerçeve sistemler kolon ve kirişlerin yapımıyla oluşan ve yüklerin üreteceği momente beraberce direnebilen yapılar olarak tanımlanabilir. Rijit çerçeve sistemler daha fazla kararlılık sağlarlar. Rijit çerçeve sistemler diğer çerçeve sistemlerden daha etkili bir şekilde kesme, moment ve burulmaya direnç gösterebilir.

Çerçeve sistemlerin imalatlarının kolay olması, hızlı bir şekilde inşa edilebilmesi önemli avantajlarındandır. Bunun yanında bina tasarımlarında ekonomi önemli bir faktör olduğu için çerçeve sistemler tercih edilmektedir.

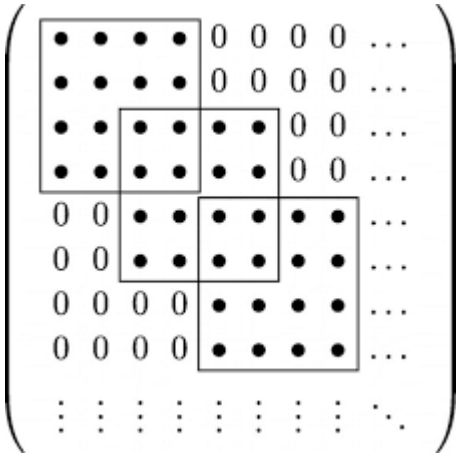
Çerçeve sistemi oluşturan kiriş ve kolon elemanlar basitleştirme yapılarak çubuk elemana dönüştürülebilir ve bu çubuk elemanlar düğüm noktalarında birleştirilir (bknz Şekil 3.12). Düğüm noktalarında normal kuvvet, kesme kuvveti ve moment dengesi sağlanır.



Şekil 3.12 : İki boyutlu çerçeve sistem.

Eleman bazında elde edilen kütle, rijitlik ve sönüm matrisi sistemin matematiksel modelini oluşturacak şekilde genel matrislere toplanırlar. Serbestlik dereceleri kullanılarak yapının genel kütle ve rijitlik matrisi daha büyük matrislerde depolanır. Bir düğüm noktasına bağlanan birden fazla çubuk eleman için o düğüm noktasının serbestlik dereceleri geçerli olacaktır. Dolayısıyla belirlen serbestlik derecesinin kütle ve rijitliğine alakalı bütün çubuk elemanlar katkıda bulunacaklardır (bknz Şekil 3. 13).

Sistemin üç boyutlu olduğu düşünülürse her düğüm noktasında 6 tane dolayısıyla her çubuk eleman için 12 tane serbestlik derecesi tanımlanır. Eğer sistem iki boyutlu bir çerçeve sistem ise her düğüm noktasında iki öteleme bir dönme serbestlik derecesi olacak ve her bir çubuk elemanında 6 tane serbestlik derecesi ortaya çıkacaktır. Yapısal sistemin genel serbestlik derecesi düğüm noktasına bağlı olarak artacaktır. Genel kütle, rijitlik ve sönüm matrislerin boyutu ise toplam serbestlik derecesi kadar olacaktır.



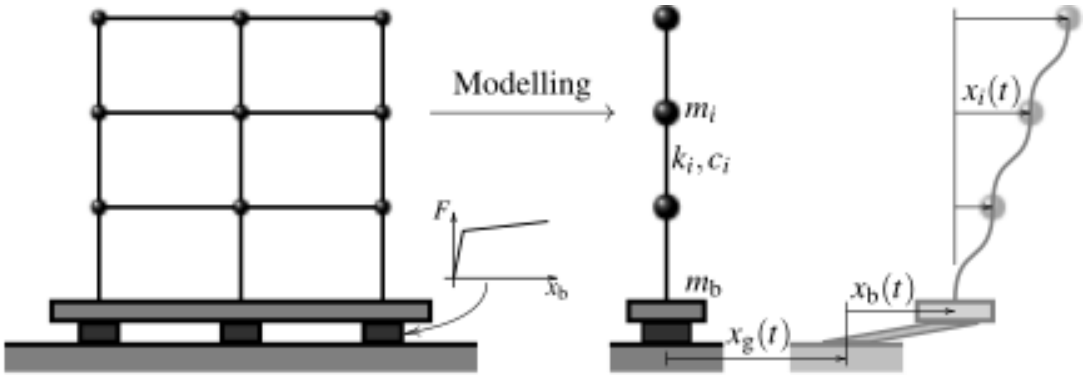
Şekil 3.13 : Genel matrisleri oluşturma yaklaşımı.

3.2.5. İzolatörlü Sistem

Günümüzde izolatör, yapıların depremden korunabilmesi için oldukça geniş bir şekilde uygulanan bir yöntemdir. İzolatörden beklenen temel davranış, depremin yol açtığı yer hareketi veya mesnet hareketlerini yapıdan ayırarak yapısal deformasyonları azaltmaktır. Depreme karşı izolatör ile yalıtılmış bir binada temel ve yapı arasında uygun sönümlemeye sahip esnek bir katman inşa edilir. Yapı zemin üstü katların oluşturduğu üst yapı ve izolasyon içeren temel olarak ikiye ayrılabilir. Üst yapının doğrusal ve kat döşemelerinin rijit olduğu kabul edilir. Herbir katın davranışı bir noktaya yoğunlaştırılmış şekilde modellenir ve üç serbestlik derecesiyle

ifade edilebilir. Temel, kütle merkezine konumlandırılmış üç serbestlik derecesiyle ifade edilir. Üst yapıdan kaynaklanan kesme kuvveti, yer hareketinin sebeb olduğu atalet kuvveti ve izolatör kuvvetleri gözönünde bulundurulur. Kuvvetler temelin kütle merkezine aktarıldıklarında ötelenme ve dönme davranışı oluşturabilirler. Mesnetler doğrusal veya doğrusal olmayan davranış gösterebilirler. Doğrusal mesnetlerde doğrusal rijitlik ve sönümleme beklenirken izolatör gibi doğrusal olmayan mesnetlerden doğrusal olmayan izolatör kuvvetleri oluşturması beklenir.

Şekil 3.14' de izolatörlü bir sistem, basitleştirilmiş eşdeğer sistemi ve hareketi gösterilmiştir.



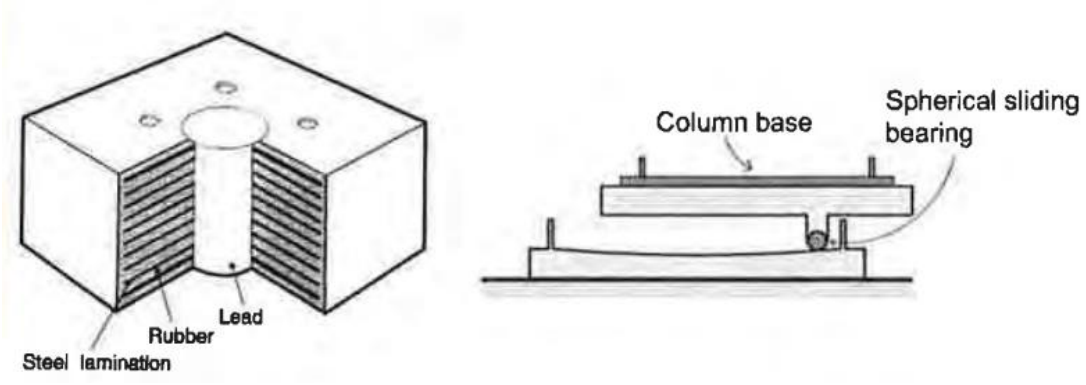
Şekil 3.14 : İzolatörlü çerçeve sistem.

İzolatörde oluşacak kuvvetlerin doğrusal olmayan aşamaya geçmesi ile yapıda enerji sönümlenmeye başlar. Sönümleyicisiz bir yapı enerjiyi yapısal elemanlarda oluşan doğrusalsızlık ile sönümler. Gelen enerjinin miktarına bağlı olarak yapının göstereceği davranış değişecektir. Artan deplasman değerleri yapıda büyük hasarlara sebeb olur. İzolatörler normalde yapı tarafından sönümlenecek enerjinin bir kısmını sönümleyerek yapıya giden enerjiyi azaltır. Dolayısıyla yapının yerdeğiştirmeleri ve oluşacak hasar azalır.

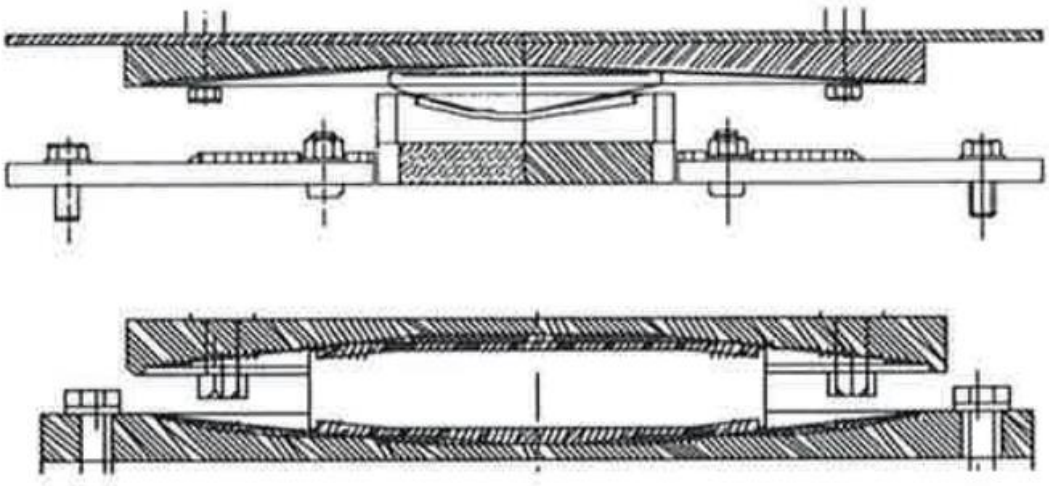
İzolatörlü yapılarda üst yapıya gelen kuvvetlerin azalacağı farkedilebilir. Yapısal tasarımlarda bu kuvvetlere göre yapılacaktır.

İzolatörler farklı yollar ile enerji sönümleyebilirler. Hıza bağlı olarak viskoz sönümleme, yerdeğiştirmelere bağlı olan ise akma/çevrimsel ve sürtünme mekanizmalarının oluşturduğu sönümlemelerdir. Bu iki izolatörün hesap esasları birbirine göre farklılık gösterebilir.

İzolatör çeşitlerinde en yaygın olarak kullanılan iki tanesi: kavuçuk izolatörler ve sürtünmeye dayalı izolatörlerdir. Bu izolatör çeşitleriyle alakalı görseller Şekil 3.15 ve Şekil 3.16'da verilmiştir (University of Canterbury Research, 2011).



Şekil 3.15 : Kavuçuk izolatörler.



Şekil 3.16 : Sürtünmeye dayalı izolatörler.

3.3. Yapı Analiz Çeşitleri

Yapısal analiz, fiziksel kanunları ve matematiksel denklemleri kullanarak yapısal sistemin davranışını tahmin etmek için sistemin çözümlenmesi sürecidir. Yapısal analizin amacı, farklı yükleme durumları için yapının içsel kuvvetleri, gerilmeleri ve yerdeğiştirmelerini belirlemektir.

3.3.1. Doğrusal statik analiz

Tüm yapının doğrusal olması durumunda kuvvet, rijitlik ve yerdeğiştirme matrislerini içeren statik denge denklemleri yerdeğiştirme yöntemleri ile elde edilebilir. Bir yapının doğrusal statik analizi doğrusal denklem sistemlerinin çözümünü içerir (bknz Şekil 3.17).

$$\mathbf{F} = \mathbf{KD} \quad \begin{bmatrix} \mathbf{F}_A \\ \mathbf{F}_B \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{bmatrix} \begin{bmatrix} \mathbf{D}_A \\ \mathbf{D}_B \end{bmatrix}$$

$\mathbf{D}_A$: Hareketli Serbestlik Dereceleri

$\mathbf{D}_B$: Tutulu/Mesnetli Serbestlik Dereceleri

Şekil 3.17 : Yapı Matrisleri (Statik Analizler).

Bu analizde yapının düğüm noktalarına gelen kuvvetler dış yük olarak düşünülür. Yapının serbestlik derecelerine göre rijitlik matrisi oluşturulur. Dış kuvvetlere karşı koyabilecek içsel kuvvetler yapının rijitliği ve o doğrultudaki deplasmanına bağlıdır.

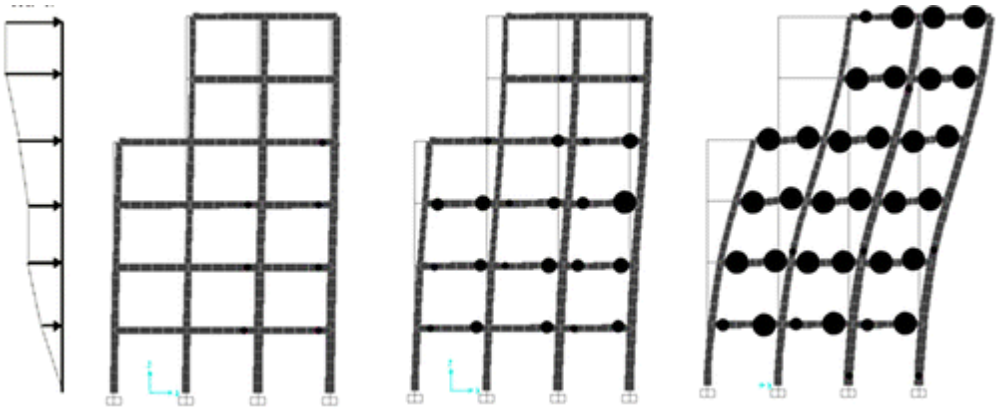
Denklem takımı çözülmeden önce yapıda tutulu olan serbestlik dereceleri belirlenir ve denklem takımının alt satırlarına gelecek şekilde tekrardan düzenlenir. Düzenleme sonucu serbest bulunan noktaların yerdeğiştirmeleri öncelikle hesaplanır. Daha sonra ise tutulu olan serbestlik derecelerinde oluşacak kuvvetler matris çarpımı ile elde edilir. Böylelikle yapıya dair kuvvet yerdeğiştirme ilişkisi çözülmüş olur.

Elemanlarda oluşacak içsel kuvvetler, elemanların rijitlik matrisleri ile elemanların yerdeğiştirmeleri çarpılarak hesaplanabilir. Bulunan normal, kesme ve moment kuvvetleri diyagramlar ile gösterilebilir.

3.3.2. Doğrusal olmayan statik analiz

Doğrusal olmayan statik analiz bir yapının malzeme ve geometrik doğrusalsızlığını ölçmek, sonraki analizler için P-delta rijitliğini oluşturmak, değişen malzeme davranışına göre yapıyı incelemek gibi bir çok amaç için kullanılabilir. Doğrusal olmayan analizlerde rijitlik ve yükleme yerdeğiştirmeye bağlı olarak değişiklik gösterebilir. Bu değişim denge denklemlerinin çözümünün yinelemeli olmasını gerektirebilir. Yükleme veya malzeme özelliklerindeki küçük değişiklikler doğrusal olmayan davranışlarda büyük değişiklere neden olabilir. Bu yüzden birçok farklı yükleme durumunu gözönünde bulundurmak oldukça önemlidir. Aynı zamanda yapı sisteminin özelliklerini değiştirecek etkiler üzerine hassaslık çalışması yapılabilir.

Artırımlı yükleme durumları için bir yapıda meydana gelebilecek yerdeğiştirmeler ilk başlarda doğrusal olsa da yüksek yükleme durumları için yapı doğrusal olmayan davranışa geçecektir. Mekanizma durumuna yaklaşıldığında yapının bir çok noktası akma noktasını geçmiş olacaktır (Şekil 3.18).



Şekil 3.18 : Artımlı yükler altında statik analiz.

Başlangıç koşulları, yükleme durumunun başlangıcındaki yapının durumunu tarif eder. Yerdeğiştirme ve hızları, içsel kuvvetleri ve gerilmeleri, dış kuvvetleri ve buna benzer parametreleri başlangıç koşulu olarak değerlendirilir. Statik analizler için başlangıç hız ve yerdeğiştirme değeri sıfır alınmalıdır. Öncelik doğrusal olmayan adımın verileri ile bir sonraki analiz yapmak istenirse önceki doğrusal olmayan analiz sonuçları başlangıç koşulu olarak alınır.

Yineleme, analizin herbir adımında ve alt adımında hedeflenen yakınsama kriterine ulaştığından emin olmak için kullanılır.Her adım için sabit rijitlikle yineleme başlayabilir. Eğer yakınsama gerçekleşmemiş ise Newton Raphson yinelemesi denenebilir. Newton Raphon yinelemesinde tanjant rijitliği kullanılır.

Doğrusal olmayan statik analiz, bir yapının nihai yükleme ve yerdeğiştirme kapasitesini belirlemek için yapılan itme analizi olarakta bilinmektedir. Yapıya yerel doğrusal olmayan eleman tanımlanır ve sonrasında plastik mafsallar göçme mekanizmasına ulaşana kadar veya plastik mafsallar için plastik deformasyon sınırları aşılana kadar yapı artan kuvvetler altında ittirilir. Bu itme analizi yapının kapasitesi ölçülebilir.

3.3.3. Doğrusal dinamik analiz

Malzemelerin doğrusal davrandığı yapılarda deprem gibi dinamik yüklerin etkimesiyle oluşacak yerdeğiştirmelerin belirlenmesi zaman tanım aralığında yapılan analizler ile mümkündür. Zaman tanım aralığındaki analizler, zamanla değişebilecek belirli bir yüke karşı bir yapının dinamik davranışını adım adım analiz etmek olarak düşünülebilir. Bu analiz doğrusal olabildiği gibi doğrusal olmayan bir şekilde de yapılabilir.

Deprem esnasında yapıda oluşacak atalet, sönüm ve yay kuvvetleri düğüm noktalarında dengelenmelidir. Bu kuvvetlerin dengesi ile deprem süresi boyunca yapı dinamik dengede hareketine devam edecektir.

Dinamik denge denkleminde statik denge denkleminden farklı olarak kütlelerden kaynaklanan atalet kuvvetleri ve sönümleme kuvvetleri vardır. Bunların göz önünde bulundurulduğu serbest cisim diyagramından dinamik denge denklemi çıkmaktadır.

3.3.3.1. Dinamik hareket denge denklemi

Dinamik denge denklemi şöyle yazılabilir.

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\ddot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) + \mathbf{F}_s(t) = \mathbf{P}(t) \tag{3.4}$$

Burada

x(t) : zemine göre röletif olan yerdeğiştirmeleri ve dönmeler

M **:** kütle matrisi

C **:** sönümleme matrisi

K **:** doğrusal elemanlardan gelen rijitlik matrisi

$\mathbf{F}_s(t)$: doğrusal olmayan eleman kuvvetleri

P(t) : dış kuvvet vektörü

3.3.3.2. Newmark beta metodu

Denklem 3.4'ün t anındaki hali ve $t + \Delta t$ anındaki halinden çıkarılırsa aşağıdaki denklem elde edilir:

$$\mathbf{M}\Delta\ddot{\mathbf{x}}^t + \mathbf{C}\Delta\ddot{\mathbf{x}}^t + \mathbf{K}\Delta\mathbf{x}^t + \Delta\mathbf{F}_s^t = \Delta\mathbf{P}^t \tag{3.5}$$

Burada $\Delta[\ ]^t$, büyüklüğün $t + \Delta t$ ve t anlarındaki değerleri arasındaki farkı göstermektedir. Hareket denklemi Newmark-β yöntemi ile aşağıda gösterilen artımsal ve cebirsel forma çevrilebilir:

$$\mathbf{A}\Delta\mathbf{x}^t + \Delta\mathbf{F}_s^t = \Delta\hat{\mathbf{P}}^t \tag{3.6}$$

$$\mathbf{A} = \frac{\beta}{\beta\Delta t^2}\mathbf{M} + \frac{\gamma}{\beta\Delta t}\mathbf{C} + \mathbf{K} \tag{3.7}$$

$$\Delta\hat{\mathbf{P}}^t = \Delta\mathbf{P}^t + \left(\frac{1}{\beta\Delta t}\mathbf{M} + \frac{\gamma}{\beta}\mathbf{C}\right)\dot{\mathbf{x}}^t + \left[\frac{1}{2\beta}\mathbf{M} + \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\mathbf{C}\right]\ddot{\mathbf{x}}^t \tag{3.8}$$

Burada, β ve γ Newmark parametreleridir. Analizde, her zaman adımında artımsal doğrusal olmayan kuvvet için bir kabul yapılır. Bu kabul, doğrusal olmayan elemanların tanjant rijitliği üzerinden olabilir. Bu durumda

$$\Delta\mathbf{F}_{\mathrm{s}}^{t,\mathrm{kabul}} = \mathbf{K}_{\mathrm{T}}\Delta\mathbf{x}^{t,\mathrm{kabul}} \tag{3.9}$$

$$\left(\mathbf{A} + \mathbf{K}_{\mathrm{T}}\right)\Delta\mathbf{x}^{t,\mathrm{kabul}} = \Delta\hat{\mathbf{P}}^t \tag{3.10}$$

olur.

3.3.3.3. Dengelenmemiş kuvvet düzeltme metodu

Bu tez kapsamında dengesiz kuvvet düzeltme yöntemi kullanılmıştır.

Kabul edilen yer değiştirme için, bünye fonksiyonlarından doğrusal kuvvet $\Delta\mathbf{F}^{t,\text{çiftdoğ}}$ hesaplanabilir. Bu durumda, *t* adımı için dengelenmemiş kuvvet şu şekilde olur:

$$\Delta\mathbf{F}_{\mathrm{s}}^{t,\mathrm{denge}} = \Delta\mathbf{F}_{\mathrm{s}}^{t,\mathrm{kabul}} - \Delta\mathbf{F}_{\mathrm{s}}^{t,\text{çiftdoğ}} \tag{3.11}$$

Dengelenmemiş kuvvet bir sonraki zaman adımında ek dış kuvvet olarak yapıya etkitilerek sistem çözülür:

$$\mathbf{A}\Delta\mathbf{x}^{t+1} + \Delta\mathbf{F}_{\mathrm{s}}^{t+1} = \Delta\hat{\mathbf{P}}^{t+1} + \Delta\mathbf{F}_{\mathrm{s}}^{t,\mathrm{denge}} \tag{3.12}$$

3.3.4. Doğrusal olmayan dinamik analiz

Deprem gibi dinamik etkilerin yapıyı zorlamasıyla malzemenin doğrusal olmayan evreye geçtiği analiz çeşitidir. Doğrusal olmayan dinamik analizlerde doğrusal dinamik analizler gibi zaman tanım aralığında adım adım çözülürler.

Denklem 3.4' de verilen dinamik hareket denge denklemi doğrusal olmayan dinamik analiz için Newmark β ve dengelenmemiş kuvvet düzeltme metodu ile adım adım çözülür.

Zaman tanım aralığı analiz çeşiti belirlemenin birkaç seçeneği vardır. Doğrusal veya doğrusal olmayan malzeme özelliği ile modellenebilir. Modal ve direkt integrasyon olmak üzere iki farklı çözüm metodu vardır. Direk integrasyon sonuçları zaman adımı boyutuna karşı fazlasıyla hassastırlar. Direk integrasyon analizleri programın

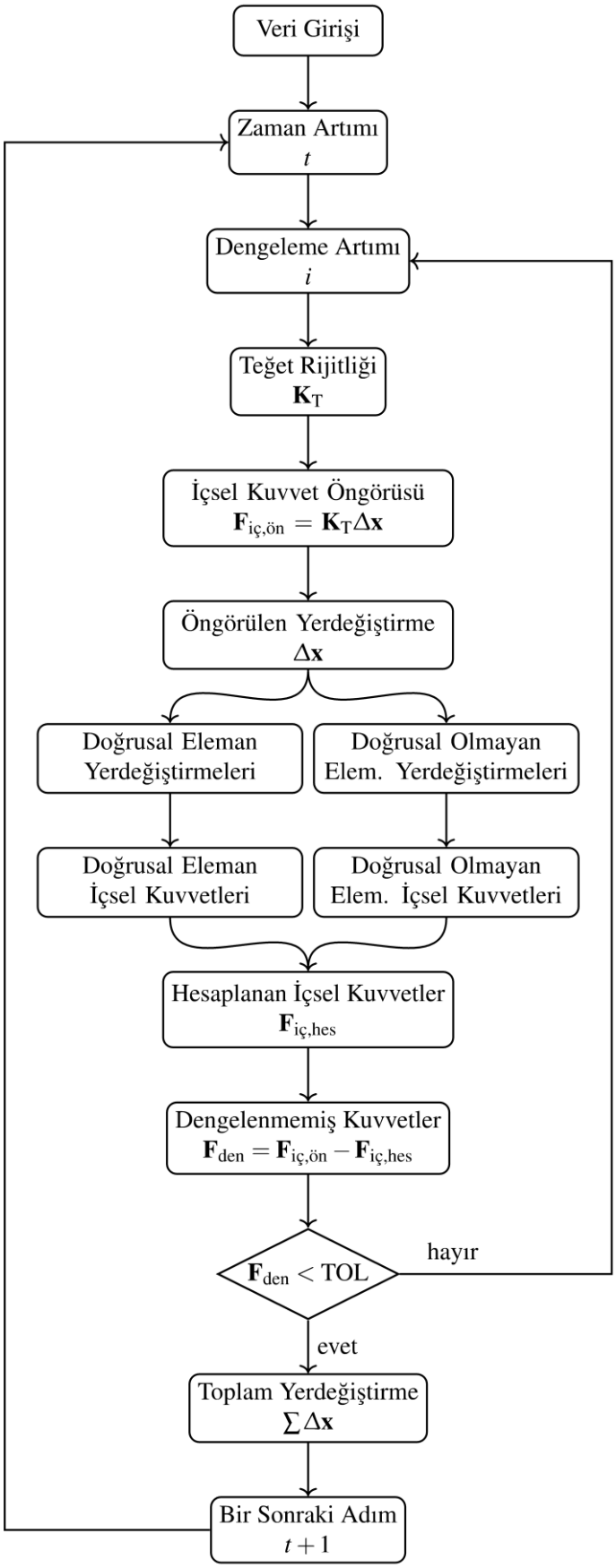
çalışmasına engel olmayacak kadar küçük zaman adım boyutuna kadar küçültülerek çalıştırılmalıdır.

Doğrusal olmayan analizlerde rijitlik, sönümleme ve yükleme yerdeğiştirme, hız ve zamana bağlı olabilir. Bu da çözümün yinelemeli olarak yapılmasını gerektirebilir. Doğrusal olmayan denklemler her adımda yinelemeli bir şekilde çözülürler. Denklemlerin bir tarafındaki ifadenin zamana bağlı doğrusal arttığı ve herbir yinelemede diğer tarafın yakınsama koşulları kontrol edilerek çözüme devam ettiği kabul edilir. Çözüm yakınsayana kadar yineleme devam eder.

Kütle, rijitlik ve sönüm matrisleri yapısal sistemin geometrisine, elastisite modülüne, ataletine, kesit özelliklerine göre elde edilir. Bu genel matrisler elde edildikten sonra dinamik diferansiyel denklemin adım adım çözümü yapıya ait yerdeğiştirme, hız ve ivme değerlerini zamana bağlı şekilde verir.

Doğrusal olmayan davranışı temsil eden dönme yayları kolon ve kiriş yüzeylerine yerleştirilebilir. Dönme yaylarının davranışı, kolon ve kiriş kesitlerinin moment-eğrilik ilişkilerinin çift doğrusal hale getirilmesi ile elde edilebilir.

Doğrusal olmayan deprem analizi ile ilgili işlem sıralaması Şekil 3.19’da verilmiştir.



Şekil 3.19 : Doğrusal olmayan dinamik analizin akış şeması.

3.4. Programlama Esasları

Programlama, bilgisayarın donanıma nasıl davranacağını anlatan, bilgisayara yön veren komutlar, kelimeler ve aritmetik işlemlerdir. Günümüzde bilgisayar programları tarım, sağlık, savunma, eğlence, iletişim, haberleşme, inşaat gibi birçok alanda kullanılmaktadır.

Bilgisayar programlarını anlayabilmek için bir programlama diliyle yazılan komutlar belirlenen bir görev için çalıştırılabilir. Matlab (2017) dilinde bir değişkene yazı atamak örnek olarak verilebilir (Şekil 3.20).

```
>> yazi= ' Merhaba '

yazi =

 Merhaba
```

Şekil 3.20 : Bilgisayar ekranına yazma islemi.

Bilgisayar programı, bilgisayar yazılımı olarakta isimlendirilebilir. Aynı zamanda bilgisayar programlama, bilgisayar kodlama olarakta isimlendirilebilir.

İnsanlar kendi aralarında iletişime geçebilmek için birçok dil geliştirdikleri gibi bilgisayar mühendisleri de bilgisayar donanımı ile iletişime geçebilecek birçok programlama dili geliştirmişlerdir. Bu programlama dillerine Java, C, C++, Php, Matlab, Python örnek gösterilebilir.

Bilgisayar programlama dilleri bazı temel parçalardan oluşmaktadır. Bunlar:

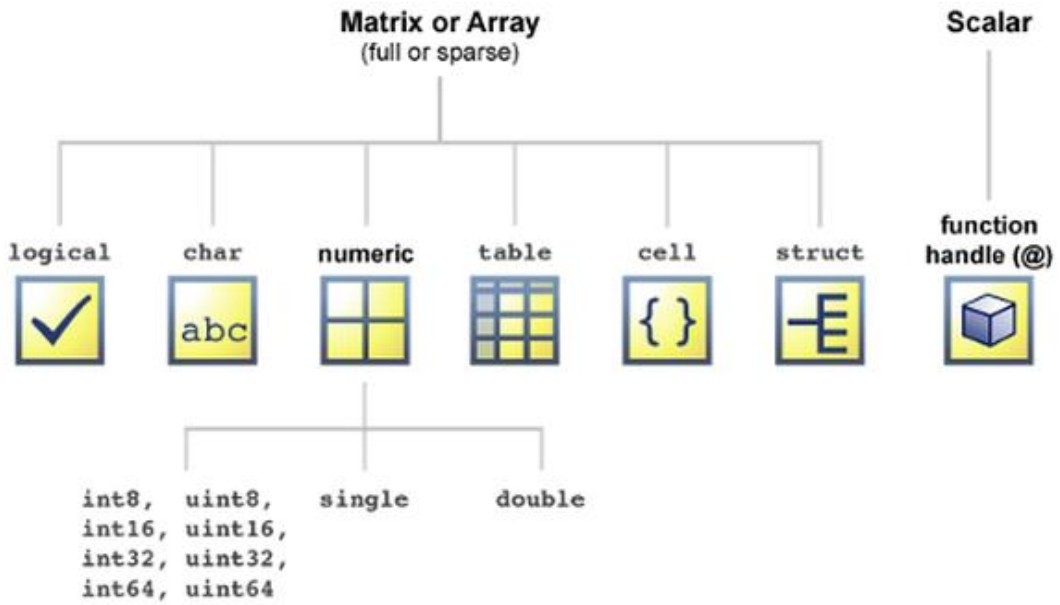
- Programlama ortamı
- Temel sözdizimi
- Veri çeşitleri
- Değişkenler
- Anahtar kelimeler
- Temel operasyonlar (toplama, çarpma gibi)
- Kontrol mekanizması
- Döngüler
- Numaralar

- Karakterler
- Vectörler ve matrisler
- Fonksiyonlar
- Girdi ve çıktı dosyaları

Bu temel parçaların bazıları,Matlab programındaki uygulamarıyla bu başlık altında irdelenecektir.

Matlab programında çalışırken kullanlabilecek çok farklı veri çeşiti ve sınıfı vardır. Tam sayı ve ondalık sayı verileri ile oluşturulmuş vektör ve matrisler, metin ve karakter, mantıksal doğru ve yanlış ifadeleri Matlab tarafından oluşturulabilir. Tablolar, yapılar, hücre dizileri benzer olmayan yapıdaki verilerin bilgisayar hafızasında depolanmasını sağlar.

Matlab programında 16 farklı temel sınıf vardır (bknz Şekil 3.21). Bu sınıfların herbiri matris ve vektör formatındadır. Bu verileri düzenleme ve aralarında geçişler yapabilmek için birçok fonksiyon vardır.



Şekil 3.21 : Temel Matlab sınıfları ve veri çeşitleri.

İşlem operasyonları tarafından programa girilen veriler işlenerek farklı sonuçlar çıkarılabilir. Matlab programında farklı işlem operasyonları vardır. Matris operasyonları için kullanılan işlem operatörleri şöyledir:

- \+ Toplama

- - Çıkarma
- * Çarpma
- / Bölme
- \ Soldan bölme
- ^ Kuvvet
- ‘ Transpose
- ( ) İşlem önceliğini belirlemek için parantez

Vektör operasyonları toplama ve çıkarmada matris elemanlarına benzese bile çarpma, bölme, kuvvet alma ve transpozesini alma gibi işlemlerde farklılık gösterir. Aynı satırda bulunan elemanları işleme tabi tutabilmek için genellikle matris operatörlerinin önüne ‘.’ işareti eklenir. Vektör işlemleri için kullanılan işlem operatörleri şöyledir:

- + Toplama
- - Çıkarma
- * Eleman elemana çarpma
- ./ Eleman elemana bölme
- .\ Eleman elemana soldan bölme
- .^ Eleman elemana kuvvet alma
- .’ Eleman elemana tersini alma

Çarpma operasyonuna örnek verilmiştir (bknz Şekil 3.22).

```
>> a=(0:9)';
>> carpma=[a a.^2 2.^a]

carpma =

     0     0     1
     1     1     2
     2     4     4
     3     9     8
     4    16    16
     5    25    32
     6    36    64
     7    49   128
     8    64   256
     9    81   512
```

Şekil 3.22 : Matlab programında eleman elemana çarpma operasyonu.

Verilerin birbiri ile karşılaştırılmasında ilişki operatörleri kullanılır. İlişki operatörleri şu şekildedir:

- < Küçüktür
- > Büyüktür
- <= Küçük eşittir
- >= Büyük eşittir
- = = Eşittir
- ~= Eşit değildir

Kontrol mekanizması, bir çok programlama dilinde 'if' 'elseif' 'else' ifadeleri kullanılarak programlanmaktadır. Kontrol ifadeleri, farklı durumlar karşısında bilgisayarın nasıl davranacağını tarif eder. Bilgisayar kontrol ifadeleri sayesinde farklı komutları farklı durumlar için kullanacaktır. Bilgisayar, bir programda ne yapacağına dair kararı verirken farklı durumlarda hangi komutları çalıştıracağını anlatan kontrol ifadeleriyle donatılmış olmalıdır. Matlab programında kontrol ifadeleri şartı söyledikten sonra istenen işlem ifadesinin yazılmasıyla kullanılabilir (bknz Şekil 3.23).

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

Şekil 3.23 : Matlab programında kontrol ifadelerinin genel kullanımı.

Programlama esnasında bir işlemi tekrar tekrar yapmak için döngüler kullanılır. Aynı işlemi farklı veriler içinde döngüler sayesinde çalıştırabiliriz. Sıralı bir şekilde tanımlanan aralıkta işlem yapılır ve sonuçlar yine sıralı bir şekilde çıkarılır. Matlab döngüleri 'for' ifadesiyle yapmaktadır.

```
for index = values
   statements
end
```

Şekil 3.24 : Matlab programında 'for' döngüsünün kullanımı.

Bu başlık altında buraya kadar anlatılan kısım seri hesaplama mantığı ile gözönünde bulundurulmuştur. Seri programlama, ard arda ve sırayla işlemlerin gerçekleştirilmesini ifade eder. Seri hesaplamada olayların kronolojik sıralaması doğrultusunda oluşturulan problemler modellenir. Seri işlemleri zihinde canlandırmak için bir yemeğin yapılma süreci düşünülebilir. Öncelikle malzemeler elde edilir, sonrasında işlenir ve sonuç olarak bir yemek ortaya çıkar. Burada yemek tarifine göre malzemelerin işlenmesi düşünüldüğünde yemek tarifi bir program komutu gibi olmaktadır. Bütün bu işlemleri yapmak için mutfakta ki araçların kullanıldığı gibi bilgisayarda da donanım kullanılır. Başka bir örnek verilmek istenirse, yemek yapmak, yemeği yemek ve bulaşıkları yıkamak bir sıradır. Önce yemekleri yemek sonra bulaşıkları yıkamak daha sonra ise yemek yapmak çok daha az akla yatan bir sıradır. Bu iki örnekle seri proglamanın felsefesi hakkında bilgi edinmek mümkündür.

Seri programlama da girdiler sırayla alınır ve program bunları işledikten sonra çıktıyı sırayla verir.

3.4.1. Paralel programlama

Bilgisayar mimarisinin programlama esasları üzerinde bazı değişikliklere neden olduğu bilinmektedir. Yıllar öncesinde tek işlemcili olarak üretilen bilgisayarlar son yıllarda çok işlemcili olarak üretilip paralel programlamaya uygun hale getirilmişlerdir. Birden fazla işlemci sayesinde defalarca tekrarlanması gereken döngüler farklı çekirdeklere gönderilerek aynı zamanda işlenebilir.

Paralel programlama eşzamanlı hesaplama veya aynı anda işleme anlamlarına gelebilir. Seri programlamanın aksine, işler eşzamanlı yapılıyorken paralel programlamada alt işlemler arasında bir iletişim ve sinyal gönderme gerçekleşebilir. Paralel programlama, merkezi işlem birimi (MİB) kullanımı hakkında düşünmeyi zorunlu kılan bir ortamdır.

3.4.2. Matlab paralel programlama araçları

Matlab yazılım dili bilgisayarda bulunan işlemcileri aktif bir şekilde kullanmak ve paralel programlama için uygun ortamı sağlamak için parallel hesaplama araç kutusunu geliştirmiştir. Paralel hesaplama araç kutusu, hesapsal ve veri bakımından yoğun problemleri çok çekirdekli işlemciler, grafik kartları ve bilgisayar kümelerini kullanarak çözüme ulaştırma imkanı sağlamaktadır.

Paralel for döngüleri, özel vektör çeşitleri ve paralelleştirilmiş sayısal algoritmalar gibi yüksek seviye programala ifadeleri Matlab uygulamalarını CUDA ve MPI programlama kavramlarını kullanmadan paralelleştirmeyi sağlar.

Bu araç kutusu, uygulamaları yerel olarak çalışan işçiler (Matlab hesaplama motoru) ile işleyerek çok işlemcili bilgisayarların tüm işleme gücünü kullanmayı sağlar. Kodlamada herhangibir değişiklik yapmadan, aynı uygulama bilgisayar kümelerinde veya hesaplama sunucularında çalıştırılabilir.

Paralel programlama birçok hesaplamayı eşzamanlı olarak yapmayı sağlar. Büyük problemleri küçük parçalara ayırarak aynı anda çözebilir. Paralel programlamanın gözönünde bulundurulmasının asıl sebebleri şunlardır:

- Görevler ayrılarak ve eşzamanlı işlenerek zaman kazanımı sağlaması
- Büyük verili problemleri veri ayrıştırmasına giderek çözmesi
- Masaüstü bilgisayarların kaynakları kullanmak, bilgisayar kümeleri oluşturmak ve buluttan hesap yapabilmek.

Bunların yanı sıra Matlab paralel hesaplama araçkutusunun bazı temel özellikleri vardır.

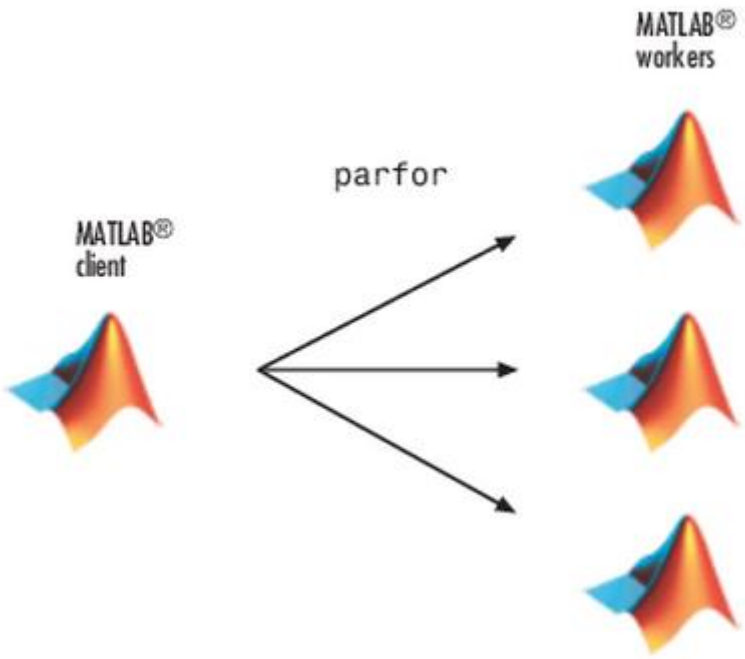
- CUDA özelliği olan grafik işlem birimleri destekleyerek paralel hesaplama yapar ve merkezi işlem birimine göre oldukça hızlıdır.
- Paralel uygulamalar iletişim halinde veya yığın olarak hesaplanabilir. Çekirdekler iletişim halinde olduğunda değişkenlere ulaşma imkanı varken yığın olarak hesaplanan uygulamalarda işlemin bitmesi beklenmelidir.
- Tek program çoklu veri özelliği ile aynı programın farklı veriler ile çalışmasını ve her veriyi ayrı çekirdeklerin işlemesini sağlar.

Matlab paralel hesaplama araç kutusunda ‘parfor’ ve ‘spmd’ olmak üzere paralel programlama için iki temel fonksiyon vardır.

3.4.2.1. Parfor ve örnekleri

Matlab programındaki ‘parfor’ döngüsünün temel konsepti standar ‘for’ döngüsü ile aynıdır. Matlab oturumu paralel havuzun içerisindeki Matlab işçilerini koordine etmektedir. Bu yüzden döngü içerisindeki yinelemeler paralel havuzda

gerçekleşmektedir. Parfor operasyonu için gerekli olan veri oturumdan işçilere gönderilir (Şekil 3.25). İşlemler olduktan sonra sonuclar oturuma tekrar gönderilir.



Şekil 3.25 : Matlab oturumu ve işçileri.

Matlab işçileri aynı döngü üzerinde eşzamanlı olarak hesaplama yapabildikleri için 'parfor' döngüsü normal 'for' döngüsüne göre önemli bir şekilde daha iyi performans gösterir. 'Parfor' döngüsü, Monte Carlo simülasyonu gibi basit bir hesabın çok fazla iterasyonuna ihtiyaç olduğu durumlarda faydalıdır.

Parfor kullanımının kolaylığını gösteren bir örnek verilebilir (Şekil 3.26).

```
for i = 1:1024
  A(i) = sin(i*2*pi/1024);
end
```

```
parfor i = 1:1024
  A(i) = sin(i*2*pi/1024);
end
```

Şekil 3.26 : Parfor kullanımı.

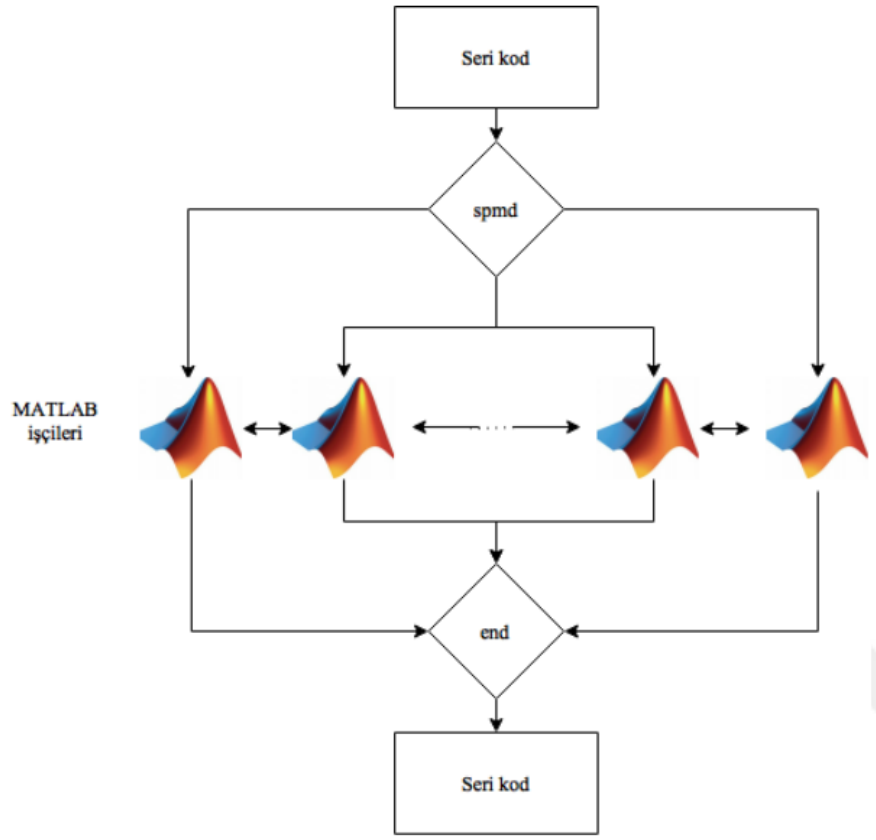
Parfor kullanılırken dikkat edilmesi gereken en önemli nokta döngü esnasında işlenen verilerin birbirine bağlı olmamasıdır. Döngünün bir adımındaki veri sonraki veri için gerekli ise parfor kullanılamaz. Fibonacci serisinin hesaplanmasında bir önceki işlem sonucu bir sonraki işlemde kullanıldığı için parfor ile paralelleştirmeye uygun değildir (Şekil 3.27).

```
f = zeros(1,8)
f(1) = 1;
f(2) = 2;
parfor n =3:8
      f(n) = f(n-1) + f(n-2);
end
```

Şekil 3.27 : Parfor ile paralleştirmeye uygun olmayan durum.

3.4.2.2. Spmd ve örnekleri

Spmd dil yapısı, seri ve paralel programlamanın kusursuz şekilde birbirini takip etmesini sağlar. Spmd ifadesi bir kod bloğunu birçok işçi ile aynı anda çalıştırılmasını sağlar. Seri ve paralel hesaplama arasında geçiş yapmak fazlasıyla kolaydır. Spmd bloğu içindeki kodlar paralel çalışırken dışındaki kodlar seri çalışır (bknz Şekil 3.28).



Şekil 3.28 : Spmd çalışma prensibi.

Spmd (single program multiple data) tek program çoklu veri anlamına gelmektedir. Buradaki tek program ile kastedilen aynı kodların birçok işçide çalışmasıdır. Çoklu veri ile kastedilen ise bütün işçilerde aynı kod çalışsa bile herbir işçi farklı ve özel bir veriyi çalıştırmasıdır.

Spmd ifadesi içerisine yazılan kodlar büyük veri setleri üzerinde çalışıyorsa verileri işlemcilere paylaştırarak çalıştırır. Spmd ifadesinin yanına parantez içerisinde yazılan sayı, paralel havuzda açılacak işçi sayısını belirlemektedir (Şekil 3.29).

```
spmd (m,n)
    <statements>
end
```

Şekil 3.29 : Spmd için işçi sayısının belirlenmesi.

Parfor'dan farklı olarak spmd ifadesi çekirdeklere ulaşım imkanı da sağlamaktadır. Herbir çekirdek bir lab olarak düşünülmekte ve her lab için bir kimlik numarası verilmiştir. Bilgisayarda bulunan çekirdek sayısı kadar lab numarası otomatik olarak oluşturulur. İstenilen bölüme tanımlanan kod çalıştırılabilir.

Bu bölmelere 'labindex' ifadesiyle ulaşılabilir. Aşağıdaki örnekte ilk işlemcide 9x9 luk bir rastgele matris oluşturulurken diğer işlemcilerde 4x4 lük bir rastgele matris oluşturulur (bknz Şekil 3.30).

```
spmd (3)
    if labindex==1
        R = rand(9,9);
      else
        R = rand(4,4);
    end
end
```

Şekil 3.30 : Bilgisayarda bulunan işlemcilere ulaşım.

Herbir çekirdekteki veriler değişkene depolanarak erişim sağlanabilir. Matlab oturum bölümündeki kompozit nesneler işçilerdeki verilere direk ulaşım sağlar. Bu kompozit nesnelere en yaygın olarak spmd ifadelerinde rastlanır.

Matlab programında iki farklı şekilde kompozit nesne tanımlanabilir. İlk olarak oturumda 'composite' fonksiyonu kullanılarak oluşturulur. İkinci olarak spmd içinde tanımlanan değişkenler kompozit olarak bilgisayar hafızasında yerini alır. Spmd bloğu içerisinde normal değişken olarak bulunur fakat oturuma geçildiğinde kompozit nesne olarak erişilebilir.

Matlab oturumunda herbir işçi için bir tane kompozite nesne tanımlanır. Şekil 3.31' de 'a' matrisi spmd bloğu içerisinde labindex e göre tanımlanmıştır. Bunun için b ilk

çıktısında 2x2 lik bir matris verirken ikinci çıktısında 4x4 lük bir matris verir ve her ikiside kompozit nesne olarak hafızaya kaydedilmiştir.

```
>> spmd
a=magic(labindex+1);
end
>> b=a{1}

b =

     1     3
     4     2

>> b=a{3}

b =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

Şekil 3.31 : Composite ‘b’ elemanları.

4. OLUŞTURULAN PROGRAMLAR VE ÖRNEK YAPILAR

4.1. Doğrusal Statik Analiz Programı

Bu tez kapsamında yazılan ilk program çerçeve sistemlerin doğrusal analizini otomatik yapan bir Matlab programıdır (bknz Ek A). Programda denklemlerin düzenlenmesi ve çözülmesi için matris yöntemlerden faydalanılmıştır.

Analiz için gerekli olan verileri programa tanıtmak için izlenen yöntem iki şekildedir. Birincisi programa otomatik olarak girdileri hesaplatmaktır. Bu otomatize işleminde kullanılan değişkenlerin sonuna '_automatic' kelimesi eklenmiştir. İkincisi ise programa verilerin elle girilmesidir. Bunun için kullanılan değişkenlerin sonuna da '_manual' kelimesi eklenmiştir. Sonra bu ikisi arasında bir tercih yapmak gerekmektedir. Bu işlem sırası bazı parametreler için programda bolca kullanılmıştır. Örnek verecek olursak;

`coordinates_of_nodes_automatic` → kat adedi, kat yüksekliği, açıklık adedi, açıklık aralığı ve ilk düğüm noktasının koordinatları tanıtıldıktan sonra düğüm noktalarının koordinatlarını kendisi otomatik olarak hesaplar.

`coordinates_of_nodes_manual` → `[0 0;15 0]` her satırın ilk kolunu x eksenini ikinci kolunu ise y eksenini verecek şekilde düğüm noktalarının koordinatlarını elle girilmesi gerekmektedir.

Düğüm noktaları için bu iki işlemi yaptıktan sonra hangisini kullanılacağını programa söylemek için `determinate_coordinates` değişkenine '0' veya '1' değerleri girilmelidir. Eğer '0' değeri girilirse manual olarak oluşturulan düğüm noktası koordinatlarını alacak '1' değeri girilse otomatik oluşan düğüm noktası koordinatlarını alacak ve işleme devam edecektir.

Programa girdileri verdikten sonra işlemleri adıma adım gerçekleştirmektedir. Burada girdilerin nasıl verildiği ve nasıl işlendiği anlatılacaktır.

- Düğüm noktalarının koordinatlarını belirleme

Düğüm noktaları elle veya otomatik olarak belirlendikten sonra bu koordinatlar sıralanmak ve numaralandırılmak için SortingAndNumbering_function fonksiyonuna

gönderilir. Fonksiyonun çalışma sistemi şu şekildedir. Öncelikle en alt yükseklikteki düğüm noktalarını soldan sağa doğru sıralar. O yüksekliği bitirdikten sonra bir üst yükseklik içinde aynı işlemi tekrarlar. Sıralama işlemini tamamlayınca düğüm noktalarına numara verir. Fonksiyon çıktı olarakta sıralanmış ve numaralandırılmış düğüm noktalarının koordinatlarını verir.

coordinates_sorted_and_numerated=SortingAndNumbering_function(coordinates_of_nodes);

- Düğüm noktalarını eşleştirme

Düğüm noktalarının hangi düğüm noktaları ile eşleşeceğini programa tanıttığımız bölümdür.

`connectivity_manual=[1 3;2 4];` elle yandaki gibi girdiğimizde program şöyle anlamaktadır. '1' numaralı düğüm noktasını '2' numaralı düğüm noktası ile eşleştir ve sonra da '2' numaralı düğüm noktasını '4' numaralı düğüm noktası ile eşleştir.

`connectivity_automatic` bu değişkende düğüm noktalarını birleştirmektedir. Her bir kat için ilk olarak soldan sağa doğru kolonları oluşturacak düğüm noktaları birleştirilir daha sonra yine soldan sağa doğru kirişleri oluşturacak düğüm noktalarını birleştirilir. Bu işlem tüm katlar için tekrarlanır.

- Düğüm noktalarında tutulu olan serbestlik derecelerinin belirlenmesi

Düğüm noktalarında hangi serbestlik derecesini tutacağımızı programa buradaki değişken ile söylemekteyiz.

`restrained_DoF_manual={[1 1 2];[5 3 1]};` bu şekilde `manuel olarak` verileri girebiliriz. Buradan programa şu bilgiler gider. Cell değişkeninin herbir satırı bir düğüm noktasını temsil eder. Her satırdaki vektörün ilk kolununda bulunan eleman düğüm noktasının numarasını programa söyler. Daha sonra o düğüm noktası için sıra takip etmeksizin hangi serbestlik derecesini tutmak isteniyorsa girilir. Bu örnekte program şöyle yapmaktadır. '1' numaralı düğüm noktasının 1 ve 2 yönündeki serbestlik derecelerini daha sonra '5' numaralı düğüm noktasının 3 ve 1 yönündeki serbestlik derecelerini tutmaktadır. Burada 1, 2, 3 diye bahsedilen serbestlik derecelerinin yönleri sırasıyla normal, kesme ve moment için belirlenen pozitif yönlerdir.

`restrained_DoF_automatic` bu değişkende ise çerçeve sistemin en alt seviyesindeki bütün düğüm noktalarının 1, 2, 3 yönündeki serbestlik dereceleri tutulmaktadır. Yani bu düğüm noktaları ankastre olarak teşkil edilmektedir.

- Düğüm noktalarına gelen yüklerin belirlenmesi

İstenilen herhangibir düğüm noktasında yük tanımlamak için kullanılır. Burada otomatik bir parametre tanımlanmamıştır. Değerler elle aşağıdaki değişken yardımı ile girilmektedir. Aynı zamanda grafikte yüklemeleri daha rahat görebilmek için font boyutlarını tanımlama imkanı da vardır.

`nodes_load={[4 10 -20 45];[2 -125 5 -15]}` bu cell değişkeni şu şekilde çalışmaktadır. Cellin her bir satırı bir düğüm noktasını temsil etmektedir. Satırda bulunan vektörün ilk kolonu düğüm noktasını, ikinci kolonu o düğüm noktasında global yönde X eksenindeki yükü, üçüncü kolonu Y eksenindeki yükü, son olarak dördüncü kolonu da moment miktarını gösterir. Yukarıdaki değerlerden program şöyle anlamaktadır. '4'numaralı düğüm noktası için pozitif X yönünde 10 kN, negatif Y yönünde 20 kN, pozitif moment yönünde 45 kN*m sonrada'2' numaralı düğüm noktası için negatif X yönünda 125 kN, pozitif Y yönünde 5kN, negatif moment yönünde 15 kN*m yükleme yapılacaktır.

`nodes_load_fontsize=20;` bu değişken yüklerin fontunu ayarlamak içindir.

- Tutulu düğüm noktalarında mesnet oturmalarının belirlenmesi

İstenilen herhangibir düğüm noktasında mesnet oturması tanımlamak için kullanılır. Değerler elle aşağıdaki değişken yardımı ile girilmektedir.

`support_settlement={[1 0.01 0 0];[2 0 0.05 0.0001]}` bu cell değişkeni şu şekilde çalışmaktadır. Cellin her bir satırı bir düğüm noktasını temsil etmektedir. Satırda bulunan vektörün ilk kolonu düğüm noktasını, ikinci kolonu o düğüm noktasında global yönde X eksenindeki mesnet oturmasını, üçüncü kolonu Y eksenindeki mesnet oturmasını, son olarak dördüncü kolonu da dönme miktarını gösterir.

Birimi X ve Y yönleri için metre dönme için de radyandır. Çalışma prensibi düğüm noktaların gelen yüklerin belirlenmesi bölümündeki gibidir.

- Elemanlar üzerinde noktasal ve yayılı yük tanımlanması

Elemanlar üzerine yük tanımlamak için aşağıdaki parametreler kullanılır. Bu değişkenlar tanımlanırsa bütün çubuk elemanlarda bu yüklerin etkisi görülecektir.

`point_load=10` → noktasal yükün miktarı buradan tanımlanır.

`ratio=0.5` → yükün çubuk elemanın hangi bölgesinde yer alacağı burada belirlenir.

`point_load_fontsize=12` → noktasal yükün grafiklerdeki boyutu buradan ayarlanabilmektedir.

`uniform_load=5` → yayılı yükün miktarı buradan tanımlanır.

uniform_load_fontsize=14

- Elemanın kesit özelliklerinin belirlenmesi

Çerçeve sistemde bulunan bütün çubuk elemanların kesit özellikleri aynı olarak kabul edilmiştir.

`depth=1;` → kesitin derinliği burdan tanımlanır.

`width=0.5;` → kesitin genişliği burdan tanımlanır.

`modulus_of_elasticity=35000000;` → malzemenin elastisite modülü burdan tanımlanır.

- Grafikle alakalı bazı düzenlemelerin belirlenmesi

`determinate_number_nodes=1` → düğüm noktalarının numalarını grafikte görmek için '1' görmemek için '0' yazılmalıdır.

`determinate_dir_element=1` → çubuk elemanların yönünü grafikte görmek için '1' görmemek için '0' yazılmalıdır.

`determinate_name_element=1` → çubuk elemanların isimlerini grafikte görmek için '1' görmemek için '0' yazılmalıdır.

Aşağıdaki fonksiyon sayesinde herbir çubuk elemana soldan sağa doğru öncelikle kolonlara sonrada kirişlere isim verilir. Bu işlem bütün katlar için yapılmaktadır.

name=Naming(size(connectivity,1));

- M, N, T grafiklerinde değerlerin büyüklüğünü çizerken ölçeklendirme

Analiz sonucu ortaya çıkan moment, normal kuvvet ve kesme kuvveti değerleri bazen çok büyük çıkabilmekte bu durum sonucunda ise grafik çok karışık hale

gelmektedir. Bunu önlemek için m, n, t grafikleri için ölçeklendirme parametreleri oluşturulmuştur.

fontsize_mnt=12;

scale_normal=2;

scale_shear=2;

scale_moment=10;

- M, N, T grafiklerini çizdirmek istediğimiz çubuk elemanların belirlenmesi

Program otomatik olarak sistemin m, n, t grafiklerini çıkarmaktadır. Fakat özellikle bir çubuk elemanın m, n, t grafikleri elde edilmek istenirse programa girmek gereklidir. Bunun için oluşturulan değişken aşağıdadır.

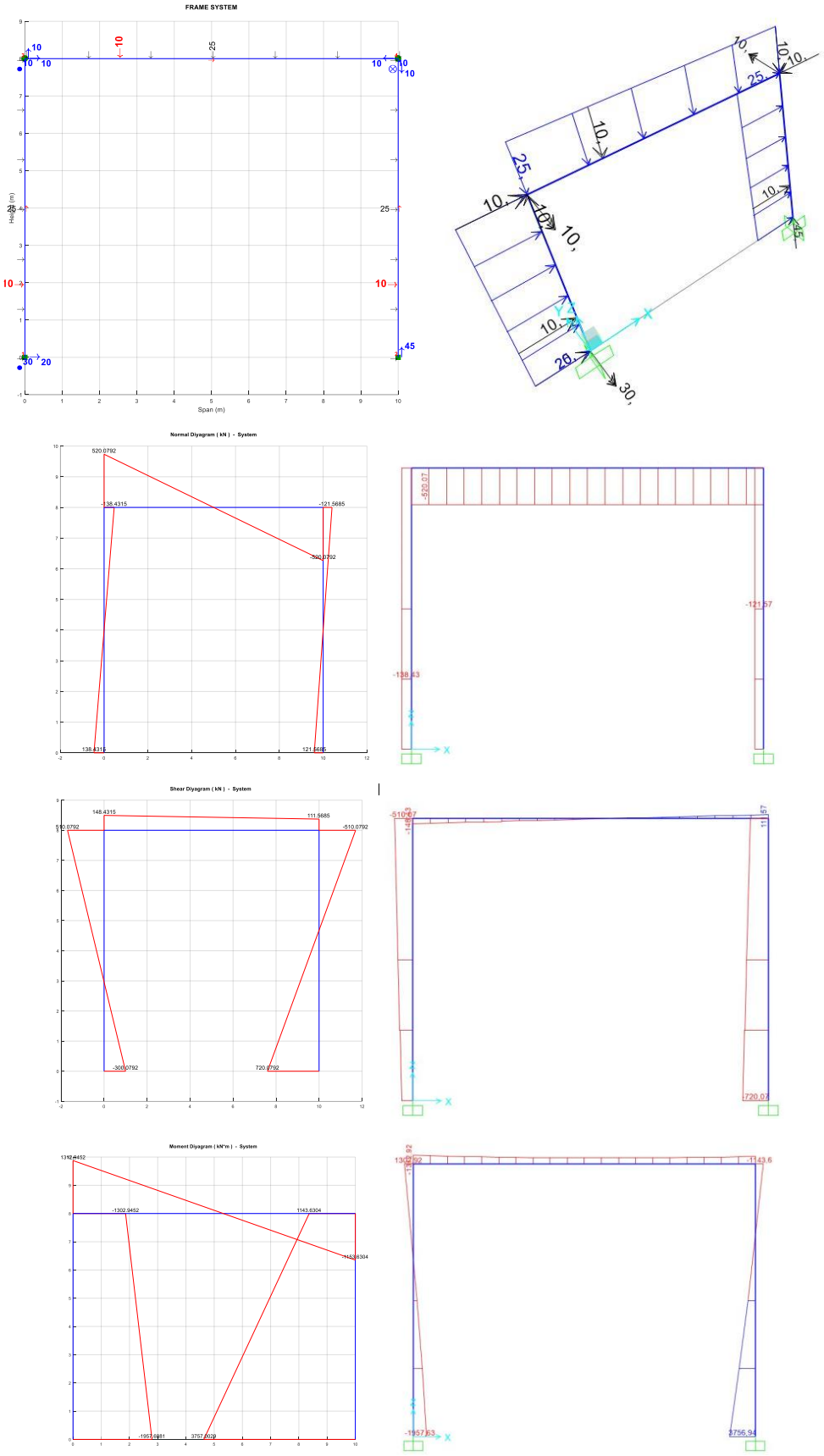
`mnt_of_element=[2 5 30];` bu vektöre tanımlanan herbir çubuk eleman için diyagramları oluşturacaktır.

4.1.1. Matlab programı ile Sap2000 sonuçlarının karşılaştırılması

Matlab programı ile çözülecek bazı çerçeve sistemler sap2000 paket programı ile de çözülecek ve sonuçlar kıyaslanacaktır. Fakat dikkat edilmesi gereken bir konu vardır. Sap2000 programının yön kabulleri ile matlab programının yön kabulleri aynı değildir. Bunun için hesaplanan değerler aynı olsada işaretlerinde farklılıklar olabilmektedir. Dikkat edilmesi gereken diğer bir konu ise matlab programı ile elde ettiğimiz M, N, T değerleri çubuk elemanların ucunda oluşan değerlerdir. Çubuk elemanın orta bölgelerindeki değerleri hesaplanmamıştır. Kıyaslama yapılırken sap2000 grafiklerinde gösterilen M, N, T değerlerinin sadece uç bölgelerine bakılacak buralar dikkate alınacaktır. Yukarıda da bahsettiğimiz gibi işaretleri farklı olabilir.

Şekil 4.1'de hem düğüm noktalarına hemde çubuk elemanlara yükleme yapılan bir örnek verilmiştir. Çubuk elemanlara yapılan yüklemeler noktasal olarak 10 kN ve yayılı olarak 25 kN olarak belirlenmiştir. Bunun yanı sıra 1. Düğüm noktasına pozitif Y yönünde 0.01 m çökme verilmiştir. 2. Düğüm noktasına ise negatif X yönünde 0.05 m ve pozitif dönme yönünde 0.002 radyanlık dönme verilmiştir Aynı işlemler Sap2000 paket programında da yapılıp her iki programın sonuçları kıyaslanmıştır.

Matlab Sonuçları Sap2000 Sonuçları



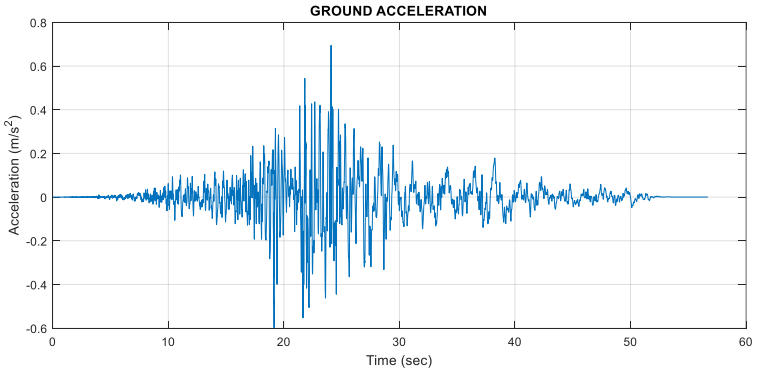
Şekil 4.1 : Doğrusal statik analizi için Matlab ve Sap2000 kıyaslaması.

4.2. Doğrusal Deprem Analiz Programı

Çok serbestlik dereceli sistemlerin doğrusal deprem analiz programı (bknz Ek B), doğrusal statik analizden farklı olarak sönüm ve kütle matrisini içermektedir. Doğrusal statik analizdeki gibi rijitlik matrisi hesaplanır. Kütle matrisi ve rijitlik matrisinin orantı katsayısı yapıya verilen %5 sönüm oranı için elde edilir. Rayleigh orantı katsayıları kütle ve rijitlik ile çarpılarak sönümleme matrisi oluşturulur. Dinamik denge denklemi Newmark-beta yöntemi ile lineer forma getirilir ve çözüme ulaşılır. Çözüm sonucu yapının davranışı ortaya çıkar. İstenen düğüm noktasındaki istenen serbestlik derecesinin yerdeğiştirme, hız ve yapısal ivme değerleri zamana bağlı olarak elde edilip figür olarak çizdirilebilir. Bununla beraber yapının frekans ve periyot değerleri eigen değeri çözümü yapılarak elde edilir. Matlab programı bütün bu işlemleri yaparken oldukça fazla matris operasyonu gerçekleştirmektedir.

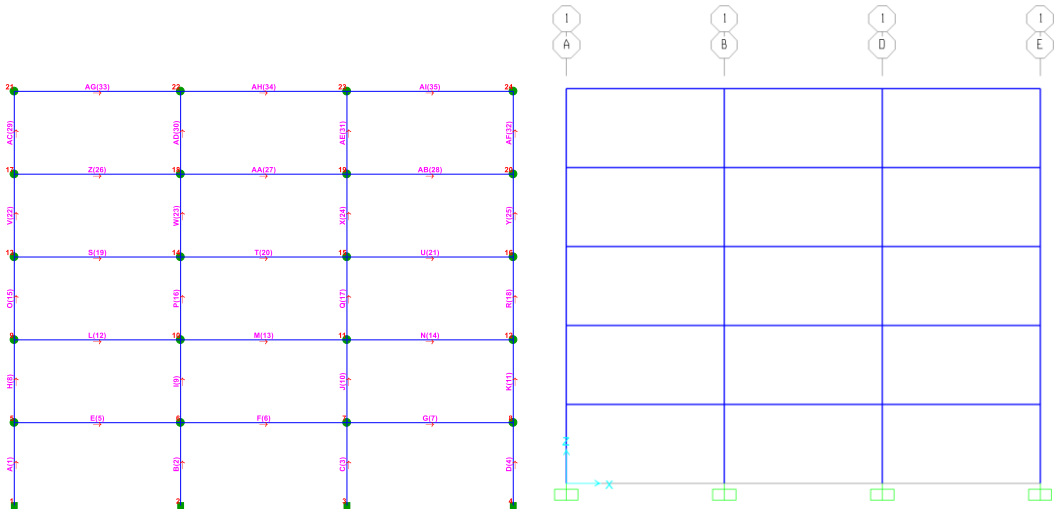
Matlab programının avantajların bir tanesi görsel araçları kullanma imkanının fazla olmasıdır. Bu tez kapsamında yazılan programlarda görsel araçlardan fazlasıyla faydalanılmıştır. Çerçeve sistemleri oluşturan çubuk elemanlar ve düğüm noktaları isimlendirilerek figürlerde gösterilmiş ve olası hataları görerek farketme ihtimali artırılmıştır.

Matlab programında hazırlanan doğrusal deprem analiz programı için hazırlanan örnekte 6 metre uzunluğunda 3 açıklıklı 3 metre yüksekliğinde 5 katlı basit bir yapı tercih edilmiştir. Elastisite modülü olarak 3e7 kN/m^2, kolon 60 cm eninde ve derinliğinde, kiriş ise 60 cm yüksekliğinde 40 cm derinliğinde belirlenmiştir. Rayleigh katsayılarını hesaplamak için girilen sönüm oranları ise %5 olarak tanıtılmıştır. Deprem kaydı olarak kuvvetli yer hareketi olan ‘darfield’ kaydı kullanılmıştır (bknz Şekil 4.2). Bütün bu girdilerden sonra program çalıştırılmış ve sonuçlar alınmıştır.

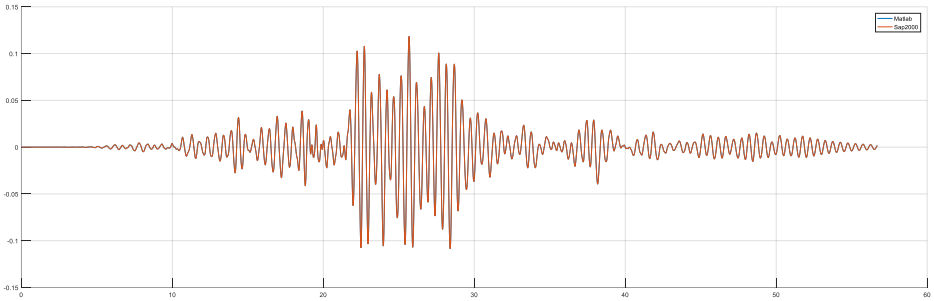


Şekil 4.2 : Darfield deprem kaydı ivme değerleri.

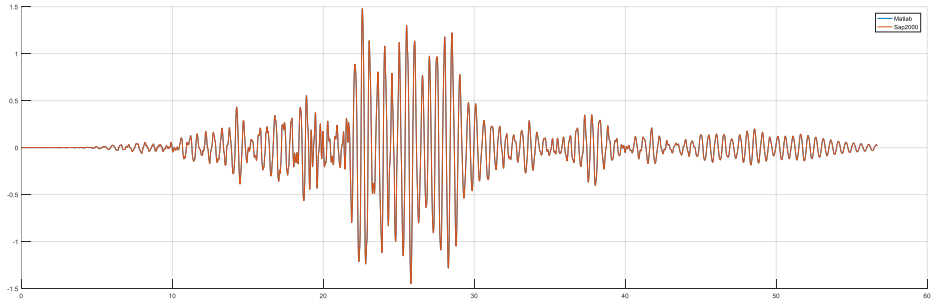
Yukarıda bahsedilen parametreler aynı değerleri ile Sap2000 paket programına da girilerek çözdürülmüştür. Her iki program sonuçları kıyaslanmıştır.



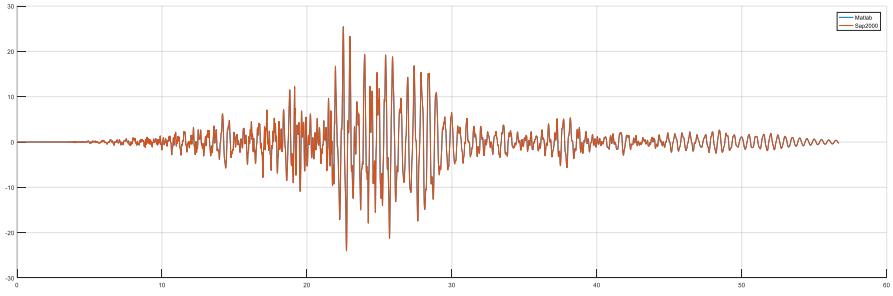
Şekil 4.3 : Matlab programı ve Sap2000'de 3 açıklık 5 katlı çerçeve sistemi.



Şekil 4.4 : Sistemin yerdeğiştirme karşılaştırması.



Şekil 4.5 : Sistemin hız karşılaştırması.



Şekil 4.6 : Sistemin ivme karşılaştırması.

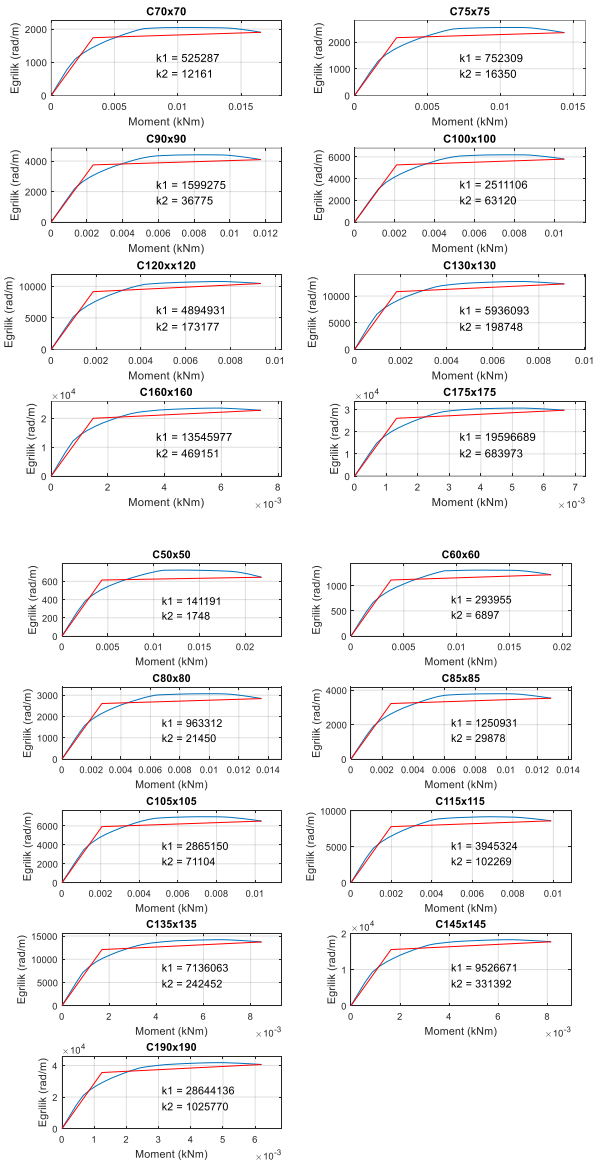
4.3. Doğrusal Olmayan Deprem Analiz Programı

Doğrusal olmayan deprem analiz programının, doğrusal deprem analizinden farklı olarak doğrusalsızlığı ifade eden elemanlar içerir. Deprem analizleri, kat sayıları değişen ve çubuk eleman kesitleri farklılaşan numune binalarda yapılmıştır. Matlab programı ile yapılan çalışmalarda, her biri 5 açıklıklı, açıklık genişliği 6 metre, kat yüksekliği 3 metre olan 10, 25, 50, 75 ve 100 katlı çerçeve yapı modeli incelenmiştir (Çizelge 4.1).

Çizelge 4.1 : İncelenen yapılar hakkında bilgiler.

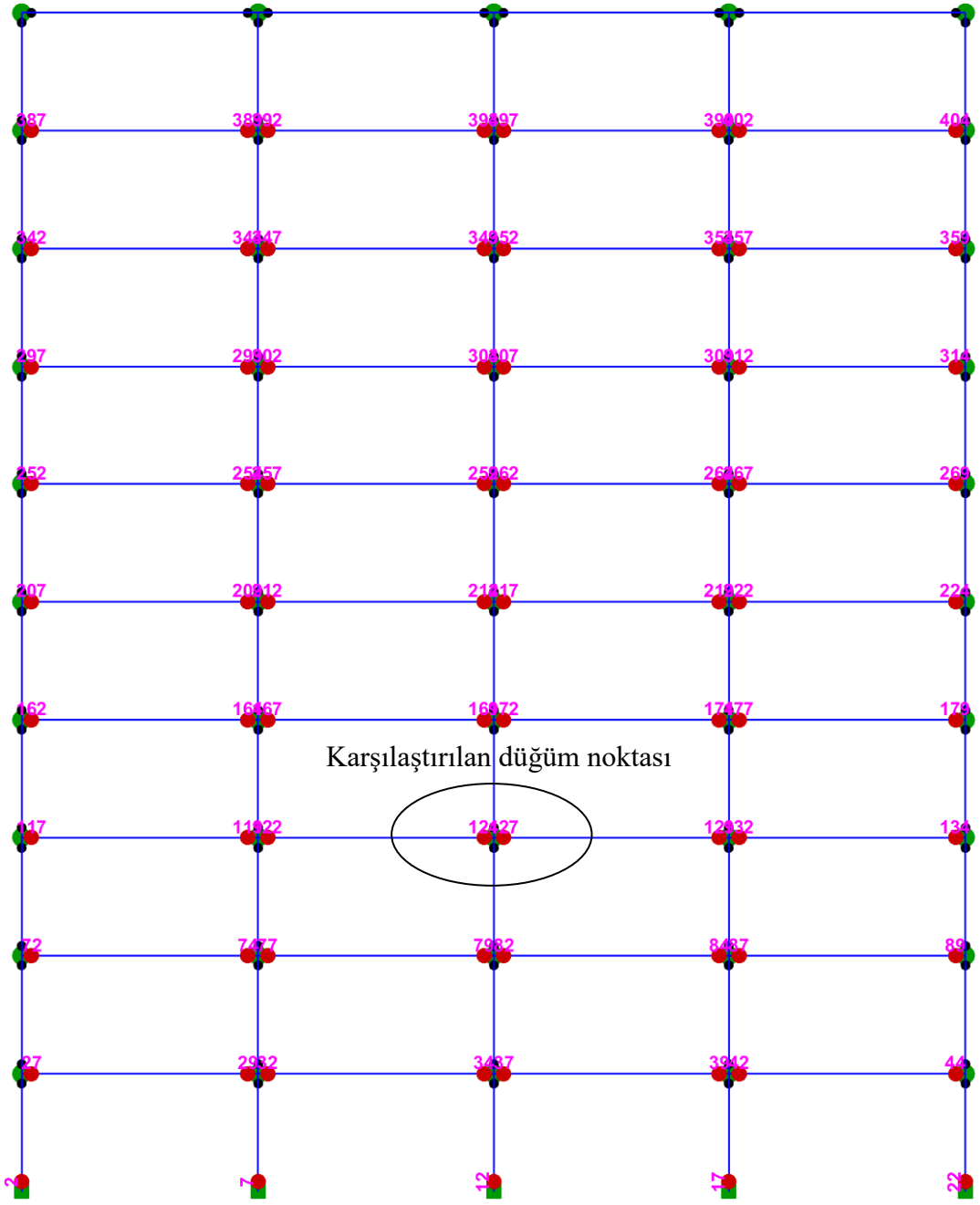
Kat Sayı	Kat Aralığı	Çubuk Elemanlar		Yay Elemanlar					
				Kolon Yay Özellikleri			Kiriş Yay Özellikleri		
		Kolon Boyutları (cm)	Kiriş Boyutları (cm)	k1	k2	Fy	k1	k2	Fy
10	0-10	50 x 50	50 x 40	564764	6992	614	101832	1664	204
25	0-10	80 x 80	50 x 40	2408280	53625	2609	101832	1664	204
	11-20	60 x 60	50 x 40	979850	22990	1114	101832	1664	204
	21-25	50 x 50	50 x 40	564764	6992	614	101832	1664	204
50	0-10	120 x 120	50 x 40	8158218	288628.3	9155	101832	1664	204
	11-20	100 x 100	50 x 40	5022212	126240	5268	101832	1664	204
	21-30	85 x 85	50 x 40	2943367	70301.18	3225	101832	1664	204
	31-40	70 x 70	50 x 40	1500820	34745.71	1741	101832	1664	204
	41-50	50 x 50	50 x 40	564764	6992	614	101832	1664	204
75	0-10	150 x 150	50 x 40	15106199	536410.7	17425	101832	1664	204
	11-20	135 x 135	50 x 40	10571945	359188.1	12079	101832	1664	204
	21-30	120 x 120	50 x 40	8158218	288628.3	9155	101832	1664	204
	31-40	105 x 105	50 x 40	5457429	135436.2	5919	101832	1664	204
	41-50	90 x 90	50 x 40	3553944	81722.22	3752	101832	1664	204
	51-60	75 x 75	50 x 40	2006157	43600	2167	101832	1664	204
	61-70	60 x 60	50 x 40	979850	22990	1114	101832	1664	204
	71-75	50 x 50	50 x 40	564764	6992	614	101832	1664	204
100	0-10	190 x 190	50 x 40	30151722	1079758	35513	101832	1664	204
	11-20	175 x 175	50 x 40	22396216	781683.4	26044	101832	1664	204
	21-30	160 x 160	50 x 40	16932471	586438.8	20035	101832	1664	204
	31-40	145 x 145	50 x 40	13140236	457092.4	15538	101832	1664	204
	41-50	130 x 130	50 x 40	9132451	305766.2	10812	101832	1664	204
	51-60	115 x 115	50 x 40	6861433	177859.1	7804	101832	1664	204
	61-70	100 x 100	50 x 40	5022212	126240	5268	101832	1664	204
	71-80	85 x 85	50 x 40	2943367	70301.18	3225	101832	1664	204
	81-90	70 x 70	50 x 40	1500820	34745.71	1741	101832	1664	204
	91-100	50 x 50	50 x 40	564764	6992	614	101832	1664	204

Yapı kütlesi, bina ölü ağırlığına 30kN/m hareketli yük etkisi eklenerek hesaplanmıştır. Kolon kesitleri eksenel yüklerin kesit kapasitesinin yaklaşık olarak %35'ine denk gelecek şekilde belirlenmiştir. Kiriş kesitleri hareketli ve ölü yük etkisi altında güvenli tarafta kalacak şekilde belirlenmiştir. Donatı oranları yaklaşık olarak kolonda % 1.2, kirişlerde % 0.7'dir. Beton sınıfı C40, çelik sınıfı S420a'dır. XTRACT (2016) programı kullanılarak, her kesit için moment-eğrilik ilişkisi elde edilmiştir. Bu ilişkilerden yaklaşık olarak çift doğrusal idealleştirilmiş davranışlar belirlenmiştir (bknz Şekil 4.7). Dönme yayı elde edilirken doğrusal olmayan bölgenin boyunun eğilmeye çalışan kesitin yarısı olarak alınmıştır. Bu uzunluk kullanılarak moment-eğrilik ilişkisinden moment-dönme ilişkisine geçilmiştir. Bu modellerin analizleri için MATLAB'da program geliştirilmiştir.

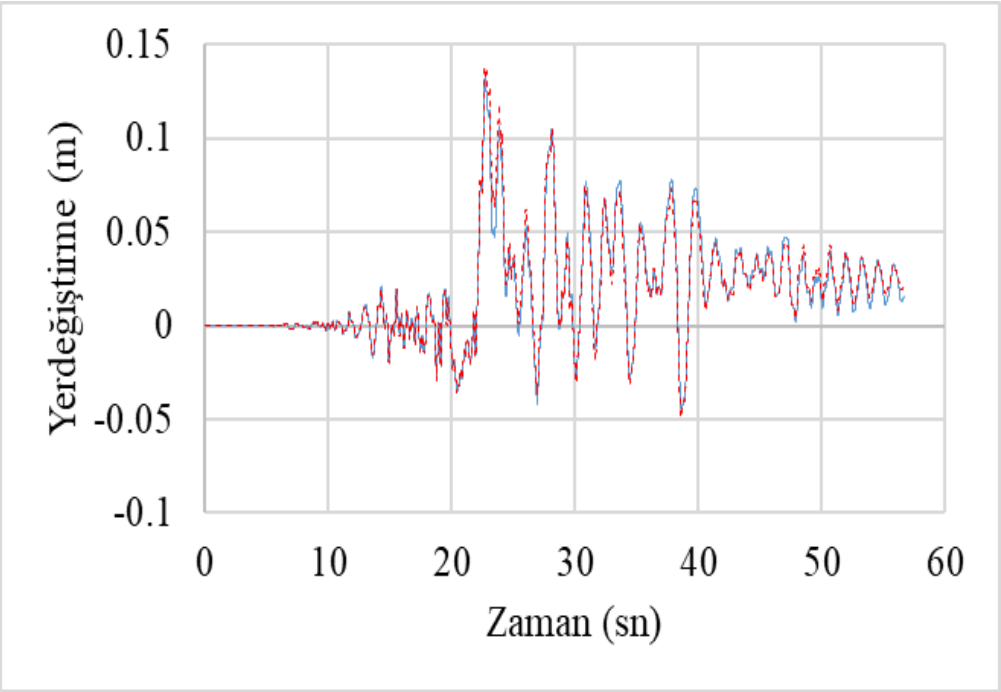


Şekil 4.7 : Xtract kesit analizi sonuçları.

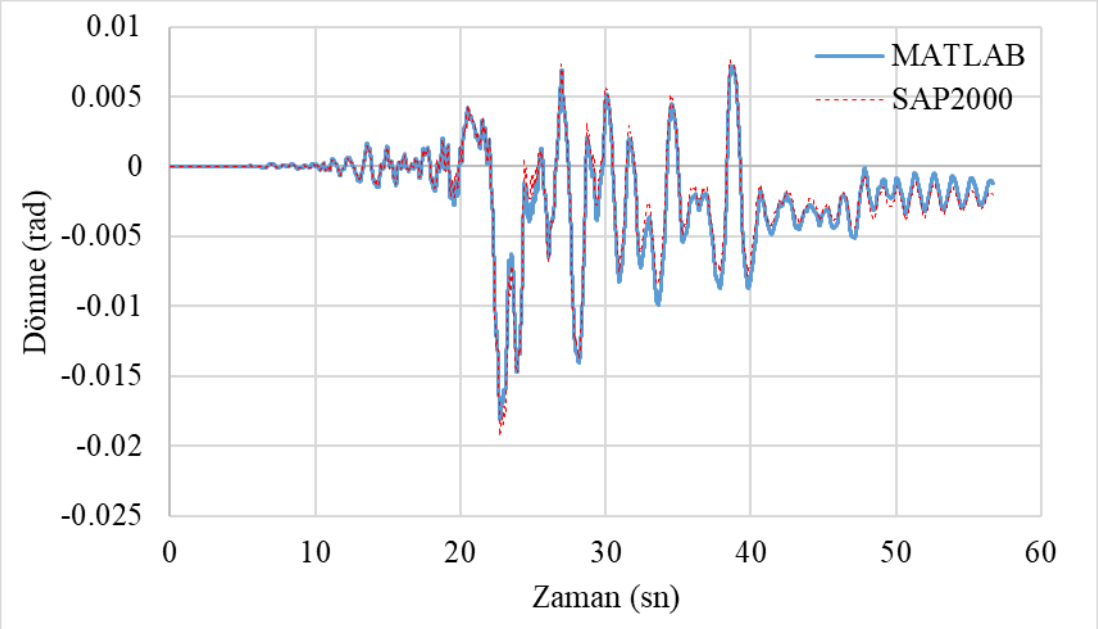
Geliştirilen programın doğruluğu hem doğrusal hem de doğrusal olmayan analizlerde 10 katlı örnek bir yapı (Şekil 4.8) için doğruluğu genel kabul görmüş SAP2000 (2016), programı ile kontrol edilmiştir. SAP2000 modelin rijit-plastik mafsallar kullanılmış, benzer modeller olmaları için yazılan programda dönme yaylarının ilk rijitlikleri çok yüksek alınmıştır. Burada sadece doğrusal olmayan analiz kontrolünün sonuçları verilmiştir (Şekil 4.9 ve Şekil 4.10).



Şekil 4.8 : 10 katlı örnek yapı.



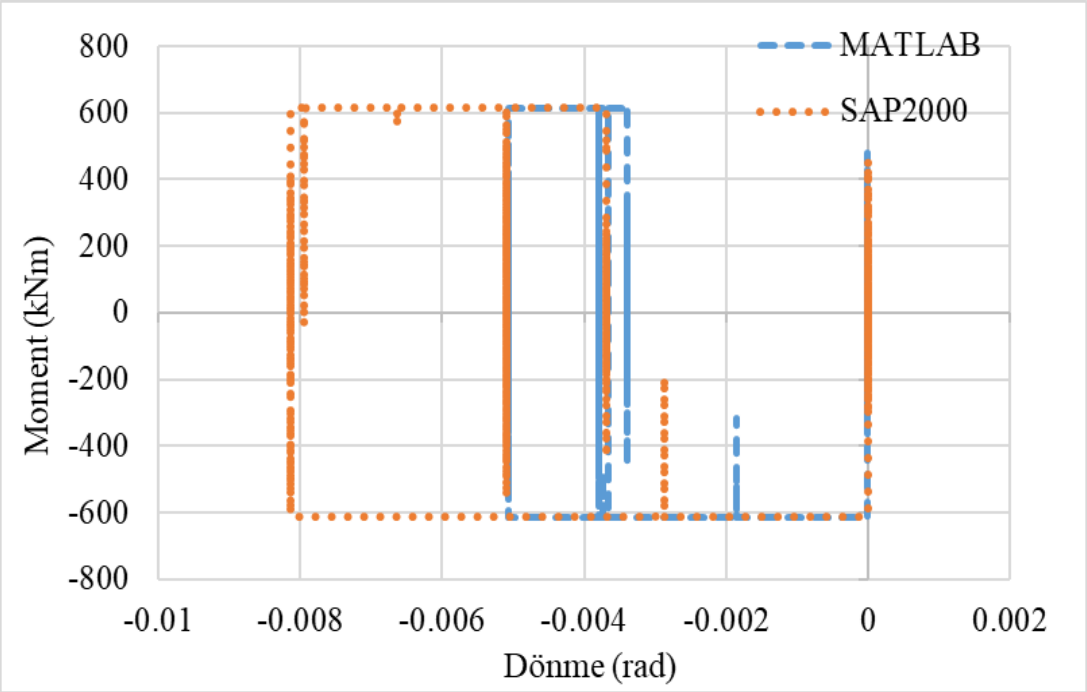
Şekil 4.9 : Karşılaştırılan noktanın (bknz Şekil 4.8) yatay yerdeğiştirmesi.



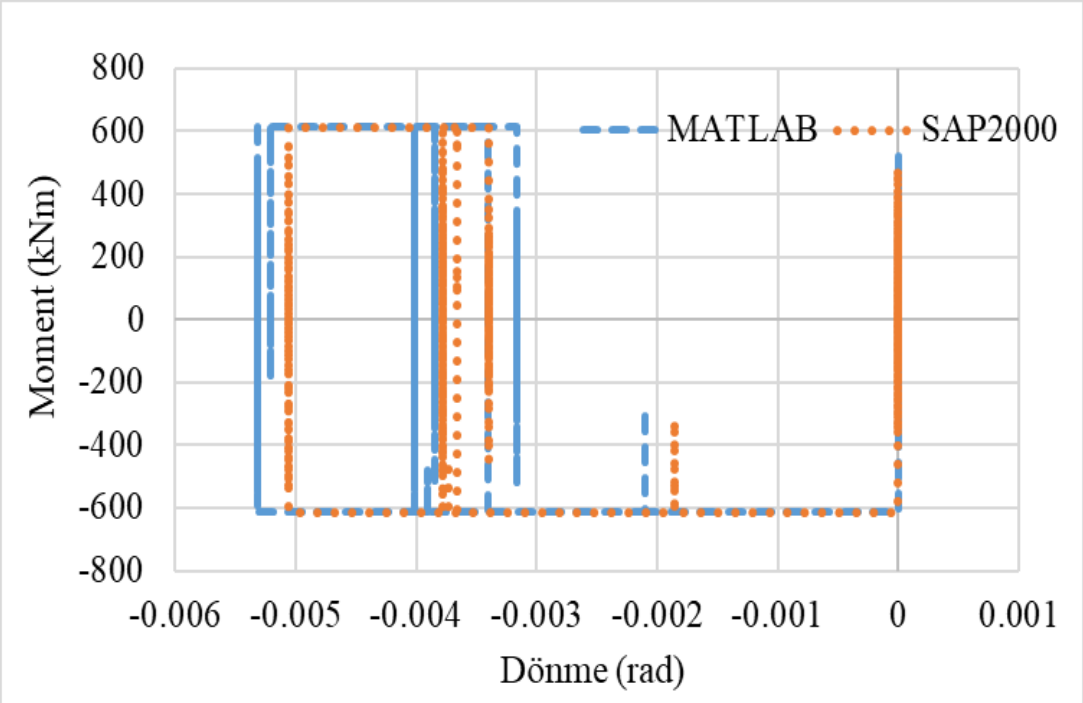
Şekil 4.10 : Karşılaştırılan noktanın (bknz Şekil 4.9) dönmesi.

Görüldüğü üzere, yazılan program ile SAP2000 sonuçları benzer sonuçlar vermektedir. İki sonuç arasındaki farkın, kullanılan doğrusal olmayan eleman modellerindeki farklılıktan kaynaklandığı düşünülmekte olup, eleman parametrelerinin kalibrasyonu ile sonuçlar daha benzer hale getirilebilir. Ancak, bu makalede amaç bünye fonksiyonlarının paralelleştirilmesi olduğundan, ileri kalibrasyona gerek duyulmamıştır.

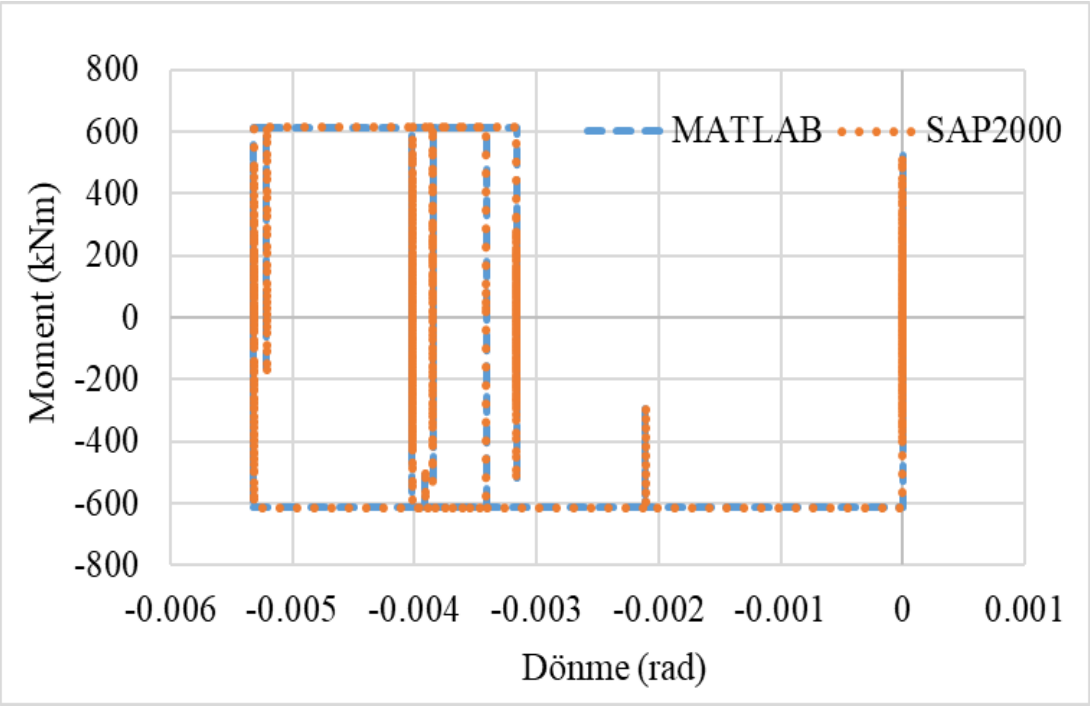
Şekil 4.8’ de verilen çerçeve sistemde en alt kolonların alt seviyelerinde bulunan 2,7,12,17,22 numaralı çift-doğrusal elemanlar için histeritik davranış elde edilmiş ve Sap2000 ile kıyaslanmıştır.



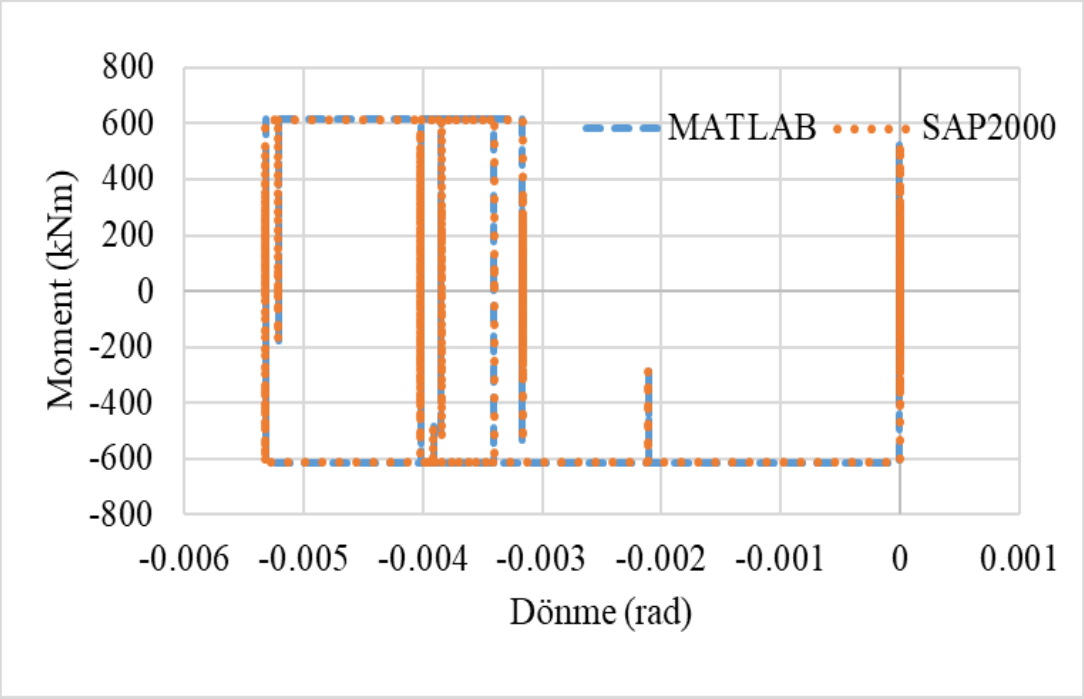
Şekil 4.11 : 2 numaralı çift-doğrusal eleman için moment-dönme ilişkisi.



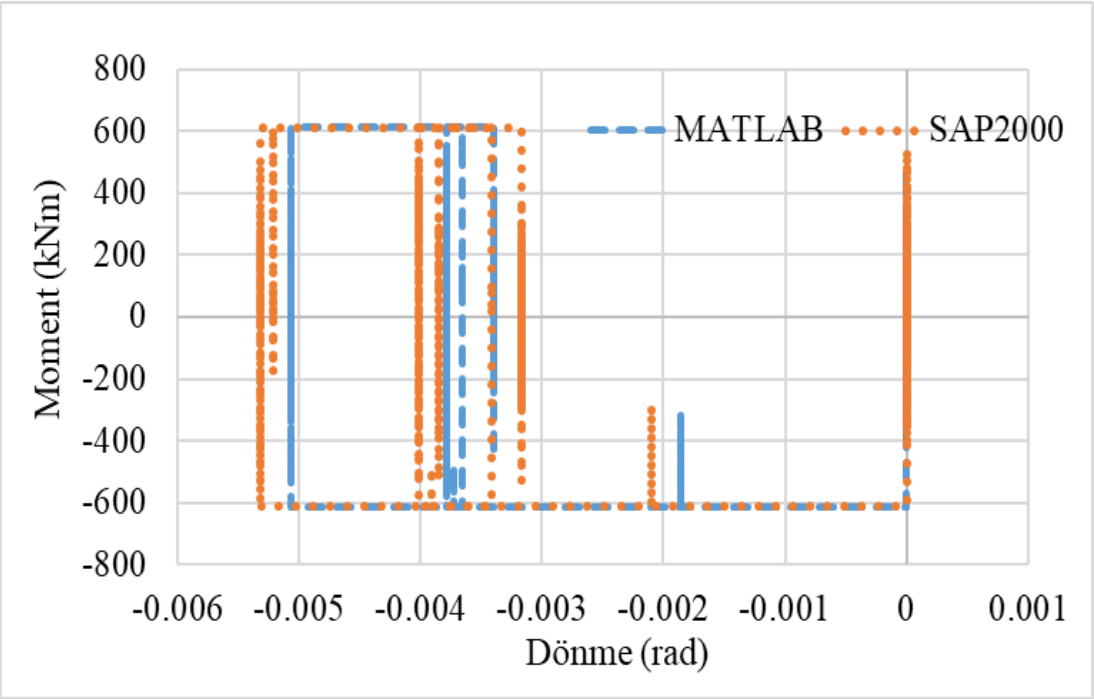
Şekil 4.12 : 7 numaralı çift-doğrusal eleman için moment-dönme ilişkisi.



Şekil 4.13 : 12 numaralı çift-doğrusal eleman için moment-dönme ilişkisi.



Şekil 4.14 : 17 numaralı çift-doğrusal eleman için moment-dönme ilişkisi.



Şekil 4.15 : 22 numaralı çift-doğrusal eleman için moment-dönme ilişkisi.

5. MATLAB PARALEL PROGRAMLAMA

Bu tez kapsamında Matlab programının ‘parfor’ ve ’spmd’ fonksiyonları kullanılarak paralelleştirme çalışmaları yapılmıştır. Doğrusal olmayan analiz esnasında kullanılan çift-doğrusal bünye fonksiyonun paralel programlama ile geliştirilmesi düşülmüştür.

5.1. Parfor ile Çift-Doğrusal Elemanı Paralelleştirme Çalışması

İç içe döngülerden oluşan doğrusal olmayan analizi temsil edebilecek bir algoritme oluşturulmuş ve çift-doğrusal bünye fonksiyonunun bulunduğu bölüm ‘parfor’ yazılarak paralelleştirilmiştir. Fakat doğal ek süre etkileri çok büyüktür. Çift-doğrusal fonksiyonun kısa ve basit bir fonksiyon olmasından dolayı doğal ek süreyi parfor ile yenebilmesi pek mümkün olmamıştır.

Parfor ile paralelleştirme çalışmasından aşağıda bir kod bloğu paylaşılmıştır.

```
aa = 1;
bb = 3;
cc = 4;
mm = (aa:bb:cc)';
lm = length(mm);
rn = 2;
iteration=1e4;
ntstart=1;
dnt = 500;
ntend = 501;
nt = (ntstart: dnt: ntend)';
lnt = length(nt);
plotind = 1;

%PARALEL with n-time step

for j = 1:lm
    vec=rn*rand(mm(j),1);
    lv = length(vec);

    res = zeros(lv,1);

    for nn = 1:lnt
        start=tic;
        for k = 1:nt(nn)
            parfor i=1:length(vec)
```

```
                res(i) = parallel_func(vec(i),iteration);
            end
        end
        time(j,nn)=toc(start);
        disp(['Vector Size = ', num2str(mm(j)),', dt Size
= ', num2str(nt(nn)), ', Time = ', num2str(time(j,nn)),
'secs']);
    end
end
```

5.2. Spmd ile Çift-Doğrusal Elemanı Paralelleştirme Çalışması

Paralel işleme yönteminde kullanılan 'spmd' ile çift-doğrusal fonksiyonun çalıştırılması 'parfor' fonksiyonuna göre daha serbest bir şekilde yapılabilmektedir. 'spmd' bloğu bütün çekirdeklere erişim imkanı sağlamaktadır. Çift-doğrusal eleman sayısını çekirdek sayısına bölerek her elemana işin belirli bir kısmı paylaştırılmıştır. Çekirdekler kendi hafızalarına yönlendirilen verileri eşzamanlı olarak işlemişlerdir. Fakat paralel programlamanın doğası gereği parfor fonksiyonundaki gibi 'spmd' bloğunda da bir doğal ek süre ortaya çıkmıştır.

Spmd ile paralelleştirme çalışmasından aşağıda bir kod paylaşılmıştır.

```
%% Bilinear element
tic
ini=1e6;
del=1e6;
last=10e6;
num_ele = ini:del:last;
% num_ele = last;
for j=1:length(num_ele)
% Input
rotation = rand(num_ele(j),1);
fs = rand(num_ele(j),1);
delta_rotation = rand(num_ele(j),1);
k_1 = ones(num_ele(j),1)*1000;
k_2 = ones(num_ele(j),1)*100;
f_y = ones(num_ele(j),1)*100;
% Output
delta_fs_eif=zeros(num_ele(j),1);
k_t_new=zeros(num_ele(j),1);
id_spring_yield=zeros(num_ele(j),1);
% ANALYSIS
% Seri
start = tic;
for i=1:num_ele(j)
```

```
[delta_fs_eif(i,1),k_t_new(i,1),id_spring_yield(i,1)]=bil
in_for(rotation(i),fs(i),delta_rotation(i),k_1(i),k_2(i),
f_y(i));
end
time_seri(j) = toc(start);
% Parallel
spmd
rotation_p=rotation((labindex-
1)*(num_ele(j)/4)+1:labindex*(num_ele(j)/4));
fs_p=fs((labindex-
1)*(num_ele(j)/4)+1:labindex*(num_ele(j)/4));
delta_rotation_p=delta_rotation((labindex-
1)*(num_ele(j)/4)+1:labindex*(num_ele(j)/4));
k_1_p=k_1((labindex-
1)*(num_ele(j)/4)+1:labindex*(num_ele(j)/4));
k_2_p=k_2((labindex-
(num_ele(j)/4)+1:labindex*(num_ele(j)/4));
f_y_p=f_y((labindex-
(num_ele(j)/4)+1:labindex*(num_ele(j)/4));

start=tic;
for i=1:num_ele(j)/4
fs_upper_limit  =  f_y_p(i)*(1-
k_2_p(i)/k_1_p(i))+k_2_p(i)*(rotation_p(i)+delta_rotation
_p(i));

fs_bottom_limit = -f_y_p(i)*(1-
k_2_p(i)/k_1_p(i))+k_2_p(i)*(rotation_p(i)+delta_rotation
p(i));
fs_k_1   =  fs_p(i)+k_1_p(i)*delta_rotation_p(i);

if     fs_k_1>fs_upper_limit;
fs_spring=fs_upper_limit;
k_t_new_composite_3(i,1)=k_2_p(i);
id_spring_yield_composite_3(i,1)=1;
elseif fs_k_1<fs_bottom_limit;
fs_spring=fs_bottom_limit;
k_t_new_composite_3(i,1)=k_2_p(i);
id_spring_yield_composite_3(i,1)=1;
else;                              fs_spring=fs_k_1;
k_t_new_composite_3(i,1)=k_1_p(i);
id_spring_yield_composite_3(i,1)=0;
end

delta_fs_eif_p(i,1)=fs_spring-fs_p(i);

end
paralel=toc(start);
end
```

```
time_par(j)=max([paralel{1},paralel{2},paralel{3},paralel
{4}]);
speedup(j)=time_seri/time_par;
end

toc
```

5.3. Matlab Programında Doğal Ek Süre

Bu çalışmada MATLAB programlama dilinin "parfor" ve "spmd" fonksiyonları kullanılmıştır. "parfor", çalışma süresi fazla bir fonksiyonun bir döngü içinde çok defa çağrılması durumunda etkili olmakta (örnek: Monte Carlo similasyonu) ve zaman bakımından kazanım sağlamaktadır. Ancak, içiçe döngülerin olduğu durumlarda doğal ek süre (overhead) nedeni ile tüm süre açısından 'parfor' fonksiyonunun etkinliği azalmaktadır. İç içe olan döngülerde, 'spmd' fonksiyonu daha faydalı olabilmektedir. Tek işlem çok veri olduğu durumlarda bu fonksiyon kullanılarak veri parçalara ayrılabilir ve aynı işlem farklı işlemcilerde farklı veriler için eş zamanlı çalıştırılarak paralel programlama yapılabilir. Ancak algoritma uygun değilse, burada da en büyük zorluk doğal ek süre etkisi olabilmektedir.

Paralel programlamda karşılaşılan bir konu verinin işlenmesi için gerekli olan süreden farklı olarak paralel programlamanın doğasında olan işlemlerden dolayı oluşan ek süredir. Örnek olarak MATLAB paralel programlama için istemci (client) hafıza bölgesinden işçi (worker) tarafında veri aktarmaktadır. Bu aktarım için gerekli olan süre doğal ek süredir (overhead). Bu aktarım içiçe iki döngü nedeni ile birden fazla yapılması gerekiyorsa, doğal ek süre döngü sayısı kadar artacaktır ve paralel işlem ile kazanılan süreyi perdeleyebilir. Matemetiksel olarak paralel ve seri işlem süreleri ve doğal ek süre ile ilgili koşul şu şekilde gösterilebilir:

$$t_p = t_{des} + t_{is}(n_{is}/n_ç) \tag{5.1}$$

$$t_s = n_{is} t_{is} \tag{5.2}$$

$$t_p < t_s \tag{5.3}$$

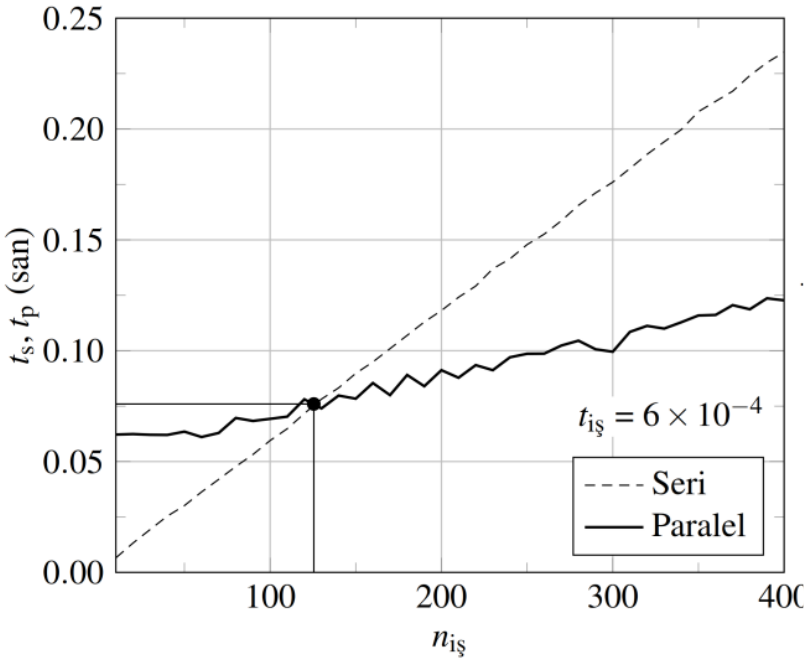
Burada, t_p paralel işlem süresi, t_s seri işlem süresi, t_{des} paralel işleme özel doğal ek süre, $n_{iş}$ işlem adeti ve $n_ç$ çekirdek ya da işlemci sayısıdır.

5.4. Paralel programlama için analizler ve değerlendirilmesi

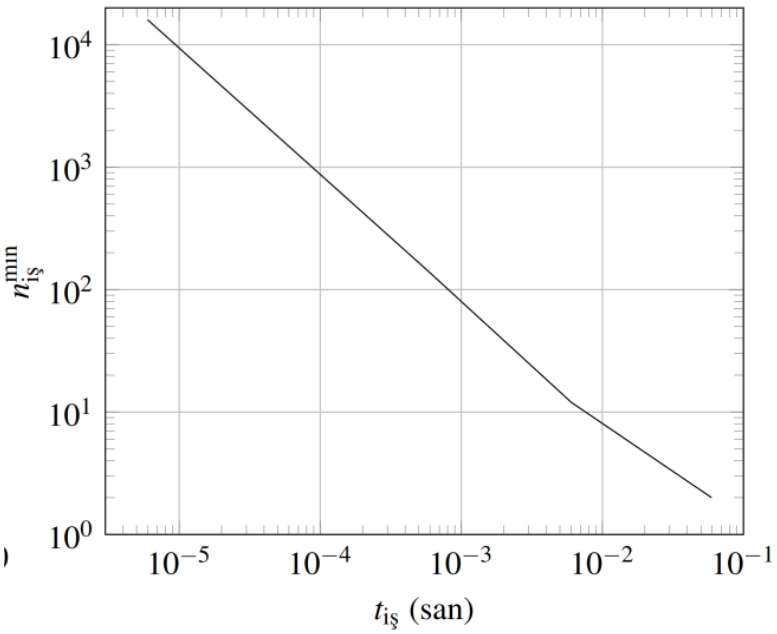
Bu bölümde MATLAB programında parfor ve spmd yaklaşımları için doğal ek süre çalışması yapılmıştır.

Yapıların DOAlerinin programlanmasında genel de iç-içe olan iki ana döngü bulunmaktadır (bknz Şekil 3.19). Bunlardan ilki zaman artım döngüsü (1. Döngü) diğer dengelenmemiş kuvvet döngüsünü azaltan döngüdür (2. Döngü). İkinci döngü içinde bünye fonksiyonlarının her eleman için çağrıldığı üçüncü bir döngü bulunmaktadır (3. Döngü). Bu çalışmada, ikinci döngü, kullanılan dengelenmemiş kuvvet düzeltme yöntemi nedeni ile mevcut değildir ve sadece 1. ve 3. Döngüler mevcuttur. MATLAB programında 3. Döngü paralelleştirildiği zaman 1. Döngünün her adımında 3. Döngü için doğal ek süre oluşturmaktadır. Bundan dolayı, paralelleştirme işleminin seri işlemden daha hızlı olabilmesi için kullanılması gereken en az doğrusal olmayan eleman sayısının bilinmesi gerekir. Bu sayıyı bulmak için parfor ve spmd için iki ayrı çalışma yapılmıştır. Bu çalışmalar çekirdek sayısı 4 olan bir bilgisayarda gerçekleştirilmiştir.

parfor için yapılan çalışmada farklı işlem süresi verebilen bir fonksiyon hazırlanmış, her bir işlem süresi için gerekli olan minimum çağrı sayısı belirlenmiştir. Bunun için ilk önce, verilen bir fonksiyon için seri çalışma hızını yakalayan fonksiyon çağrım sayısı belirlenmiştir (bknz Şekil 5.1). Bu sayı parfor da doğal ek süreyi yenmek için gerekli olana eleman sayısına denk gelmektedir. Bu çalışma farklı çağrı sürelerine sahip birçok fonksiyon için yapılmış ve minimum eleman sayı grafiği elde edilmiştir (Şekil 5.2). MATLAB parfor yaklaşımı için bünye fonksiyonlarının çalışma sürelerinin ve doğal ek süreyi aşmak için gerekli minimum eleman sayısı belirlenmiştir.



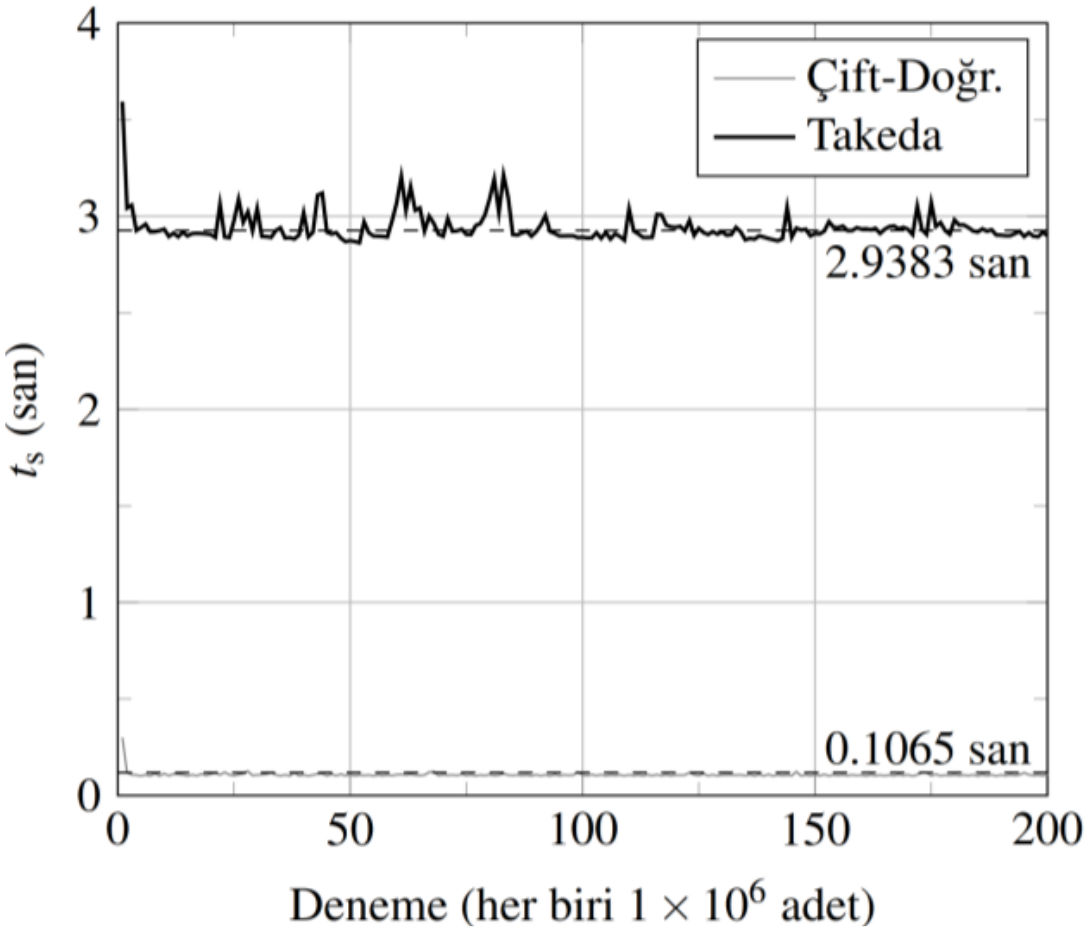
Şekil 5.1 : Minimum eleman sayısının bulunması.



Şekil 5.2 : Minimum eleman sayı grafiği.

Örnek olması açısından çift-doğrusal ve Takeda fonksiyonları için işlem süreleri olarak hesaplanmıştır (Şekil 5.3). Bu hız değerlerine göre parfor ile paralel programlamanın seri programlama ile aynı hızda ya da daha hızlı olması için yapıda yaklaşık olarak en az 10^6 adet çift doğrusal eleman veya en az 3×10^4 adet Takeda elemanı olması gerekmektedir. Bu değerler, minimum eleman sayı grafiğinin dış değerlemesi ile bulunmuştur. Bunun nedeni, parfor fonksiyonu nedeni ile oluşan doğal ek sürenin yüksek olmasıdır ki bu, parfor fonksiyonunun çift doğrusal ve

Takeda elemanlarına çok uygun olmadığı anlamına gelmektedir. Bunun bir nedeni, MATLAB programının if else yapısına sahip parçalı-doğrusal fonksiyonları çok hızlı çağırması olabilir.

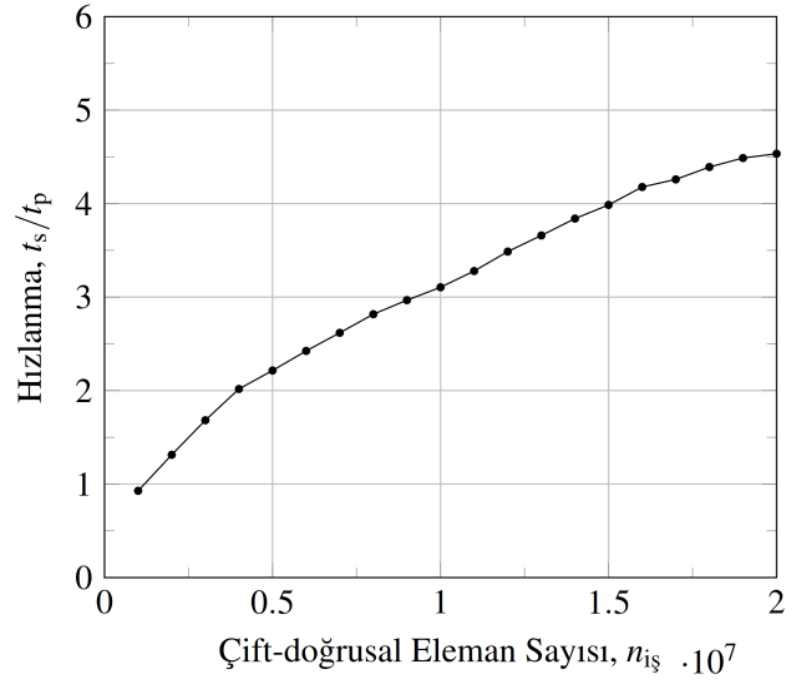


Şekil 5.3 : Bünye fonksiyonları çalışma süreleri.

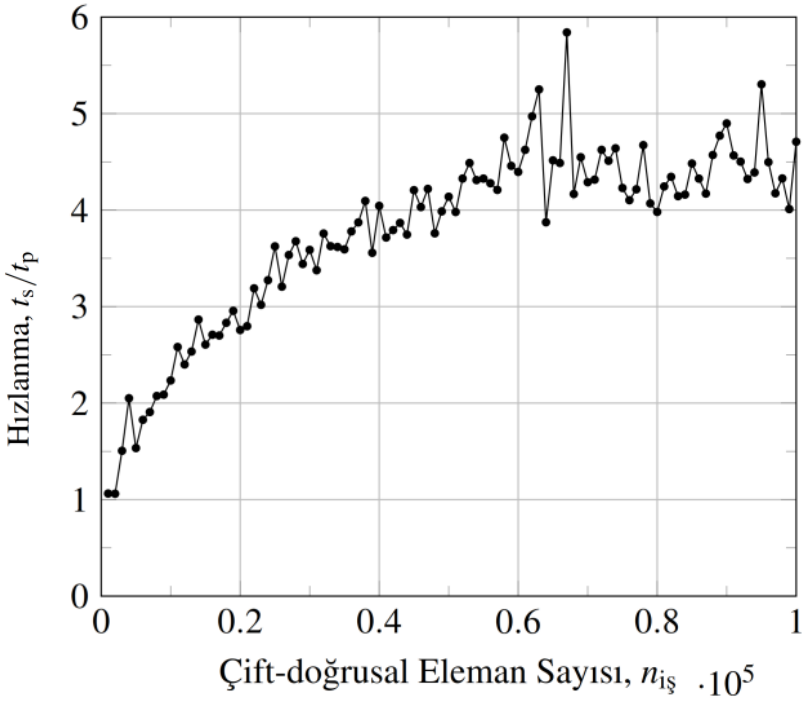
İkinci olarak spmd için yapılan çalışmada, MATLAB programının çift-doğrusal ve Takeda elemanları çok hızlı çalıştırmasından dolayı spmd çerçeve yapısını analiz eden programında çağrılmamıştır; bu fonksiyonların çerçeve yapısında çağrılabilmeleri için yapının çok büyük ölçekli olması gerekmektedir.

Örnek olması için çift-doğrusal fonksiyonu spmd döngüsü içerisinde programlanmış ve seri çağrıma göre hızlanması incelenmiştir (Şekil 5.4 ve Şekil 5.5). Burada bünye fonksiyonlarının paralel ve seri çağrım süreleri hesaplanmış ve seri çağrım için gerekli süre paralel çağrım süresine bölünerek hızlanma değerleri elde edilmiştir. Sadece fonksiyon çağrımında spmd ile hızlanma yaklaşık olarak 10^6 adet çift doğrusal ile başlamıştır ve eleman sayısı arttıkça hızlanmada artmıştır (Şekil 5.4). Bu sonuçlar parfor yönteminin aksine, spmd ile paralel işlemin etkin bir şekilde gerçekleştiğini göstermektedir. Benzer şekilde çift-doğrusal elamanlar ile statik itme

bir simülasyonu gerçekleştirilmesi durumunda hızlanma elde edilmiştir (Şekil 5.5). Statik simülasyonlarda gerekli olan minimum eleman sayısının daha az olduğu gözlemlenmiştir.



Şekil 5.4 : Sadece fonksiyon çağırımında çift doğrusal eleman için hızlanma değerleri.



Şekil 5.5 : Statik simülasyonda çift doğrusal eleman için hızlanma değerleri.

6. SONUÇ VE ÖNERİLER

Bu makalede, yapıların zaman-tanım alanında doğrusal olmayan deprem analizlerinde bünye fonksiyonlarının paralelleştirilmesi araştırılmıştır. Bu amaçla MATLAB betik dili programlama dilleri olarak seçilmiştir. Bünye fonksiyonları olarak çift-doğrusal eleman ve Takeda modeli için hazırlanmış fonksiyonlar kullanılmıştır.

MATLAB dilinde bünye fonksiyonlarının çok hızlı çalışmalarından dolayı yapı analiz programında bünye fonksiyonlarının paralelleştirilmesi programlanmamıştır. MATLAB dilinde iki paralelleştirme yöntemi incelenmiştir.Parfor yönteminde özellikle doğal ek süre (overhead)'den dolayı doğrusal olmayan analizlerin algoritmasında bulunan iç içe döngü akışına uygun olmadığı belirlenmiştir. Doğal ek süreyi yenmek için gerekli olana minimum bünye fonksiyon sayılarını çift-doğrusal ve Takeda modelleri için belirlenmiştir. Diğer yöntem olan spmd yönteminde paralelleştirme gerçekleştirilmiştir ve sadece fonksiyon çağrımı için ve statik itme tipi analiz için hızlanma miktarları eleman sayısına bağlı olarak belirlenmiştir.

KAYNAKLAR

Abrahamson, N. A., & Kaye, M. S. (1997). "Overview." *Seismological Research Letters* 68, no. 1: 9.

Anderson, J. P., Samuel A. H., Joseph, S., & Robert J Williams. (1962). "D825-a Multiple-Computer System for Command & Control." Paper presented at the Proceedings of the December 4-6, 1962, fall joint computer conference.

Ben-Israel, A. (1966). "A Newton-Raphson Method for the Solution of Systems of Equations." *Journal of Mathematical Analysis and Applications* 15, no. 2: 243-52.

Celep, Z. (2007). "Betonarme Sistemlerde Doğrusal Olmayan Davranış : Plastik Mafsal Kabulü Ve Çözümleme." *Altıncı Ulusal Deprem Mühendisliği Konferansı, 16-20 Ekim 2007, İstanbul*.

Dizon, A., & Bermie, R. (2016). "A Hybrid-Parallel Framework for the Nonlinear Seismic Analysis of Very Tall Buildings." California Institute of Technology.

Erkus, B. (2004). "Comparison of the Techniques Used in the Newmark Analysis of Nonlinear Structures." Paper presented at the 17th ASCE Engineering Mechanics Conference.

Flynn, M. J. (1966). "Very High-Speed Computing Systems." *Proceedings of the IEEE* 54, no. 12: 1901-09.

Fung, L. (1983). "Massively Parallel Processor Computer." Google Patents.

Gill, S. (1958). "Parallel Programming." *The Computer Journal* 1, no. 1 : 2-10.

Karaduman, A. (1993). "Değişken Kesitli Düzlem Taşıyıcı Sistemlerin Matris Deplasman Yöntemi Ile Statik Çözümü.".

Mckenna, F. T. (1997). "Object-Oriented Finite Element Programming: Frameworks for Analysis, Algorithms and Parallel Computing." Ph.D. Thesis, University of California, Berkeley.

McKenna, F., & Gregory L. F. (2008). "Using the Opensees Interpreter on Parallel Computers." *Network for earthquake engineering simulations* .

Medina, R. A., & Helmut, K. (2003). "Seismic Demands for Nondeteriorating Frame Structures and Their Dependence on Ground Motions.".

Newmark, N. M. (1959). "A Method of Computation for Structural Dynamics." *Engineering Mechanics Division* .

PEER. (2010). "Modeling and Acceptance Criteria for Seismic Design and Analysis of Tall Buildings.".

Powell, G. H. (1973). Berkeley University of California, and Center Earthquake Engineering Research. *Drain-2d User's Guide.* [in English] Berkeley: Earthquake Engineering Research Center, University of California.

Punzo, G., Federico, M., & Sauro, S. (1994). "High-Resolution Lattice–Boltzmann Computing on the Ibm Sp1 Scalable Parallel Computer." *Computers in Physics* 8, no. 6 : 705-11.

Rayleigh, L. (1896). "The Theory of Sound.".

University of Canterbury Research. (2011). *Base Isolation and Damping Devices (Rapor No : 2011-02).*

Takeda, T., Mete, A. S., & Norby, N. (1970). "Reinforced Concrete Response to Simulated Earthquakes."

Wilson, G. V. (1994). "The History of the Development of Parallel Computing.".

Url-1 <http://www.deprem.gov.tr/>, erişim tarihi 01.08.2017.

Url-2 <www.mathworks.com>, erişim tarihi 01.08.2017.

Url-3 <https://www.intel.com>, erişim tarihi 01.08.2017.

Url-4 <https://www.nag.com/>, erişim tarihi 01.08.2017.

Url-5 <http://ngawest2.berkeley.edu/>, erişim tarihi 01.08.2017.

EKLER

EK A : Doğrusal Statik Analiz Programı

```
clear all; close all; clc; format long;

% 1) INPUT

% 1.1) NODES
% a) Location
determinate_coordinates=1;          % Determination of
coordinates system (please write '0' for manual or '1'
for automatic)
% a1) Manual
coordinates_of_nodes_manual=[0 0;6 3;14 3;22 0;22 6;30
6;30 0];

% a2) Automatic
number_of_bay=3;                    % number of bay
span=6;                             % span (m)
number_of_floor=5;                  % number of floor
height=3;                           % height of floor(m)
start_coordinate=[0];             % start point coordinate


% b) Connectivity
determinate_connectivity=1;         % Determination of
combining vector (please write '0' for manual or '1' for
automatic)
% b1) Manual
connectivity_manual=[1 4;4 5;2 6;6 7;7 3];


% c) Restrained degree of freedom
determinate_restrained_DoF=1;       % Determination of
restrained DoF (please write '0' for manual or '1' for
automatic)
% c1) Manual
restrained_DoF_manual={[1 1 2];[2 1 3 2];[4 2];[5 2];[3
2]};


% d) Loading
nodes_load={[21 0 0]};              % (kN)
nodes_load_fontsize=20;
```

```
% e) Support settlement
support_settlement={[1 0 0];[2 0 0 0]};



% 1.2) ELEMENTS
% a) Loading
determinate_loading_vector=0;     % Determination of
loading vector (please write '0' for nothing  or '1' for
point load  or '2' for uniform load or '3' for both point
and uniform load)

% a1) Point Load
point_load=10;                    % (kN)
ratio=0.25;
point_load_fontsize=28;

% a2) Uniform Load
uniform_load=25;                   % (kN/m)
uniform_load_fontsize=14;


% b) Dimensions and Modulus of elasticity
depth=0.5;                           % (m)
width=0.5;                         % (m)
modulus_of_elasticity=30000000;    % (kN/m^2)



% 1.3) LAYOUT SIGNAL
determinate_number_nodes=1;        % Determination of
numbering of nodes   (please write '0' for switch on   or
'1' for switch off)
determinate_dir_element=1;         % Determination of
direction of element (please write '0' for switch on   or
'1' for switch off)
determinate_name_element=1;        % Determination of name
of element      (please write '0' for switch on   or '1'
for switch off)



% 1.4) PLOT OUTPUT
% a) Scale plot
fontsize_mnt=12;
scale_normal=50;
scale_shear=50;
scale_moment=200;
```

```
% b) Element plotting
mnt_of_element=[];



% 2) NODES

% 2.1) LOCATION
% a) Automatic
for i=1:(number_of_floor+1)
coordinates_of_nodes_automatic_cell{i,1}=[linspace(start_
coordinate(1),start_coordinate(1)+number_of_bay*span,numb
er_of_bay+1)' linspace(start_coordinate(5)+(i-
1)*height,start_coordinate(2)+(i-
1)*height,number_of_bay+1)'];
end
coordinates_of_nodes_automatic=cell2mat(coordinates_of_no
des_automatic_cell);

% b) Determinate
if determinate_coordinates==1;
coordinates_of_nodes=coordinates_of_nodes_automatic;
else
coordinates_of_nodes=coordinates_of_nodes_automatic;
end


% c) Sorting and numbering with function
coordinates_sorted_and_numerated=SortingAndNumbering_func
tion(coordinates_of_nodes);



% 2.2) CONNECTIVITY
% a) Automatic
if number_of_floor==0
connectivity_automatic=[[2:number_of_bay]'
[2:number_of_bay+1]'];
else
for i=1:number_of_floor
connectivity_automatic_cell{i,1}=[[(i-
1)*(number_of_bay+1)+1:i*(number_of_bay+1)
i*(number_of_bay+1)+1:i*(number_of_bay+1)+number_of_bay]'
[i*(number_of_bay+1)+1:(i+1)*(number_of_bay+1)
i*(number_of_bay+1)+2:(i+1)*(number_of_bay+1)]'];
end
connectivity_automatic=cell2mat(connectivity_automatic_ce
ll);
end
```

```
% b) Determinate
if determinate_connectivity==1;
connectivity=connectivity_automatic;
else
connectivity=connectivity_manual;
end



% 2.3) RESTRAINED DEGREE OF FREEDOM
% a) Automatic
for i=1:(number_of_bay+1)
restrained_DoF_automatic{i,1}=[i 1 3];
end

% b) Determinate
if determinate_restrained_DoF==1;
restrained_DoF=restrained_DoF_automatic;
else
restrained_DoF=restrained_DoF_manual;
end


% c) Shapes and numbers of DoF restrained
fully_freedom_nodes=coordinates_sorted_and_numerated(:,1)
;

if size(restrained_DoF,1)~=0
for i=1:size(restrained_DoF,1)
restrained_nodes(i,1)=restrained_DoF(1);
end

fully_freedom_nodes(restrained_nodes)=[];

for i=1:size(restrained_nodes,1)
for j=1:3
DoF{restrained_nodes(i),j}=find(restrained_DoF{i}(2:end)=
=j);
end
if       DoF{restrained_nodes(i),1}>0  &
DoF{restrained_nodes(i),2}>0  &
DoF{restrained_nodes(i),3}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='s';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-2; restrained_nodes(i)*3-1;
restrained_nodes(i)*3];
elseif   DoF{restrained_nodes(i),1}>0  &
DoF{restrained_nodes(i),2}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='^';
```

```
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-2; restrained_nodes(i)*3-1];
elseif   DoF{restrained_nodes(i),1}>0  &
DoF{restrained_nodes(i),3}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='p';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-2; restrained_nodes(i)*3];
elseif   DoF{restrained_nodes(i),2}>0  &
DoF{restrained_nodes(i),3}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='h';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-1; restrained_nodes(i)*3];
elseif   DoF{restrained_nodes(i),1}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='>';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-2];
elseif   DoF{restrained_nodes(i),2}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='x';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-1];
elseif   DoF{restrained_nodes(i),3}>0;
restrained_DoF_shapes{restrained_nodes(i),1}='d';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3];
else
restrained_DoF_shapes{restrained_nodes(i),1}='o';
restrained_DoF_number{restrained_nodes(i),1}=[];
end
end
end

for i=1:size(fully_freedom_nodes,1)
restrained_DoF_shapes{fully_freedom_nodes(i)}='o';
end



 % 3) ELEMENTS

cross_section_area=width*depth;          % (Element
Properties -->  Cross section area (m^2))
moment_of_inertia=(width*depth^3)/12;    % (Element
Properties -->  Moment of inertia  (m^4))

for i=1:size(connectivity,1)
% 3.1) PROPERTIES
% a) Angle with respect to global coordinate system
(Degree)
if
coordinates_sorted_and_numerated(connectivity(i,2),2)-
```

```
coordinates_sorted_and_numerated(connectivity(i,1),2)<0;
angle(i,1)=(atan((coordinates_sorted_and_numerated(connec
tivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))/(c
oordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)))/(
2*pi))*360+180;
elseif
coordinates_sorted_and_numerated(connectivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3)<0
&& coordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)==0
|| coordinates_sorted_and_numerated(connectivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3)<0
&& coordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)>0;
angle(i,1)=(atan((coordinates_sorted_and_numerated(connec
tivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))/(c
oordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)))/(
2*pi))*360+360;
else
angle(i,1)=(atan((coordinates_sorted_and_numerated(connec
tivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))/(c
oordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)))/(
2*pi))*360;
end

% b) Lenght  (m)
length_of_element(i,1)=sqrt((coordinates_sorted_and_numer
ated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2))^2+
(coordinates_sorted_and_numerated(connectivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))^2)
;

% c) Local Stiffness matrix
stiffness_of_elements_local{i,1}=[
(modulus_of_elasticity*cross_section_area)/length_of_elem
ent(i)                                          0
0                                       -
(modulus_of_elasticity*cross_section_area)/length_of_elem
ent(i)                                          0
0                                  ;

0
(12*modulus_of_elasticity*moment_of_inertia)/length_of_el
ement(i)^3
```

```
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2                                          0
-
(12*modulus_of_elasticity*moment_of_inertia)/length_of_el
ement(i)^3
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2 ;

0
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2
(4*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)                                            0
-
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2
(2*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)   ;
                                        -
(modulus_of_elasticity*cross_section_area)/length_of_elem
ent(i)                                          0
0
(modulus_of_elasticity*cross_section_area)/length_of_elem
ent(i)                                          0
0                                    ;

0                                     -
(12*modulus_of_elasticity*moment_of_inertia)/length_of_el
ement(i)^3    -
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2                                          0
(12*modulus_of_elasticity*moment_of_inertia)/length_of_el
ement(i)^3    -
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2 ;

0
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2
(2*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)                                            0
-
(6*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)^2
(4*modulus_of_elasticity*moment_of_inertia)/length_of_ele
ment(i)    ];



% d) Transformation  matrix
```

```
transformation_matrix{i,1}=[ cos(angle(i)*((2*pi)/360))
sin(angle(i)*((2*pi)/360))   0                  0
0            0;
                           -sin(angle(i)*((2*pi)/360))
cos(angle(i)*((2*pi)/360))   0                  0
0            0;
                                              0
0              1                 0
0            0;
                                              0
0              0   cos(angle(i)*((2*pi)/360))
sin(angle(i)*((2*pi)/360)) 0;
                                              0
0              0  -sin(angle(i)*((2*pi)/360))
cos(angle(i)*((2*pi)/360)) 0;
                                              0
0              0                 0
0            1  ];


% e) Global Stiffness Matrix
stiffness_of_elements_global{i,1}=
transformation_matrix{i}'*stiffness_of_elements_local{i}*
transformation_matrix{i};



% 3.2)LOADING FOR ALL ELEMENTS
% a) Point Load
first_length(i)=ratio*length_of_element(i);
% (m) parameter of point load for all elements
second_length(i)=length_of_element(i)-first_length(i);
% (m) parameter of point load for all elements
% Local
point_loading_of_elements_local{i,1}=[0;(point_load*secon
d_length(i)^2*(3*first_length(i)+second_length(i)))/lengt
h_of_element(i)^3;(point_load*first_length(i)*second_leng
th(i)^2)/length_of_element(i)^2;0;(point_load*first_lengt
h(i)^2*(first_length(i)+3*second_length(i)))/length_of_el
ement(i)^3;-
(point_load*first_length(i)^2*second_length(i))/length_of
_element(i)^2];


% b) Uniform Load
% Local
% c) Determinate
if      determinate_loading_vector==1;
fix_end_forces_and_moments_of_elements_local{i,1}=point_l
oading_of_elements_local{i,1};
```

```
elseif  determinate_loading_vector==2;
fix_end_forces_and_moments_of_elements_local{i,1}=uniform
_loading_of_elements_local{i,1};
elseif  determinate_loading_vector==3;
fix_end_forces_and_moments_of_elements_local{i,1}=point_l
oading_of_elements_local{i,1}+uniform_loading_of_elements
_local{i,1};
else
fix_end_forces_and_moments_of_elements_local{i,1}=zeros(6
,1);
end


% d) Global fix end forces and moments
fix_end_forces_and_moments_of_elements_global{i,1}=transf
ormation_matrix{i}'*fix_end_forces_and_moments_of_element
s_local{i,1};
end




% 4) SYSTEM

% 4.1) LOADING FOR NODES
nodes_loads_of_frame_system{1}=zeros(3*coordinates_sorted
_and_numerated(end,1),1);d
for i=1:size(nodes_load,1)
nodes_loads_of_frame_system{1}([nodes_load{i}(1)*3-2
nodes_load{i}(1)*3-1
nodes_load{i}(1)*3])=nodes_loads_of_frame_system{1}([node
s_load{i}(1)*3-2 nodes_load{i}(1)*3-1
nodes_load{i}(1)*3])+(nodes_load{i}(2:end))';
end


% 4.2) SUPPORT SETTLEMENT
support_settlement_of_frame_system{1}=zeros(3*coordinates
_sorted_and_numerated(end,1),1);
for i=1:size(support_settlement,1)
support_settlement_of_frame_system{1}([support_settlement
{i}(1)*3-2 support_settlement{i}(1)*3-1
support_settlement{i}(1)*3])=support_settlement_of_frame_
system{1}([support_settlement{i}(1)*3-2
support_settlement{i}(1)*3-1
support_settlement{i}(1)*3])+(support_settlement{i}(2:end
))';
end


% 4.3) STIFFNESS MATRIX
```

```
stiffness_of_frame_system{1}=zeros(3*coordinates_sorted_a
nd_numerated(end,1));
for i=1:size(connectivity,1)
stiffness_of_frame_system{1}([connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3],[connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])=stiffness_of_frame_system{1}([conne
ctivity(i,1)*3-2 connectivity(i,1)*3-1
connectivity(i,1)*3 connectivity(i,2)*3-2
connectivity(i,2)*3-1
connectivity(i,2)*3],[connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])+stiffness_of_elements_global{i};
end


% 4.4) LOADING FOR ALL FRAME ELEMENTS
fix_end_forces_and_moments_of_frame_system{1}=zeros(3*coo
rdinates_sorted_and_numerated(end,1),1);
for i=1:size(connectivity,1)
fix_end_forces_and_moments_of_frame_system{1}([connectivi
ty(i,1)*3-2 connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])=fix_end_forces_and_moments_of_frame
_system{1}*3 connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])+fix_end_forces_and_moments_of_eleme
nts_global{i};
end
ratio_uniform=1*(0:1/6:1);



% 5) LAYOUT

% 5.1) NAMING OF ELEMENT
if size(connectivity,1)==0 name={};
else
name=Naming(size(connectivity,1));
end


% 5.2) CENTER ALINGMENT
if number_of_floor*height >= number_of_bay*span;
xlim_first=start_coordinate(1)+(number_of_bay*span-
number_of_floor*height)/2; xlim_end=
start_coordinate(1)+(number_of_bay*span+number_of_floor*h
```

```
eight)/2; ylim_first=start_coordinate(2);
ylim_end=start_coordinate(2)+number_of_floor*height;
else
xlim_first=start_coordinate(1);
xlim_end=  start_coordinate(1)+number_of_bay*span;
ylim_first=start_coordinate(2)+(number_of_floor*height-
number_of_bay*span)/2;
ylim_end=start_coordinate(2)+(number_of_floor*height+numb
er_of_bay*span)/2;
end



% 6) LINEAR STATIC ANALYSIS FOR FRAME SYSTEM

% 6.1) MATCHING VECTOR FOR OBTAINING GLOBAL MATRIX
% a) All DoF
all_DoF_number=(1:3*coordinates_sorted_and_numerated(end,
1))';

% b) Restrained DoF
restrained_DoF_number_for_matching=cell2mat(restrained_Do
F_number);

% c) Unrestrained DoF
unrestrained_DoF_number_for_matching=all_DoF_number;
unrestrained_DoF_number_for_matching(restrained_DoF_numbe
r_for_matching)=[];

% d) Matching vector for reorganize
matching_vector_first=[unrestrained_DoF_number_for_matchi
ng;restrained_DoF_number_for_matching];

% e) Matching vector for orijinal position
for i=1:length(matching_vector_first)

matching_vector_end(i,1)=find(matching_vector_first==i);
end


% 6.2) REORGANIZE LOADING, STIFFNESS, FIXEND FORCES AND
MOMENT, SUPPORT SETTLEMENT MATRIX
loading_for_nodes_reorganize=nodes_loads_of_frame_system{
1}(matching_vector_first);
stiffness_of_frame_system_global_reorganize=stiffness_of_
frame_system{1}(matching_vector_first,matching_vector_fir
st);
fix_end_forces_and_moments_of_frame_system_reorganize=fix
_end_forces_and_moments_of_frame_system{1}(matching_vecto
r_first);
```

```
support_settlement_of_frame_system_reorganize=support_set
tlement_of_frame_system{1}(matching_vector_first);


% 6.3) LINEAR SOLUTION WITH MATRIX THEORY FOR FRAME
SYSTEM
% a) Input
% a1) Forces
F_A=loading_for_nodes_reorganize(1:length(unrestrained_Do
F_number_for_matching));
F_B_first_part_nodes_loads=loading_for_nodes_reorganize(l
ength(unrestrained_DoF_number_for_matching)+1:end);

% a2) Stiffness
K_BB=stiffness_of_frame_system_global_reorganize([length(
unrestrained_DoF_number_for_matching)+1:end],[length(unre
strained_DoF_number_for_matching)+1:end]);

% a3) Displacements ( Support Settlements )
D_B=support_settlement_of_frame_system_reorganize((length
(unrestrained_DoF_number_for_matching)+1:end));

% a4) Fix end forces and moments
P_A=fix_end_forces_and_moments_of_frame_system_reorganize
(1:length(unrestrained_DoF_number_for_matching));
P_B=fix_end_forces_and_moments_of_frame_system_reorganize
(length(unrestrained_DoF_number_for_matching)+1:end);


% b) Solution
D_A=inv(K_AA)*(F_A-K_AB*D_B-P_A);
F_B_second_part_joint_reactions=K_BA*D_A+K_BB*D_B+P_B-
F_B_first_part_nodes_loads;



% 6.4) ORIJINAL POSITION DISPLACEMENT AND FORCE MATRIX
displacements_of_frame_system_reorganized=[D_A;D_B];
displacements_of_frame_system_orijinal_position=displacem
ents_of_frame_system_reorganized(matching_vector_end);

nodes_loads_both_point_loads_and_joint_reactions{1}=nodes
_loads_of_frame_system{1};
nodes_loads_both_point_loads_and_joint_reactions{1}(restr
ained_DoF_number_for_matching)=nodes_loads_both_point_loa
ds_and_joint_reactions{1}(restrained_DoF_number_for_match
ing)+F_B_second_part_joint_reactions;

% Control
error_control=nodes_loads_both_point_loads_and_joint_reac
tions{1}-
```

```
stiffness_of_frame_system{1}*displacements_of_frame_syste
m_orijinal_position-
fix_end_forces_and_moments_of_frame_system{1};



% 7) FORCES AND MOMENTS OF EACH ELEMENT (M, N, T)

for i=1:size(connectivity,1)
% 7.1) GLOBAL DISPLACEMENTS AND FORCES
% Displacements
displacements_of_elements_global{i,1}=displacements_of_fr
ame_system_orijinal_position([connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3]);

% Forces
forces_of_elements_global{i,1}=stiffness_of_elements_glob
al{i}*displacements_of_elements_global{i}+fix_end_forces_
and_moments_of_elements_global{i};


% 7.2) LOCAL DISPLACEMENTS AND FORCES
% Displacements
displacements_of_elements_local{i,1}=transformation_matri
x{i}*displacements_of_elements_global{i};

% Forces
forces_of_elements_local{i,1}=transformation_matrix{i}*fo
rces_of_elements_global{i};


% 7.3) NEW COORDINATES FOR PLOTTING M, N, T
for j=1:2
start_coordinate_for_mnt{i,j}=[coordinates_sorted_and_num
erated(connectivity(i,j),2)
coordinates_sorted_and_numerated(connectivity(i,j),3)];

angle_deg=angle(i)+90;
angle_rad=(angle_deg/180)*(2*pi);

if     angle_deg>=0   && angle_deg<=90   || angle_deg>=360
&& angle_deg<=450;  sign_conv_x= 1;  sign_conv_y= 1;
elseif angle_deg>90   && angle_deg<=180 || angle_deg>450
&& angle_deg<=540;  sign_conv_x=-1;  sign_conv_y=-1;
elseif angle_deg>180 && angle_deg<=270 || angle_deg>540
&& angle_deg<=630;  sign_conv_x=-1;  sign_conv_y=-1;
elseif angle_deg>270 && angle_deg<360  || angle_deg>630
&& angle_deg<=720;  sign_conv_x= 1;  sign_conv_y= 1;
```

```
end

% Normal
normal{i,j}=forces_of_elements_local{i}((j-
1)*3+1)/scale_normal;
x_2_normal=start_coordinate_for_mnt{i,j}(1)+
sign_conv_x*normal{i,j}*(1/(sqrt(tan(angle_rad)^2+1)));
y_2_normal=start_coordinate_for_mnt{i,j}(2)+
sign_conv_y*normal{i,j}*(tan(angle_rad)/(sqrt(tan(angle_r
ad)^2+1)));
end_coordinate_normal{i,j}=[x_2_normal  y_2_normal];

% Shear
shear{i,j}=forces_of_elements_local{i}((j-
1)*3+2)/scale_shear;
x_2_shear=start_coordinate_for_mnt{i,j}(1)+
sign_conv_x*shear{i,j}*(1/(sqrt(tan(angle_rad)^2+1)));
y_2_shear=start_coordinate_for_mnt{i,j}(2)+
sign_conv_y*shear{i,j}*(tan(angle_rad)/(sqrt(tan(angle_ra
d)^2+1)));
end_coordinate_shear{i,j}=[x_2_shear  y_2_shear];

% Moment
moment{i,j}=forces_of_elements_local{i}((j-
1)*3+3)/scale_moment;
x_2_moment=start_coordinate_for_mnt{i,j}(1)+
sign_conv_x*moment{i,j}*(1/(sqrt(tan(angle_rad)^2+1)));
y_2_moment=start_coordinate_for_mnt{i,j}(2)+
sign_conv_y*moment{i,j}*(tan(angle_rad)/(sqrt(tan(angle_r
ad)^2+1)));
end_coordinate_moment{i,j}=[x_2_moment  y_2_moment];
end

end

% 8) PLOT FRAME SYSTEM

fig1=figure; fig1.Position=[970 45 950 950];
subplot('Position',[0.05 0.05 0.90 0.90]); hold on; grid
on;

% 8.1) NODES
for i=1:size(coordinates_sorted_and_numerated,1)
% a) Plot each node and determinate restrained DoF shapes
plot(coordinates_sorted_and_numerated(i,2),coordinates_so
rted_and_numerated(i,3), restrained_DoF_shapes{i}
,'markersize',10,'markeredgecolor',[0 0.6
0],'markerfacecolor',[0 0.6 0]);

% b) Texting number of nodes
```

```
if determinate_number_nodes==1;
text(coordinates_sorted_and_numerated(i,2),coordinates_so
rted_and_numerated(i,3),num2str(coordinates_sorted_and_nu
merated(i,1)),'horizontalalignment','right','verticalalig
nment','bottom','Color','r','Fontweight','bold','FontSize
',10); end;
end
title({'FRAME SYSTEM',''}); xlabel('Span (m)');
ylabel('Height (m)'); if  (number_of_bay==0 &&
number_of_floor==0) || determinate_coordinates~=1; xlim
auto; ylim auto; else   xlim([xlim_first xlim_end]);
ylim([ylim_first ylim_end]); end;

% c) Texting nodes load
for i=1:size(nodes_load,1)
    if  nodes_load{i}(2)>0;
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['
\rightarrow_{' num2str(abs(nodes_load{i}(2)))
'}'],'horizontalalignment','left','verticalalignment','mi
ddle','color','b','fontweight','bold','fontsize',nodes_lo
ad_fontsize);   elseif  nodes_load{i}(2)<0
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['_
{' num2str(abs(nodes_load{i}(2))) '}\leftarrow
'],'horizontalalignment','right','verticalalignment','mid
dle','color','b','fontweight','bold','fontsize',nodes_loa
d_fontsize);    end;
    if  nodes_load{i}(3)>0;
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['\
uparrow^{' num2str(abs(nodes_load{i}(3)))
'}'],'horizontalalignment','left','verticalalignment','bo
ttom','color','b','fontweight','bold','fontsize',nodes_lo
ad_fontsize);       elseif  nodes_load{i}(3)<0;
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['\
downarrow_{' num2str(abs(nodes_load{i}(3)))
'}'],'horizontalalignment','left','verticalalignment','to
p','color','b','fontweight','bold','fontsize',nodes_load_
fontsize);           end;
    if  nodes_load{i}(4)>0;
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['\
bullet^{' num2str(abs(nodes_load{i}(4))) '}'],
'horizontalalignment','center','verticalalignment','cap',
'color','b','fontweight','bold','fontsize',nodes_load_fon
tsize);        elseif  nodes_load{i}(4)<0;
text(coordinates_sorted_and_numerated(nodes_load{i}(1),2)
,coordinates_sorted_and_numerated(nodes_load{i}(1),3),['\
otimes^{' num2str(abs(nodes_load{i}(4))) '}'],
```

```
'horizontalalignment','center','verticalalignment','cap',
'color','b','fontweight','bold','fontsize',nodes_load_fon
tsize);          end;
end


% 8.2) ELEMENTS
if size(coordinates_of_nodes,1)>=2
for i=1:size(connectivity,1)
% a) Plot each element
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],
coordinates_sorted_and_numerated(connectivity(i,2),3)],'-
','color','b','linewidth',1);

% b) Texting direction of element
if determinate_dir_element==1;
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+coordinates_sorted_and_numerated(connectivity(i,2),2))
/2,(coordinates_sorted_and_numerated(connectivity(i,1),3)
,'\rightarrow','horizontalalignment','center','verticalal
ignment','middle','Color','r','Fontweight','bold','FontSi
ze',12,'rotation',angle(i)); end;

% c) Texting name of element
if determinate_name_element==1;
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+coordinates_sorted_and_numerated(connectivity(i,2),2))
/2,3)+coordinates_sorted_and_numerated(connectivity(i,2),
3))/2,[name{i} '(' num2str(i)
')'],'horizontalalignment','center','verticalalignment','
bottom','Color','m','Fontweight','bold','FontSize',10,'ro
tation',angle(i)); end;

% d) Texting point loading vector
if determinate_loading_vector==1 ||
determinate_loading_vector==3
if point_load<0; angle_for_point_load(i)=angle(i)+270;
else             angle_for_point_load(i)=angle(i)+90; end

if angle_for_point_load(i)>=0 &&
angle_for_point_load(i)<=90 ||
angle_for_point_load(i)>270 &&
angle_for_point_load(i)<=360 ||
angle_for_point_load(i)>=360 &&
angle_for_point_load(i)<=450 ||
angle_for_point_load(i)>630 &&
angle_for_point_load(i)<=720
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+ratio*(coordinates_sorted_and_numerated(connectivity(i
```

```
,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2))),(
coordinates_sorted_and_numerated(connectivity(i,1),3)+rat
io*(coordinates_sorted_and_numerated(connectivity(i,2),3)
-
coordinates_sorted_and_numerated(connectivity(i,1),3))),[
'\leftarrow'
num2str(abs(point_load))],'horizontalalignment','left','v
erticalalignment','middle','Color','r','Fontweight','bold
','FontSize',point_load_fontsize,'rotation',angle_for_poi
nt_load(i));
else
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+ratio*(coordinates_sorted_and_numerated(connectivity(i
,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2))),(
coordinates_sorted_and_numerated(connectivity(i,1),3)+rat
io*(coordinates_sorted_and_numerated(connectivity(i,2),3)
-
coordinates_sorted_and_numerated(connectivity(i,1),3))),[
num2str(abs(point_load))
'\rightarrow'],'horizontalalignment','right','verticalali
gnment','middle','Color','r','Fontweight','bold','FontSiz
e',point_load_fontsize,'rotation',angle_for_point_load(i)
+180);
end
end

% e) Texting uniform loading vector
if determinate_loading_vector==2 ||
determinate_loading_vector==3
if uniform_load<0;
angle_for_uniform_load(i)=angle(i)+270;
else                angle_for_uniform_load(i)=angle(i)+90;
end

% 9) PLOT M,N,T

% 9.1) ALL SYSTEM PLOT

fig2=figure; fig2.Position=[1 45 950 950];
subplot('Position',[0.05 0.05 0.90 0.90]); hold on; grid
on;
for i=1:size(connectivity,1)
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[c
oordinates_sorted_and_numerated(connectivity(i,1),3)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','b','linewidth',1);
```

```
plot([end_coordinate_normal{i,1}(1)
end_coordinate_normal{i,2}(1)],[end_coordinate_normal{i,1
}(2)
end_coordinate_normal{i,2}(2)],'color','r','linewidth',1.
5);
plot([end_coordinate_normal{i,1}(1)
coordinates_sorted_and_numerated(connectivity(i,1),2)],[e
nd_coordinate_normal{i,1}(2)
coordinates_sorted_and_numerated(connectivity(i,1),3)],'c
olor','r','linewidth',1.5);
plot([end_coordinate_normal{i,2}(1)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[e
nd_coordinate_normal{i,2}(2)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','r','linewidth',1.5);
text(end_coordinate_normal{i,1}(1),end_coordinate_normal{
i,1}(2),num2str(forces_of_elements_local{i}(1)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
text(end_coordinate_normal{i,2}(1),end_coordinate_normal{
i,2}(2),num2str(forces_of_elements_local{i}(4)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
title({'Normal Diyagram ( kN )  -  System',''})
end


fig3=figure; fig3.Position=[1 45 950 950];
subplot('Position',[0.05 0.05 0.90 0.90]); hold on; grid
on;
for i=1:size(connectivity,1)
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[c
oordinates_sorted_and_numerated(connectivity(i,1),3)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','b','linewidth',1);
plot([end_coordinate_shear{i,1}(1)
end_coordinate_shear{i,2}(1)],[end_coordinate_shear{i,1}(
2)
end_coordinate_shear{i,2}(2)],'color','r','linewidth',1.5
);
plot([end_coordinate_shear{i,1}(1)
coordinates_sorted_and_numerated(connectivity(i,1),2)],[e
nd_coordinate_shear{i,1}(2)
coordinates_sorted_and_numerated(connectivity(i,1),3)],'c
olor','r','linewidth',1.5);
plot([end_coordinate_shear{i,2}(1)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[e
nd_coordinate_shear{i,2}(2)
```

```
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','r','linewidth',1.5);
text(end_coordinate_shear{i,1}(1),end_coordinate_shear{i,
1}(2),num2str(forces_of_elements_local{i}(2)),'verticalal
ignment','bottom','horizontalalignment','center','fontsiz
e',fontsize_mnt);
text(end_coordinate_shear{i,2}(1),end_coordinate_shear{i,
2}(2),num2str(forces_of_elements_local{i}(5)),'verticalal
ignment','bottom','horizontalalignment','center','fontsiz
e',fontsize_mnt);
title({'Shear Diyagram ( kN )  -  System ',''})
end


fig4=figure; fig4.Position=[1 45 950 950];
subplot('Position',[0.05 0.05 0.90 0.90]); hold on; grid
on;
for i=1:size(connectivity,1)
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[c
oordinates_sorted_and_numerated(connectivity(i,1),3)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','b','linewidth',1);
plot([end_coordinate_moment{i,1}(1)
end_coordinate_moment{i,2}(1)],[end_coordinate_moment{i,1
}(2)
end_coordinate_moment{i,2}(2)],'color','r','linewidth',1.
5);
plot([end_coordinate_moment{i,1}(1)
coordinates_sorted_and_numerated(connectivity(i,1),2)],[e
nd_coordinate_moment{i,1}(2)
coordinates_sorted_and_numerated(connectivity(i,1),3)],'c
olor','r','linewidth',1.5);
plot([end_coordinate_moment{i,2}(1)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[e
nd_coordinate_moment{i,2}(2)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','r','linewidth',1.5);
text(end_coordinate_moment{i,1}(1),end_coordinate_moment{
i,1}(2),num2str(forces_of_elements_local{i}(3)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
text(end_coordinate_moment{i,2}(1),end_coordinate_moment{
i,2}(2),num2str(forces_of_elements_local{i}(6)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
title({'Moment Diyagram ( kN*m )  -  System ',''})
end
```

```
% 9.2) ELEMENT PLOT

for i=mnt_of_element

fig5=figure; fig5.Position=[1 500 600 495]; hold on; grid
on;
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[c
oordinates_sorted_and_numerated(connectivity(i,1),3)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','b','linewidth',1);
plot([end_coordinate_normal{i,1}(1)
end_coordinate_normal{i,2}(1)],[end_coordinate_normal{i,1
}(2)
end_coordinate_normal{i,2}(2)],'color','r','linewidth',1.
5);
plot([end_coordinate_normal{i,1}(1)
coordinates_sorted_and_numerated(connectivity(i,1),2)],[e
nd_coordinate_normal{i,1}(2)
coordinates_sorted_and_numerated(connectivity(i,1),3)],'c
olor','r','linewidth',1.5);
plot([end_coordinate_normal{i,2}(1)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[e
nd_coordinate_normal{i,2}(2)
coordinates_sorted_and_numerated(connectivity(i,2),3)],'c
olor','r','linewidth',1.5);
text(end_coordinate_normal{i,1}(1),end_coordinate_normal{
i,1}(2),num2str(forces_of_elements_local{i}(1)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
text(end_coordinate_normal{i,2}(1),end_coordinate_normal{
i,2}(2),num2str(forces_of_elements_local{i}(4)),'vertical
alignment','bottom','horizontalalignment','center','fonts
ize',fontsize_mnt);
title({['Normal Diyagram ( kN )  -  Element: ' name{i}
'(' num2str(i) ')'],''})
```

EK B : Doğrusal Deprem Analiz Programı

```
clear all; close all; clc; format long;

% 1) INPUT
tic
% 1.1) NODES
number_of_bay=3;                    % number of bay
span=6;                             % span (m)
number_of_floor=5;                  % number of floor
height=3;                           % height of floor(m)
start_coordinate=[0 0];             % start point
coordinate

% 1.2) ELEMENTS

modulus_of_elasticity=3e7;    % Modulus of elasticity
(kN/m^2)
% Column
depth_column=0.6;                       % Dimensions (m)
width_column=0.6;                       % Dimensions (m)

% Beam
depth_beam=0.6;                       % Dimensions (m)
width_beam=0.4;                       % Dimensions (m)


% 1.3) DAMPING
damping_ratio_first_mode=0.05;     % Rayleigh damping
ratio for first mode
damping_ratio_last_mode=0.05;      % Rayleigh damping
ratio for last mode

% 1.4) EARTHQUAKE RECORD
earthquake_record='Darfield6953X';

% 1.5) LAYOUT SIGNAL
determinate_number_nodes=1;        % Determination of
numbering of nodes   (please write '0' for switch on   or
'1' for switch off)
determinate_dir_element=1;         % Determination of
direction of element (please write '0' for switch on   or
'1' for switch off)
determinate_name_element=1;        % Determination of name
of element      (please write '0' for switch on   or '1'
for switch off)



% 2) NODES
```

```
% 2.1) LOCATION
for i=1:(number_of_floor+1)
coordinates_of_nodes_automatic_cell{i,1}=[linspace(start_
coordinate(1) start_coordinate(2)+(i-
1)*height,number_of_bay+1)'];
end
coordinates_of_nodes=cell2mat(coordinates_of_nodes_automa
tic_cell);
coordinates_sorted_and_numerated=SortingAndNumbering_func
tion(coordinates_of_nodes);    % Sorting and numbering
with function


% 2.2) CONNECTIVITY
if number_of_floor==0
connectivity_automatic=[[1:number_of_bay]'
[2:number_of_bay+1]'];
else
for i=1:number_of_floor
connectivity_automatic_cell{i,1}=[[(i-
1)*(number_of_bay+1)+1:
i*(number_of_bay+1)+number_of_bay]'
[i*(number_of_bay+1)+1:(i+1)*(number_of_bay+1)
i*(number_of_bay+1)+2:(i+1)*(number_of_bay+1)]'];
end
connectivity=cell2mat(connectivity_automatic_cell);
end


% 2.3) RESTRAINED DEGREE OF FREEDOM
% a) Automatic
for i=1:(number_of_bay+1)
restrained_DoF{i,1}=[i 1 2 3];
end

% b) Shapes and numbers of DoF restrained
for i=1:size(restrained_DoF,1)
restrained_nodes(i,1)=restrained_DoF{i}(1);
restrained_DoF_shapes{restrained_nodes(i),1}='s';
restrained_DoF_number{restrained_nodes(i),1}=[restrained_
nodes(i)*3-2; restrained_nodes(i)*3-1;
restrained_nodes(i)*3];
end

fully_freedom_nodes=coordinates_sorted_and_numerated(:,1)
;
fully_freedom_nodes(restrained_nodes)=[];

for i=1:size(fully_freedom_nodes,1)
restrained_DoF_shapes{fully_freedom_nodes(i)}='o';
```

```
end


% 2.4) MATCHING VECTOR FOR OBTAINING GLOBAL SYSTEM MATRIX
% a) All DoF
all_DoF_number=(1:3*coordinates_sorted_and_numerated(end,
1))';

% b) Restrained DoF
restrained_DoF_number_for_matching=cell2mat(restrained_Do
F_number);

% c) Unrestrained DoF
unrestrained_DoF_number_for_matching=all_DoF_number;
unrestrained_DoF_number_for_matching(restrained_DoF_numbe
r_for_matching)=[];

% d) Matching vector for reorganize
matching_vector_first=[unrestrained_DoF_number_for_matchi
ng;restrained_DoF_number_for_matching];

% e) Matching vector for orijinal position
for i=1:length(matching_vector_first)

matching_vector_end(i,1)=find(matching_vector_first==i);
end


% 3) ELEMENTS

for i
cross_section_area(i)=width_column*depth_column;
% (Element Properties -->  Cross section area (m^2))
moment_of_inertia(i)=(width_column*depth_column^3)/12;
% (Element Properties -->  Moment of inertia  (m^4))
end

for i=
cross_section_area(i)=width_beam*depth_beam;          %
(Element Properties -->  Cross section area (m^2))
moment_of_inertia(i)=(width_beam*depth_beam^3)/12;    %
(Element Properties -->  Moment of inertia  (m^4))
end


for i=1:size(connectivity,1)
% a) Angle with respect to global coordinate system
(Degree)
if
coordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)<0;
```

```
angle(i,1)=(atan((coordinates_sorted_and_numerated(connec
tivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))/(c
oordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)))/(
2*pi))*360+180;
else
angle(i,1)=(atan((coordinates_sorted_and_numerated(connec
tivity(i,2),3)-
coordinates_sorted_and_numerated(connectivity(i,1),3))/(c
oordinates_sorted_and_numerated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2)))/(
2*pi))*360;
end

% b) Length  (m)
length_of_element(i,1)=sqrt((coordinates_sorted_and_numer
ated(connectivity(i,2),2)-
coordinates_sorted_and_numerated(connectivity(i,1),2))^2+
(coordinates_sorted_and_numerated(connectivity(i,1),3)-
coordinates_sorted_and_numerated(connectivity(i,1),1))^2)
;

% c) Local Stiffness matrix
stiffness_of_elements_local{i,1}=[
(modulus_of_elasticity*cross_section_area(i))/length_of_e
lement(i)                                          0
0                                      -
(modulus_of_elasticity*cross_section_area(i))/length_of_e
lement(i)                                          0
0                                  ;

0
(12*modulus_of_elasticity*moment_of_inertia(i))/length_of
_element(i)^3
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2                                        0
-
(12*modulus_of_elasticity*moment_of_inertia(i))/length_of
_element(i)^3
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2 ;

0
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2
(4*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)                                          0
-
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2
```

```
(2*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)    ;
                                    -
(modulus_of_elasticity*cross_section_area(i))/length_of_e
lement(i)                                              0
0
(modulus_of_elasticity*cross_section_area(i))/length_of_e
lement(i)                                              0
0                                  ;

0                                   -
(12*modulus_of_elasticity*moment_of_inertia(i))/length_of
_element(i)^3    -
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2                                            0
(12*modulus_of_elasticity*moment_of_inertia(i))/length_of
_element(i)^3    -
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2 ;

0
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2
(2*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)                                              0
-
(6*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)^2
(4*modulus_of_elasticity*moment_of_inertia(i))/length_of_
element(i)    ];


% d) Transformation  matrix
transformation_matrix{i,1}=[ cos(angle(i)*((2*pi)/360))
sin(angle(i)*((2*pi)/360))   0                 0
0            0;
                            -sin(angle(i)*((2*pi)/360))
cos(angle(i)*((2*pi)/360))   0                 0
0            0;
                                              0
0              1                 0
0            0;
                                              0
0              0   cos(angle(i)*((2*pi)/360))
sin(angle(i)*((2*pi)/360)) 0;
                                              0
0              0  -sin(angle(i)*((2*pi)/360))
cos(angle(i)*((2*pi)/360)) 0;
                                              0
0              0                 0
0            1  ];
```

```
% e) Global Stiffness Matrix
stiffness_of_elements_global{i,1}=
transformation_matrix{i}'*stiffness_of_elements_local{i}*
transformation_matrix{i};

end



% 4) SYSTEM

% 4.1) MASS MATRIX
mass_of_frame_system{1}=zeros(3*coordinates_sorted_and_nu
merated(end,1));

for i=
mass_of_frame_system{1}((i-1)*3+1,(i-1)*3+1)=13.77;
mass_of_frame_system{1}((i-1)*3+2,(i-1)*3+2)=13.77;
mass_of_frame_system{1}((i-1)*3+3,(i-1)*3+3)=0.001;
end

mass_of_frame_system_reorganize{1}=mass_of_frame_system{1
}(matching_vector_first,matching_vector_first);
% Partitioning mass matrix according to unrestrained DoF
M_BB=mass_of_frame_system_reorganize{1}([length(unrestrai
ned_DoF_number_for_matching)+1:end],[length(unrestrained_
DoF_number_for_matching)+1:end]);


% 4.2) STIFFNESS MATRIX
stiffness_of_frame_system{1}=zeros(3*coordinates_sorted_a
nd_numerated(end,1));
for i=1:size(connectivity,1)
stiffness_of_frame_system{1}([connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3],[connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])=stiffness_of_frame_system{1}([conne
ctivity(i,1)*3-2 connectivity(i,1)*3-1
connectivity(i,1)*3 connectivity(i,2)*3-2
connectivity(i,2)*3-1
connectivity(i,2)*3],[connectivity(i,1)*3-2
connectivity(i,1)*3-1 connectivity(i,1)*3
connectivity(i,2)*3-2 connectivity(i,2)*3-1
connectivity(i,2)*3])+stiffness_of_elements_global{i};
end
stiffness_of_frame_system_reorganize{1}=stiffness_of_fram
e_system{1}(matching_vector_first,matching_vector_first);
```

```
% Partitioning stiffness matrix according to unrestrained
DoF
K_AA=stiffness_of_frame_system_reorganize{1}([1:length(un
restrained_DoF_number_for_matching)],[1:length(unrestrain
ed_DoF_number_for_matching)]);
K_BB=stiffness_of_frame_system_reorganize{1}([length(unre
strained_DoF_number_for_matching)+1:end],[length(unrestra
ined_DoF_number_for_matching)+1:end]);


% 4.3) DAMPING MATRIX
% Natural Frequencies and Mode Shapes by means of Eigen
Solution
[mode_shape_vector,square_of_frequencies]=eig(K_AA,M_AA);

% Natural Frequencies
natural_frequencies_radial=diag(sqrt(square_of_frequencie
s));
natural_frequencies_hertz=natural_frequencies_radial/(2*p
i);
period=1./natural_frequencies_hertz;

% Rayleigh (Proportional Damping)
frequency_first_mode=natural_frequencies_radial(1);
frequency_last_mode=natural_frequencies_radial(end);

beta_rayleigh=(2*(damping_ratio_first_mode*frequency_firs
t_mode-
damping_ratio_last_mode*frequency_last_mode))/(frequency_
first_mode^2-frequency_last_mode^2);



damping_of_frame_system{1}=alfa_rayleigh*mass_of_frame_sy
stem{1}+beta_rayleigh*stiffness_of_frame_system{1};
damping_of_frame_system_reorganize{1}=damping_of_frame_sy
stem{1}(matching_vector_first,matching_vector_first);

% Partitioning damping matrix according to unrestrained
DoF
C_AA=damping_of_frame_system_reorganize{1}([1:length(unre
strained_DoF_number_for_matching)],[1:length(unrestrained
_DoF_number_for_matching)]);
C_AB=damping_of_frame_system_reorganize{1}([1:length(unre
strained_DoF_number_for_matching)],[length(unrestrained_D
oF_number_for_matching)+1:end]);
C_BA=damping_of_frame_system_reorganize{1}([length(unrest
rained_DoF_number_for_matching)+1:end],[1:length(unrestra
ined_DoF_number_for_matching)]);
```

```
C_BB=damping_of_frame_system_reorganize{1}([length(unrest
rained_DoF_number_for_matching)+1:end],[length(unrestrain
ed_DoF_number_for_matching)+1:end]);


% 4.4) EARTHQUAKE LOAD MATRIX
load(earthquake_record);
t=t_interpolation;
delta_t=t(2)-t(1);



% Earthquake Loading
earthquake_loading_of_frame_system{1}=mass_of_frame_syste
m{1}*ones(length(all_DoF_number),1);
earthquake_loading_of_frame_system_reorganize{1}=earthqua
ke_loading_of_frame_system{1}(matching_vector_first,[1:en
d]);
% Partitioning earthquake loading matrix according to
unrestrained DoF
P_A=earthquake_loading_of_frame_system_reorganize{1}([1:l
ength(unrestrained_DoF_number_for_matching)],[1:end]);
P_B=earthquake_loading_of_frame_system_reorganize{1}([len
gth(unrestrained_DoF_number_for_matching)+1:end],[1:end])
;



% 5) LINEAR DYNAMIC ANALYSIS FOR FRAME SYSTEM (with
Newmark beta)

% Initial Conditions
u_cell{1}=zeros(length(unrestrained_DoF_number_for_matchi
ng),1);
v_cell{1}=zeros(length(unrestrained_DoF_number_for_matchi
ng),1);
a_cell{1}=inv(M_AA)*(P_A(:,1)-C_AA*v_cell{1}-
K_AA*u_cell{1});

% Newmark-beta constant
gama=1/2;
beta=1/4;

kd=(1/(beta*delta_t^2))*M_AA+(gama/(beta*delta_t))*C_AA+K
_AA;

for i=1:(length(t)-1)
Pd{i+1}=_t*(gama/(2*beta)-1)*C_AA)*a_cell{i};
u_cell{i+1}=inv(kd)*Pd{i+1};
```

```
v_cell{i+1}=(gama/(beta*delta_t))*(u_cell{i+1}-
u_cell{i})+(1-gama/beta)*v_cell{i}+(delta_t*(1-
gama/(2*beta)))*a_cell{i};
a_cell{i+1}=(1/(beta*delta_t^2))*(u_cell{i+1}-u_cell{i})-
(1/(beta*delta_t))*v_cell{i}+(1-1/(2*beta))*a_cell{i};
end

u=cell2mat(u_cell);
v=cell2mat(v_cell);
a=cell2mat(a_cell);



% 6) LAYOUT
% 6.1) NAMING OF ELEMENT
if size(connectivity,1)==0 name={};
else
% One digit
if size(connectivity,1)>0 && size(connectivity,1)<=26
for k=1:26;  name{k,1}=char(64+k); end;

% Two digit
elseif size(connectivity,1)>26 &&
size(connectivity,1)<=26+26^2
for k=1:26;  name{k,1}=char(64+k); end;
for k=1:26;   for l=1:26;  name{26+(k-
1)*26+l,1}=char([64+k 64+l]); end; end;

% Three digit
elseif size(connectivity,1)>26+26^2 &&
size(connectivity,1)<=26+26^2+26^3
for k=1:26;  name{k,1}=char(64+k); end;
for k=1:26;   for l=1:26;  name{26+(k-
1)*26+l,1}=char([64+k 64+l]); end; end;
for k=1:26;   for l=1:26;   for m=1:26;  name{26+26^2+(k-
1)*26^2+(l-1)*26+m,1}=char([64+k 64+l 64+m]); end; end;
end;

% Four digit
elseif size(connectivity,1)>26+26^2+26^3 &&
size(connectivity,1)<=26+26^2+26^3+26^4
for k=1:26;  name{k,1}=char(64+k); end;
for k=1:26;   for l=1:26;  name{26+(k-
1)*26+l,1}=char([64+k 64+l]); end; end;
for k=1:26;   for l=1:26;   for m=1:26;  name{26+26^2+(k-
1)*26^2+(l-1)*26+m,1}=char([64+k 64+l 64+m]); end; end;
end;
for k=1:26;   for l=1:26;   for m=1:26;   for n=1:26;
name{26+26^2+26^3+(k-1)*26^3+(l-1)*26^2+(m-
1)*26+n,1}=char([64+k 64+l 64+m 64+n]); end; end; end;
end;
```

```
end
end


% 6.2) CENTER ALINGMENT
if number_of_floor*height >= number_of_bay*span;
xlim_first=start_coordinate(1)+(number_of_bay*span-
number_of_floor*height)/2; xlim_end=
start_coordinate(1)+(number_of_bay*span+number_of_floor*h
eight)/2; ylim_first=start_coordinate(2);
ylim_end=start_coordinate(2)+number_of_floor*height;
else
xlim_first=start_coordinate(1);
xlim_end=  start_coordinate(1)+number_of_bay*span;
ylim_first=start_coordinate(2)+(number_of_floor*height-
number_of_bay*span)/2;
ylim_end=start_coordinate(2)+(number_of_floor*height+numb
er_of_bay*span)/2;
end



% 7) PLOT FRAME SYSTEM

fig1=figure; fig1.Position=[970 45 950 950];
subplot('Position',[0.05 0.05 0.90 0.90]); hold on; grid
on;

% 7.1) NODES
for i=1:size(coordinates_sorted_and_numerated,1)
% a) Plot each node and determinate restrained DoF shapes
plot(coordinates_sorted_and_numerated(i,2),coordinates_so
rted_and_numerated(i,3), restrained_DoF_shapes{i}
,'markersize',10,'markeredgecolor',[0 0.6
0],'markerfacecolor',[0 0.6 0]);

% b) Texting number of nodes
title({'FRAME SYSTEM',''}); xlabel('Span (m)');
ylabel('Height (m)'); if  (number_of_bay==0 &&
number_of_floor==0); xlim auto; ylim auto; else
xlim([xlim_first xlim_end]); ylim([ylim_first ylim_end]);
end;

% 7.2) ELEMENTS
if size(coordinates_of_nodes,1)>=2
for i=1:size(connectivity,1)
% a) Plot each element
plot([coordinates_sorted_and_numerated(connectivity(i,1),
2)
coordinates_sorted_and_numerated(connectivity(i,2),2)],[c
oordinates_sorted_and_numerated(connectivity(i,1),3)
```

```
coordinates_sorted_and_numerated(connectivity(i,2),3)],'-
','color','b','linewidth',1);

% b) Texting direction of element
if determinate_dir_element==1;
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+coordinates_sorted_and_numerated(connectivity(i,2),2))
/2,(coordinates_sorted_and_numerated(connectivity(i,1),3)
12,'rotation',angle(i)); end;

% c) Texting name of element
if determinate_name_element==1;
text((coordinates_sorted_and_numerated(connectivity(i,1),
2)+coordinates_sorted_and_numerated(connectivity(i,2),2))
/2,(coordinates_sorted_and_numerated(connectivity(i,1),3)
+coordinates_sorted_and_numerated(connectivity(i,2),3))/2
,[name{i} '(' num2str(i)
')'],'horizontalalignment','center','verticalalignment','
bottom','Color','m','Fontweight','bold','FontSize',10,'ro
tation',angle(i)); end;

end
end


% 8) PLOT FRAME SYSTEM RESPONSE
% 8.1) Relative displacement with respect to ground
fig2=figure; fig2.Position=[1 45 950 950];
subplot('Position',[0.07 0.700 0.90 0.25]); hold on;  for
i=1:3:length(unrestrained_DoF_number_for_matching);
plot(t,u(i,:)); lgnd{(i-
1)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i)+2)/3)
'.node)']; end; title('HORIZONTAL DISPLACEMENT');
xlabel('Time (sec)'); ylabel('Translation (m)');
legend(lgnd); grid on;
subplot('Position',[0.07 0.375 0.90 0.25]); hold on;  for
i=2:3:length(unrestrained_DoF_number_for_matching);
plot(t,u(i,:)); lgnd{(i-
2)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i)+1)/3)
'.node)']; end; title('VERTICAL DISPLACEMENT');
xlabel('Time (sec)'); ylabel('Translation (m)');
legend(lgnd); grid on;
subplot('Position',[0.07 0.050 0.90 0.25]); hold on;  for
i=3:3:length(unrestrained_DoF_number_for_matching);
plot(t,u(i,:)); lgnd{(i-
3)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
```

```
num2str((unrestrained_DoF_number_for_matching(i))/3)
'.node)']; end;   title('ROTATIONAL DISPLACEMENT');
xlabel('Time (sec)'); ylabel('Rotation (rad)');
legend(lgnd); grid on;

% 8.2) Relative velocity with respect to ground
fig3=figure; fig3.Position=[1 45 950 950];
subplot('Position',[0.07 0.700 0.90 0.25]); hold on;  for
i=1:3:length(unrestrained_DoF_number_for_matching);
plot(t,v(i,:)); lgnd{(i-
1)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i)+2)/3)
'.node)']; end; title('HORIZONTAL VELOCITY');
xlabel('Time (sec)'); ylabel('Translational Velocity
(m/s)'); legend(lgnd); grid on;
subplot('Position',[0.07 0.375 0.90 0.25]); hold on;  for
i=2:3:length(unrestrained_DoF_number_for_matching);
plot(t,v(i,:)); lgnd{(i-
2)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i)+1)/3)
'.node)']; end; title('VERTICAL VELOCITY');
xlabel('Time (sec)'); ylabel('Translational Velocity
(m/s)'); legend(lgnd); grid on;
subplot('Position',[0.07 0.050 0.90 0.25]); hold on;  for
i=3:3:length(unrestrained_DoF_number_for_matching);
plot(t,v(i,:)); lgnd{(i-
3)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i))/3)
'.node)']; end;   title('ROTATIONAL VELOCITY');
xlabel('Time (sec)'); ylabel('Rotational Velocity
(rad/s)');  legend(lgnd); grid on;

% 8.3) Relative acceleration with respect to ground
fig4=figure; fig4.Position=[1 45 950 950];
subplot('Position',[0.07 0.700 0.90 0.25]); hold on;  for
i=1:3:length(unrestrained_DoF_number_for_matching);
plot(t,a(i,:)); lgnd{(i-
1)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
num2str((unrestrained_DoF_number_for_matching(i)+2)/3)
'.node)']; end; title('HORIZONTAL ACCELERATION');
xlabel('Time (sec)'); ylabel('Translational Acceleration
(m/s^2)'); legend(lgnd); grid on;
subplot('Position',[0.07 0.375 0.90 0.25]); hold on;  for
i=2:3:length(unrestrained_DoF_number_for_matching);
plot(t,a(i,:)); lgnd{(i-
2)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))
'. DoF ('
```

```
num2str((unrestrained_DoF_number_for_matching(i)+1)/3)
'.node)']; end; title('VERTICAL ACCELERATION');
xlabel('Time (sec)'); ylabel('Translational Acceleration
(m/s^2)'); legend(lgnd); grid on;
subplot('Position',[0.07 0.050 0.90 0.25]); hold on;  for
i=3:3:length(unrestrained_DoF_number_for_matching);
plot(t,a(i,:)); lgnd{(i-
3)/3+1}=[num2str(unrestrained_DoF_number_for_matching(i))

% 9) PLOT GROUND MOTION
fig5=figure; fig5.Position=[1 600 950 400];
plot(t,a_interpolation);  title('GROUND ACCELERATION');
xlabel('Time (sec)'); ylabel('Acceleration (m/s^2)');
grid on;
```

ÖZGEÇMİŞ

Ad-Soyad **:** Fatih YILDIZ

Doğum Tarihi ve Yeri **:** 1990 – Edremit/BALIKESİR

E-posta **:** fthyldz.itu@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2014, İstanbul Teknik Üniversitesi, İnşaat Fakültesi, İnşaat Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR ve SUNUMLAR:

Erkuş, B., Kasapoğlu, B., & Yıldız, F. (2017) "Yapıların doğrusal olmayan deprem analizlerinde bünye fonksiyonlarının paralelleştirilmesi" 4.Uluslararası Deprem Mühendisliği ve Sismoloji Konferansı, Eskişehir.