
GAZİ ÜNİVERSİTESİ
Gazi
1926

PYTHON PROGRAMLAMA DİLİ İLE TERMOELASTİK GERİLME ANALİZİ

Melike KORKMAZ

YÜKSEK LİSANS TEZİ
İNŞAAT MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2020

Melike KORKMAZ tarafından hazırlanan "PYTHON PROGRAMLAMA DİLİ İLE TERMOELASTİK GERİLME ANALİZİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi İnşaat Mühendisliği Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Bahadır ALYAVUZ

İnşaat Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……..

Başkan: Doç. Dr. Çiğdem DİNÇKAL

İnşaat Mühendisliği Ana Bilim Dalı, Çankaya Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ...……

Üye: Dr. Öğr. Üyesi Önder KOÇYİĞİT

İnşaat Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ...……

Tez Savunma Tarihi: 17/01/2020

Jüri tarafından kabul edilen bu çalışmanın Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..…….

Prof. Dr. Sena YAŞYERLİ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Melike KORKMAZ

17/01/2020

PYTHON PROGRAMLAMA DİLİ İLE TERMOELASTİK GERİLME ANALİZİ

(Yüksek Lisans Tezi)

Melike KORKMAZ

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ocak 2020

ÖZET

Bu tez çalışmasında sıcaklık etkisi altındaki iki boyutlu dikdörtgen levhada Python programlama dilinde geliştirilen Isıl Gerilme Analizi uygulamasıyla ısıl gerilme analizi yapılmıştır. Öncelikle, levha içerisindeki herhangi bir noktada zamanla sıcaklıkta değişimin olmadığı kararlı hal ısı iletim durumuna göre sıcaklık dağılımı hesaplanmıştır. Dikdörtgen levhanın çözümünde kullanılan Navier ve gerilme denklemleri düzlem gerilme kabulüne göre ifade edilmiştir. Isı iletimi, Navier denklemi ve gerilme denklemleri merkezi sonlu farklar yöntemine göre ayrıklaştırılmıştır ve sonlu fark denklemleri elde edilmiştir. Sonlu fark denklemlerinin çözümü için iteratif yöntem tercih edilmiştir. Çözüm ağındaki tüm düğüm noktalarının sıcaklık, yer değiştirme ve gerilme değerleri elde edilmiş ve çözümler grafiksel olarak ifade edilmiştir. Isıl Gerilme Analizi uygulaması ile elde edilen sonuçlar üniform sıcaklık dağılımı için analitik çözümle ve diğer sıcaklık dağılımları için ABAQUS sonlu elemanlar yazılımı kullanılarak elde edilen çözümlerle karşılaştırılmıştır.

Bilim Kodu : 91120
Anahtar Kelimeler : Termoelastisite, Sayısal modelleme, Gerilme analizi, Python
Sayfa Adedi : 177
Danışman : Doç. Dr. Bahadır ALYAVUZ

THERMOELASTIC STRESS ANALYSIS WITH PYTHON PROGRAMMING LANGUAGE

(M. Sc. Thesis)

Melike KORKMAZ

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

January 2020

ABSTRACT

In this thesis, thermal stress analysis was performed by using Thermal Stress Analysis application developed in Python programming language on a two dimensional rectangular plate under the effect of temperature. Firstly, the temperature distribution was calculated under the steady state conditions where there exists no change in time. Navier and stress equations used in the solution of the rectangular plate were expressed according to the plane stress assumption. Heat conduction, Navier equation and stress equations were discretized using the central finite difference method and finite difference equations were obtained. Iterative method was preferred for the solution of finite difference equations. Temperature, displacement and stress values of all nodes in the solution mesh were obtained and the solutions were graphically expressed. The results obtained by Thermal Stress Analysis application were compared with the analytical solution for uniform temperature distribution and were compared with the results obtained using ABAQUS finite element software for other temperature distributions.

Science Code : 91120
Key Words : Thermoelasticity, Numerical modeling, Stress analysis, Python
Page Number : 177
Supervisor : Assoc. Prof. Dr. Bahadır ALYAVUZ

TEŞEKKÜR

Yüksek lisans eğitimim boyunca tüm bilgi birikimini esirgemeden benimle paylaşan ve tecrübelerinden yararlanırken göstermiş olduğu hoşgörülü tavrından dolayı değerli hocam Doç. Dr. Bahadır ALYAVUZ'a teşekkürlerimi sunarım.

Ayrıca eğitim öğrenim hayatım boyunca üzerimde emeği olan bütün hocalarıma, aileme ve özellikle sevgili eşim Rıdvan KORKMAZ'a teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	Açıklamalar
T_0	Başlangıç sıcaklığı
ε_x	x doğrultusundaki birim şekil değiştirme
ε_{xy}	xy düzlemindeki şekil değiştirme
ε_{xz}	xz düzlemindeki şekil değiştirme
ε_y	y doğrultusundaki birim şekil değiştirme
ε_{yz}	yz düzlemindeki şekil değiştirme
ε_z	z doğrultusundaki birim şekil değiştirme
σ_x	x doğrultusundaki normal gerilme
τ_{xy}	xy kesme gerilmesi
τ_{xz}	xz kesme gerilmesi
σ_y	y doğrultusundaki normal gerilme
τ_{yz}	yz kesme gerilmesi
σ_z	z doğrultusundaki normal gerilme
∇^2	Laplace operatörü
c	Özgül ısı
°C	Santigrat derece
E	Elastisite modülü
F	Dış kuvvet
h	Isı kaynağı
k	Isıl iletkenlik tansörü
L_x	Levha uzunluğu
L_y	Levha genişliği
mm	Milimetre
MPa	MegaPascal
T	Sıcaklık
u	Yer değiştirme vektörü

Simgeler	**Açıklamalar**
x, y	Eksen takımı
α	Lineer ısıl genleşme katsayısı
β	Isı modülü
δ	Kronecker deltası
ΔT	Sıcaklık farkı
ε	Birim şekil değiştirme tansörü
θ	Sıcaklık farkı
λ, μ	Lame sabitleri
ν	Poisson oranı
σ	Gerilme tansörü
ρ	Yoğunluk

Kısaltmalar	**Açıklamalar**
IGAU	Isıl Gerilme Analizi Uygulaması
LSD	Lineer Sıcaklık Dağılımı
ÜSD	Üniform Sıcaklık Dağılımı

1. GİRİŞ

Isı iletimi ve elastisite teorisinin birleşimi olarak düşünülen termoelastisite, ısıl yükleme etkisi altında kalan elastik cisimlerdeki yer değiştirme ve gerilmeleri inceler. Elastik katı bir cisimde ısı iletilmesiyle sıcaklık ve şekil değişimi meydana gelir. Sıcaklık değişimi nedeniyle gerilmelerin oluştuğu durumda, gerilme değerleri yapısal kusurlara sebep olabilecek seviyelere ulaşabilir. Bu yüzden yüksek sıcaklık değişimini içeren birçok problem için ısıl gerilme analizi oldukça önemlidir.

Mühendislik bilimlerinde karşılaşılan problemlerin modellenmesinde sıklıkla diferansiyel denklemler kullanılmıştır. Diferansiyel denklem çözümüne dayalı problemlerin geometrileri, yükleme ve sınır şartları karmaşıklaştıkça analitik çözümler yetersiz kalmıştır. Analitik çözümlerin yetersiz kalması sayısal yöntemlerin gelişmesine neden olmuştur. Teknolojinin gelişmesiyle bilgisayar ortamında kullanılan sayısal yöntemlerin çözümü kolaylaşmış ve daha çok tercih edilmeye başlanmıştır. Çözülmesi düşünülen problemler sayısal yöntemleri kullanan paket programlar veya yeni yazılımlar geliştirilerek çözülmüştür. Yeni yazılımların ihtiyaçlara daha iyi karşılık vermesi ve daha ekonomik çözümler sağlaması en büyük avantajlarıdır.

Termoelastisite ile ilgili ilk çalışma Duhamel tarafından yapılmıştır [1]. Duhamel çalışmasında sıcaklık değişimiyle meydana gelen şekil değiştirme denklemlerinin oluşturulması ve sınır değer problemleri üzerine çalışmalar yapmıştır.

Belirli bir döneme kadar termoelastisite üzerine geliştirilen teorilerde ısıl ve mekanik etkiler birbirinden bağımsız olarak ifade edilmiştir. Sıcaklık alanları gerilme alanı hesabından bağımsız olarak belirlenmiştir. Thomson termodinamik yasalarını kullanarak sıcaklık farkından dolayı oluşan şekil değiştirme ve gerilmeleri ilk kez birbiriyle ilişkilendirerek tanımlamıştır [2].

Ayrık, birleştirilmiş ve genelleştirilmiş olmak üzere üç tip termoelastisite yaklaşımı vardır. Ayrık ve birleştirilmiş termoelastisite klasik termoelastisite olarak adlandırılmıştır. Ayrık termoelastisitede ısı iletimi denklemi herhangi bir elastik terim içermezken birleştirilmiş termoelastisitede ısı iletim denklemi elastik terim içermektedir. Carter ve Booker çalışmalarında metal türü malzemelere ayrık termoelastisite yaklaşımı uygulandığında

sonuçların tatmin edici olacağını belirtmişlerdir [3]. Kompozit türü malzemelerde ise ayrık termoelastisite yerine birleştirilmiş termoelastisite yaklaşımının daha doğru sonuçlar ortaya koyduğunu iki ayrı problem üzerinde göstermişlerdir.

Klasik termoelastisitede ısı iletim denklemi ısı dalgalarının sonsuz hızda yayıldığını öngören parabolik tiptedir. Isı dalgalarının sonlu hızda yayıldığını düşüncesinin ortaya çıkmasıyla genelleştirilmiş termoelastisite teorisi oluşmuştur. Balla, tezinde termoelastisitenin temel denklemlerini klasik ve genelleştirilmiş termoelastisite teorileri için ayrı ayrı ifade etmiştir [4]. Isı iletim denkleminde yeni bir malzeme özelliği olan gevşeme süresinin tanımlanmasıyla ısı denklemi ısı dalgalarının sonlu hızda yayıldığını öngören hiperbolik özellik göstermiştir.

Bu çalışmada levhanın içerisinde zamanla sıcaklıkta herhangi bir değişimin olmadığı durum incelendiği için sıcaklık alanı belirlenirken kullanılan ısı iletim denklemi Laplace denklemidir. Laplace denklemi problemin boyutuna göre x, y ve z konum değişkenlerine bağlıdır. Laplace denklemi herhangi bir elastik terim içermediği için tez çalışmasında ayrık termoelastisite teorisi kullanılmıştır.

Sıcaklık değerleri hesaplanırken kullanılan ısı iletimi ve yer değiştirme değerleri hesaplanırken kullanılan Navier denklemleri kısmi diferansiyel denklemler ile ifade edilmiştir. Sonlu farklar yönteminde türev ifadeleri yerine Taylor serisi açılımı kullanılarak elde edilen merkezi fark ifadeleri koyulmuştur ve sonlu fark denklemi elde edilmiştir. Sonlu fark denkleminin çözümü için başlangıçta bilinen sıcaklık ve yer değiştirme değerlerini kullanılarak bir sonraki adımdaki değerlerin hesaplanmasına dayalı olan iteratif yöntem kullanılmıştır. Sonlu farklar yönteminde hassas çözümler elde etmek için sıklaştırılan çözüm ağı büyük boyutlu matrisler üzerinde işlem yapılmasını gerektirmiştir. Matrisin boyutunun çok büyük olması çok bilinmeyenli lineer denklem sistemlerinin çözümünü gerektirir. Bu tür problemin çözümünde büyük çözüm ağları için terimlerin çoğu sıfır olur. Bu gibi seyrek sistemlere matris eliminasyon yöntemleri uygulandığında, bu sıfırları depolamak için büyük miktarda bilgisayar belleği boşa harcanır. Bu duruma alternatif çözüm eliptik denklemlerin çözümünde iteratif yöntemler uygulanabilir yaklaşım sağlarlar [5].

Patil ve Prasad makalelerinde kararlı hal ısı iletimi durumu için dikdörtgen levhalarda termoelastik problemleri incelemişlerdir [6]. İnce dikdörtgen bir levhanın sıcaklık, yer değiştirme ve gerilme fonksiyonlarını belirlemişlerdir.

Verma ve Kulkarni yaptıkları çalışmada iki boyutlu ince dikdörtgen bir levhadaki sıcaklık değişiminden dolayı meydana gelen ısıl gerilmeleri hesaplamak için sonlu elemanlar yöntemini kullanmışlardır [7]. Sıcaklık ve ısıl gerilmelerin nümerik değerlerini belirlemek için MATLAB ortamında yazılım geliştirmişlerdir. Ayrıklaştırma işlemi için çok sayıda eleman alındığında nümerik değerlerin daha doğru sonuçlar verdiğini belirtmişlerdir.

Pandit ve Kulkarni çalışmalarında ısıl iletkenliğin sıcaklığa bağlı olarak değiştiği ve değişmediği varsayımları altında ısıl gerilme analizi yapmışlardır [8]. Bakır dikdörtgen bir levhada ısı transferi analizi yapılırken Crank Nicolson metodunu kullanmışlardır. Isıl gerilme analizi yapılırken sınır şartlarını Airy gerilme fonksiyonuna bağlı ifade etmişlerdir ve sonlu farklar metodunu kullanmışlardır. Isı transferi ve ısıl gerilme analizinde sıcaklığa bağlı olarak ısıl iletkenliğin belirlenmesinin önemini ortaya koymuşlardır.

Silveira Junior ve diğerleri kararlı hal ısı iletim durumunda bir boyutlu termoelastik bir problem için sonlu farklar metodunu kullanarak ısıl gerilme analizi yapmışlardır [9]. Sonlu farklar yöntemiyle elde edilen sonuçları analitik ve deneysel sonuçlarla karşılaştırarak doğrulamışlardır. Termoelastik problemlerle ilgili değişkenlerin çözümlerini belirlemede sonlu farklar metodunun güvenilir ve iyi bir alternatif olduğunu ifade etmişlerdir.

Gandhe, Ghugal ve Kulkarni sabit ısı kaynağı etkisinde kalan kararlı hal ısı iletim durumundaki ince dikdörtgen bir levhada sıcaklık değişimi ve ısıl gerilmelerin belirlenmesi problemini ele almışlardır [10]. Isı iletim denkleminin çözümü integral dönüşüm tekniğini kullanarak elde etmişlerdir. Isıl gerilmelerin belirlenmesinde Airy gerilme fonksiyonunu kullanmışlardır.

Arnab ve diğerleri ısıl yüklere maruz kalan eş merkezli boşluğa sahip, iki bileşenli kompozit ince dairesel bir diskte termoelastik analiz yapmışlardır [11]. İki boyutlu problemi ikinci dereceden bir diferansiyel denklem çözümüne indirgemişlerdir. Sonlu fark ayrıklaştırma işlemi yardımıyla elde ettikleri cebirsel lineer denklem sistemini Gauss yok

etme yöntemini kullanarak çözmüşlerdir. İnce dairesel bir diskte yer değiştirme ve gerilme sonuçlarını elde etmişlerdir.

Hernandez ve diğerleri silikon alt tabaka üzerindeki kalay kaplamalarındaki ısıl gerilmeleri hem analitik hem de sonlu elemanlar modellemesi yaparak analiz etmişlerdir [12]. Kalay kaplamadaki ısıl gerilmelerin sıcaklık değişiminden kaynaklandığını göstermişlerdir.

Ersan tezinde fonksiyonel derecelendirilmiş disklerin ısıl yük etkisi altındaki davranışlarını analitik ve sayısal olarak incelemiştir [13]. Sabit sıcaklığa maruz kalan izotropik fonksiyonel derecelendirilmiş disk için radyal ve teğetsel gerilmelerin analitik sonuçlarını MATLAB ortamında elde etmiştir. Sonlu elemanlar metodunu kullanan ANSYS yazılımı yardımıyla modelin kurulması, yükleme ve sınır şartlarının tanımlanması ardından yer değiştirme ve gerilme değerlerini elde etmiştir. Aynı geometri ve sınır şartlarına sahip disk için analitik ve nümerik sonuçların tutarlılığı sonuç kısmında grafiksel olarak gösterilmiştir.

Akarsu tezinde farklı üniform sıcaklık etkisindeki kompozit bir plağın ısıl gerilme analizini sonlu elemanlar yöntemini kullanan ANSYS yazılımı ile hesaplamıştır [14]. ANSYS çözüm aşamalarını akış şeması üzerinde göstermiştir. Uygulanan sıcaklık arttıkça ısıl gerilmelerin arttığını belirtmiştir.

Burada sunulan tezin amacı ısıl etki altındaki iki boyutlu dikdörtgen levhada ısıl gerilme analizi yapmaktır. Bu doğrultuda sonlu farklar metodu kulanılarak Python ortamında Isıl Gerilme Analizi adında bir uygulama geliştirilmiştir. Levhadaki sıcaklık, yer değiştirme ve ısıl gerilme dağılımları geliştirilen uygulama kullanılarak elde edilmiştir. Tezi oluşturan bölümlerde ısıl gerilme analizini gerçekleştirmek için gerekli teorik bilgiler ve bu bilgilerin kullanılmasıyla Python ortamında yazılımın nasıl geliştirildiği ifade edilmiştir. Tezin bölümleri aşağıdaki gibi özetlenebilir.

Bölüm 2’de lineer-elastik, homojen ve izotropik malzeme kabulü altında termoelastisitenin temel denklemleri verilmiştir. İki boyutlu problemler için düzlem şekil değiştirme ve düzlem gerilme kabullerinin hangi şartlarda kullanılacağı, yer değiştirme ve gerilme denklemlerindeki farklılıklar anlatılmıştır.

Bölüm 3’te sayısal yöntemlere neden ihtiyaç duyulduğu, sonlu farklar yönteminin çözüm aşamaları, sonlu farklar yaklaşım türleri, birinci ve ikinci dereceden fark ifadelerinin Taylor serilerine göre elde edilişleri anlatılmıştır. Elde edilen sonlu fark ifadeleri düzlem gerilme durumu için ısı iletim, yer değiştirme ve gerilme denklemlerine uygulanarak çözülmesi gerekli sayısal denklemler elde edilmiştir.

Bölüm 4’te ABAQUS sonlu elemanlar yazılımı kullanılarak ısıl gerilme analizinin nasıl yapıldığı Python-1 örneği üzerinden anlatılmıştır.

Bölüm 5’te Python programlama dili, gelişimi, kullanım alanları ve geliştirme ortamından kısaca bahsedilmiştir. Termoelastik gerilme analizi için geliştirilen yazılımda kullanılan veri tipleri, karşılaştırma operatörleri, mantıksal bağlaçlar, kontrol deyimleri, döngüler, fonksiyonlar, sınıflar ve modüllere neden ihtiyaç duyulduğu, özellikleri, kullanımı örneklerle açıklanarak anlatılmıştır. Uygulamayı oluşturan sıcaklık analizi, yer değiştirme analizi, gerilme analizi modüllerinin nasıl geliştirildiği ve uygulamanın nasıl kullanılacağı anlatılmıştır.

Bölüm 6’da seçilen örneklerin sınır şartları, geometrik özellikleri ve malzeme özellikleri tanımlanmıştır. Üniform ve lineer sıcaklık dağılımı altında sıcaklık, yer değiştirme ve gerilme değerleri hesaplanmıştır. Çözüm sonuçları grafikler ile ifade edilmiştir. Yazılımdan elde edilen sonuçlar, analitik çözüm ve ABAQUS sonlu elemanlar yazılımı kullanılarak elde edilen çözümlerle karşılaştırılmıştır.

Bölüm 7’de tez kapsamında yapılan çalışma sonuçları değerlendirilmiştir.

2. TERMOELASTİSİTE İÇİN TEMEL DENKLEMLER

Sürekli ortamlar mekaniği, katı, sıvı ve gazlardaki gerilmeler ve bu malzemelerin şekil değiştirmeleri veya akışı ile ilgilenen mekaniğin bir dalıdır. Sürekli ifadesi, analizin temelini oluşturan basitleştirici kavramı ifade eder. Bu yaklaşım maddenin moleküler yapısını ihmal eder ve onun molekülleri arasında boşluk ve boş alanlar olmadığını varsayar [15]. Lineer-elastik, homojen ve izotropik malzeme kabulü altında üç boyutlu termoelastisitenin yönetici denklemleri sürekli ortamlar mekaniğinin kabulleri altında ifade edilir.

Sürekli ortamlar mekaniğinde yapılan işlemlerde matris notasyonu veya tansör notasyonu kullanılır. Tansörel işlemlerin yapılmasını ve gösterimini kolaylaştırmak için bu işlemlerin indisler yardımıyla yapıldığı bir notasyon kullanılabilir. İndis notasyonu gösteriminde bir vektörün bileşenlerini ifade eden indisin sayı aralığı 1'den 3'e kadardır. Herhangi bir $\underset{\sim}{u}$ vektörünün indis notasyonu ile gösterimi aşağıdaki gibidir.

$$\underset{\sim}{u} = u_1\underset{\sim}{e}_1 + u_2\underset{\sim}{e}_2 + u_3\underset{\sim}{e}_3 \tag{2.1}$$

$$\underset{\sim}{u} = \sum_{i=1}^{3} u_i\underset{\sim}{e}_i = u_i\underset{\sim}{e}_i \tag{2.2}$$

Eş. 2.2'de eşitliğin en sağında toplama anlaşımı kullanılmıştır. Bu anlaşıma göre bir terimde indis iki kez görülürse indis aralığında bir toplama işlemi anlaşılır. $u_i e_i$ bir terim olarak değerlendirilir ve iki kez görülen i indisi üzerinde 1'den 3'e kadar $u_1e_1 + u_2e_2 + u_3e_3$ toplamı yapılır.

Sürekli ortamlar mekaniği içinde yerdeğiştirme ve gerilme denklemlerin oluşturulmasında çeşitli matematiksel operatörlere ihtiyaç duyulmaktadır. Bunlardan bir skalerin konuma göre toplam değişiminin büyüklüğünü vektörel olarak ifade eden gradyan operatörü,

$$\underset{\sim}{\nabla} = \frac{\partial}{\partial x_i}\underset{\sim}{e}_i = \frac{\partial}{\partial x_1}\underset{\sim}{e}_1 + \frac{\partial}{\partial x_2}\underset{\sim}{e}_2 + \frac{\partial}{\partial x_3}\underset{\sim}{e}_3 \tag{2.3}$$

şeklinde indis notasyonu ile gösterilebilir. ϕ skaler fonksiyonunun gradyanının indis notasyonu ile gösterimi,

$$Grad\phi = \underset{\sim}{\nabla}\phi = \left(\frac{\partial}{\partial x_i}\underset{\sim}{e}_i\right)\phi = \frac{\partial\phi}{\partial x_1}\underset{\sim}{e}_1 + \frac{\partial\phi}{\partial x_2}\underset{\sim}{e}_2 + \frac{\partial\phi}{\partial x_3}\underset{\sim}{e}_3 \tag{2.4}$$

şeklindedir. $\underset{\sim}{u}$ vektörünün diverjansının indis notasyonu ile gösterimi ise

$$Div\underset{\sim}{u} = \underset{\sim}{\nabla}\underset{\sim}{u} = \left(\frac{\partial}{\partial x_i}\underset{\sim}{e}_i\right)\left(u_j\underset{\sim}{e}_j\right) = \frac{\partial u_j}{\partial x_i}(\underset{\sim}{e}_i\underset{\sim}{e}_j) = \frac{\partial u_i}{\partial x_i} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} = u_{i,i} \tag{2.5}$$

ve $\underset{\sim}{u}$ vektörünün rotasyonelinin indis notasyonu ile gösterimi,

$$Rot\underset{\sim}{u} = \underset{\sim}{\nabla}\times\underset{\sim}{u} = \left(\frac{\partial}{\partial x_i}\underset{\sim}{e}_i\right)\times\left(u_j\underset{\sim}{e}_j\right) = \frac{\partial u_j}{\partial x_i}(\underset{\sim}{e}_i\times\underset{\sim}{e}_j) = \frac{\partial u_j}{\partial x_i}\varepsilon_{ijk}\underset{\sim}{e}_k \tag{2.6}$$

şeklinde yapılır [15].

Termoelastisitenin yönetici alan denklemlerini birim şekil değiştirme-yer değiştirme, birim şekil değiştirme-uygunluk, denge ve enerji denklemleri oluşturur. Bu denklemler çözüldüğünde sıcaklık, yer değiştirme, birim şekil değiştirme ve gerilme gibi bilinmeyenler elde edilir.

Birim şekil değiştirme - yer değiştirme ilişkileri Eş. 2.7'de,

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2.7}$$

birim şekil değiştirmeler cinsinden yazılan uygunluk denklemleri Eş. 2.8'de,

$$\varepsilon_{ij,kl} + \varepsilon_{kl,ij} - \varepsilon_{ik,jl} - \varepsilon_{jl,ik} = 0 \tag{2.8}$$

denge denklemleri Eş. 2.9'da,

$$\sigma_{ij,j} + F_i = 0 \tag{2.9}$$

yer değiştirme denklemleri ya da Navier denklemleri Eş. 2.10'da,

$$\mu\nabla^2 u_i + (\lambda+\mu)u_{k,ki} - \beta(T-T_0)_{,i} + F_i = 0 \tag{2.10}$$

genelleştirilmiş Hooke kanunu Eş. 2.11'de indis notasyonu ve toplama anlaşımı kullanılarak ifade edilmiştir [16-17].

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + (\lambda\varepsilon_{kk} - \beta(T-T_0))\delta_{ij} \tag{2.11}$$

Yukarıdaki bünye denklemi ifadesinde,

$$\delta_{ij} = \begin{Bmatrix} 1 \rightarrow i = j \\ 0 \rightarrow i \neq j \end{Bmatrix} \tag{2.12}$$

Kronecker deltasını ifade eder ve

$$\beta = \alpha(3\lambda + 2\mu) \tag{2.13}$$

olarak yazılır.

Enerji denklemi Eş. 2.14'te ifade edilmiştir. Enerji denkleminde ısıl ve mekanik değişkenlerin bulunması bağlı (coupled) termoelastisite durumu için geçerlidir. Birçok malzeme için sıcaklığı etkileyen bağlı terimler çok küçüktür ve ihmal edilebilir. Bu durumda Eş. 2.15'te verilen ayrık (uncoupled) enerji denklemi kullanılabilir. Diğer bir basitleştirme olarak kararlı hal durumu düşünülebilir. Bu durumda ısı iletim denklemi Eş. 2.16'daki gibi Laplace denklemine indirgenebilir [16].

$$kT_{,ii} = \rho c\dot{T} + (3\lambda + 2\mu)\alpha T_o \dot{\varepsilon}_{ii} - \rho h \tag{2.14}$$

$$kT_{,ii} = \rho c\dot{T} - \rho h \tag{2.15}$$

$$T_{,ii} = \nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0 \tag{2.16}$$

2.1. İki Boyutlu Termoelastik Cisimler İçin Denklemler

Genel şekil değiştirme ve gerilme durumu üç boyutludur, ancak özel geometrik konfigürasyonlar için iki boyutlu olarak analiz yapılabilir. Yer değiştirme ve gerilme gibi temel nicelikler ve sınır koşulları üçüncü boyuttan bağımsız ise problem iki boyutlu olarak ele alınabilir. Termoelastik problemlerin çözümü için düzlem şekil değiştirme ve düzlem gerilme kabulleri yapılabilir.

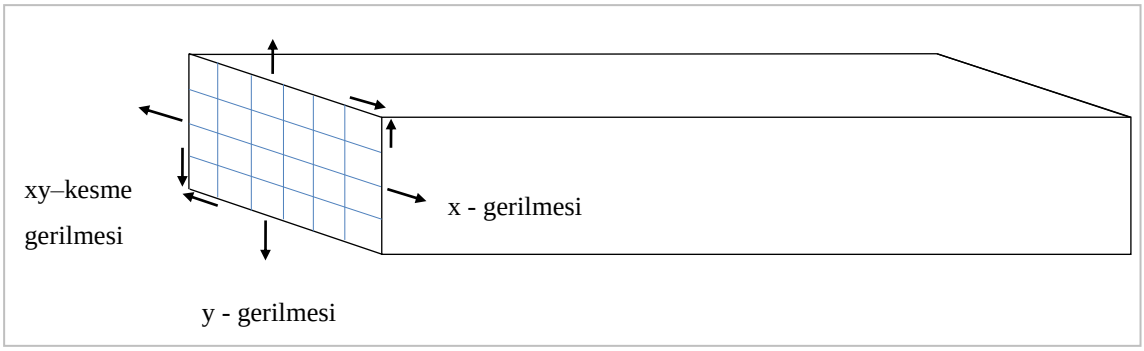
2.1.1. Düzlem şekil değiştirme durumu

x-y düzlemindeki düzlem şekil değiştirme durumu için temel varsayım, *z* yönündeki birim şekil değiştirme alanının sıfır olduğudur. Yer değiştirmeye bağlı şekil değiştirme ilişkileri aşağıda belirtilmiştir [16].

$$\varepsilon_x = \frac{\partial u}{\partial x}, \varepsilon_y = \frac{\partial v}{\partial y}, \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{2.17}$$

$$\varepsilon_z = \varepsilon_{xz} = \varepsilon_{yz} = 0 \tag{2.18}$$

Şekil 2.1'de üçüncü boyutu diğer iki boyutuna göre oldukça büyük olan bir geometri gösterilmektedir. Geometrinin uç sınırlarının boyuna doğrultudaki yerdeğiştirmesi engellendiği takdirde düzlem şekil değiştirme durumunu elde edilebilir. İki boyutlu düzlemde σ_x, σ_y normal gerilmeleri ve τ_{xy} kesme gerilmeleri mevcuttur. Bununla birlikte *z* doğrultusu boyunca şekil değiştirme değerleri sıfırdır. Bu nedenle düzlem şekil değiştirme analizi, yalnızca düzlemde birim şekil değiştirmelere izin verir.



Şekil 2.1. Düzlem şekil değiştirme durumu varsayımı

Düzlem şekil değiştirme durumu için yer değiştirme denklemleri,

$$\mu\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u+(\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)-\beta\left(\frac{\partial(T-T_0}{\partial x}\right)=0 \tag{2.19}$$

$$\mu\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)v+(\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)-\beta\left(\frac{\partial(T-T_0}{\partial y}\right)=0 \tag{2.20}$$

ve düzlem şekil değiştirme durumu için gerilme denklemleri,

$$\sigma_{ij}=\mu(u_{i,j}+u_{j,i})+(\lambda(\varepsilon_x+\varepsilon_y)-\beta(T-T_0))\delta_{ij} \tag{2.21}$$

olarak yazılabilir. Eş. 2.22'de x doğrultusundaki normal gerilme,

$$\sigma_x=(2\mu+\lambda)\left(\frac{\partial u}{\partial x}\right)+\lambda\left(\frac{\partial v}{\partial y}\right)-\beta(T-T_0) \tag{2.22}$$

Eş. 2.23'te y doğrultusundaki normal gerilme,

$$\sigma_y=(2\mu+\lambda)\left(\frac{\partial v}{\partial y}\right)+\lambda\left(\frac{\partial u}{\partial x}\right)-\beta(T-T_0) \tag{2.23}$$

Eş. 2.24'te z doğrultusundaki normal gerilme,

$$\sigma_z=\nu(\sigma_x+\sigma_y)-E\alpha(T-T_0) \tag{2.24}$$

Eş. 2.25'te ise xy kesme gerilmesi ifade edilmiştir.

$$\tau_{xy}=\mu\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right) \tag{2.25}$$

$$\tau_{xz}=\tau_{yz}=0 \tag{2.26}$$

2.1.2. Düzlem gerilme durumu

x-y düzlemindeki düzlem gerilme durumu için temel varsayım, *z* doğrultusundaki normal gerilme ve kesme gerilmelerinin sıfır olduğudur.

$$\sigma_x = \sigma_x(x, y) \tag{2.27}$$

$$\sigma_y = \sigma_y(x, y) \tag{2.28}$$

$$\tau_{xy} = \tau_{xy}\left(x, y\right) \tag{2.29}$$

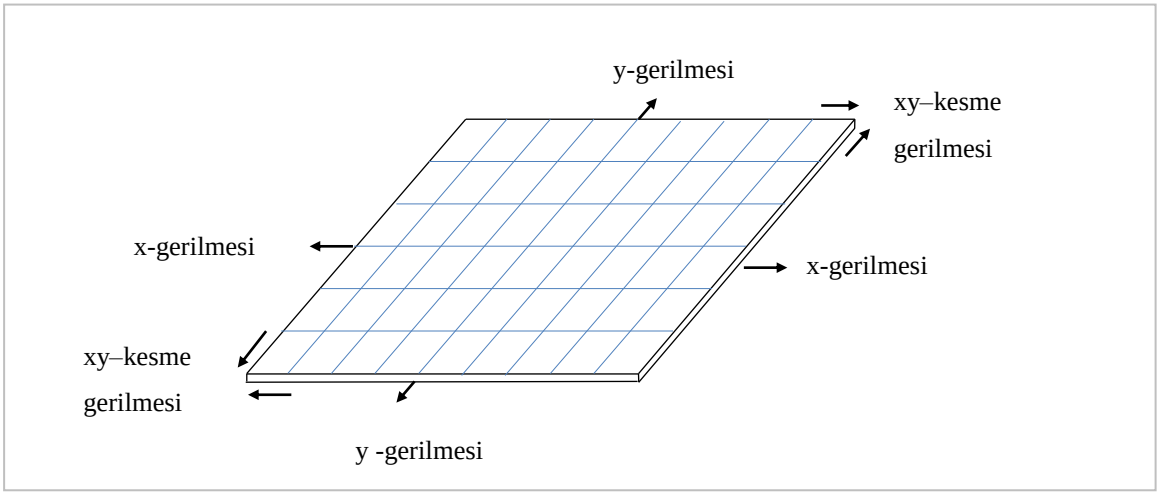
$$\sigma_z = \tau_{xz} = \tau_{yz} = 0 \tag{2.30}$$

Şekil 2.2'deki levha düzlem gerilme durumunu göstermektedir. İki boyutlu düzlemde σ_x ve σ_y normal gerilmeleri ve τ_{xy} kesme gerilmeleri mevcuttur. Bununla birlikte *z* yönünde ise hiçbir gerilme oluşmaz. Düzlem gerilme modeli üçüncü boyutu diğer iki boyutuna göre çok küçük olan cisimler için uygundur.

Düzlem gerilme durumunda düzlem şekil değiştirme durumundan farklı olarak aşağıdaki dönüşümler uygulanır [17].

$$\lambda^* = \frac{2\mu\lambda}{\lambda + 2\mu} \tag{2.31}$$

$$\beta^* = \frac{2\mu\beta}{\lambda + 2\mu} \tag{2.32}$$



Şekil 2.2. Düzlem gerilme durumu varsayımı

Düzlem gerilme durumu için yer değiştirme denklemleri aşağıda belirtilmiştir.

$$\mu\nabla^2 u_i + (\lambda^* + \mu)u_{j,ji} - \beta^* T_{,i} = 0 \tag{2.33}$$

$$\mu\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)u + \left(\frac{2\mu\lambda}{\lambda+2\mu} + \mu\right)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) - \frac{2\mu\beta}{\lambda+2\mu}\left(\frac{\partial(T-T_0)}{\partial x}\right) = 0 \tag{2.34}$$

$$\mu\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)v + \left(\frac{2\mu\lambda}{\lambda+2\mu} + \mu\right)\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) - \frac{2\mu\beta}{\lambda+2\mu}\left(\frac{\partial(T-T_0)}{\partial y}\right) = 0 \tag{2.35}$$

Düzlem gerilme durumu için gerilme denklemleri aşağıda belirtilmiştir.

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + (\lambda^*\varepsilon_{kk} - \beta^*(T-T_0))\delta_{ij} \tag{2.36}$$

$$\sigma_{ij} = \mu(u_{i,j} + u_{j,i}) + (\lambda^*(\varepsilon_x + \varepsilon_y) - \beta^*(T-T_0))\delta_{ij} \tag{2.37}$$

Eş. 2.39'da x doğrultusundaki normal gerilme,

$$\sigma_x = (2\mu + \lambda^*)\left(\frac{\partial u}{\partial x}\right) + \lambda^*\left(\frac{\partial v}{\partial y}\right) - \beta^*(T-T_0) \tag{2.38}$$

$$\sigma_x = \left(2\mu + \frac{2\mu\lambda}{\lambda+2\mu}\right)\left(\frac{\partial u}{\partial x}\right) + \frac{2\mu\lambda}{\lambda+2\mu}\left(\frac{\partial v}{\partial y}\right) - \frac{2\mu\beta}{\lambda+2\mu}(T - T_0) \tag{2.39}$$

Eş. 2.41'de y doğrultusundaki normal gerilme,

$$\sigma_y = (2\mu + \lambda^*)\left(\frac{\partial v}{\partial y}\right) + \lambda^*\left(\frac{\partial u}{\partial x}\right) - \beta^*(T - T_0) \tag{2.40}$$

$$\sigma_y = \left(2\mu + \frac{2\mu\lambda}{\lambda+2\mu}\right)\left(\frac{\partial v}{\partial y}\right) + \frac{2\mu\lambda}{\lambda+2\mu}\left(\frac{\partial u}{\partial x}\right) - \frac{2\mu\beta}{\lambda+2\mu}(T - T_0) \tag{2.41}$$

Eş. 2.42'de ise xy düzlemindeki kesme gerilmesi ifade edilmiştir.

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{2.42}$$

3. SAYISAL YÖNTEMLER

Diferansiyel denklem çözümüne dayalı problemlerin geometri, yükleme ve sınır şartları karmaşıklaştıkca analitik çözüm yapılması zorlaşmaktadır. Matematiksel bir modelin analitik çözümünü bulmak için yapılan basitleştirmeler problemi gerçek çözümünden uzaklaştırabilmektedir. Geometri, yükleme ve sınır şartları açısından gerçekçi bir modelinin yaklaşık sayısal çözümü sayısal modelde yapılan iyileştirmelerle çoğunlukla gerçeğe daha yakın sonuç üretir. Bu yüzden analitik çözümlerin olduğu durumlarda veya daha karmaşık problemler için sayısal yöntemler iyi bir seçenek haline gelmiştir.

Mühendislik hesaplarında kullanılan sayısal yöntemlerden bazıları aşağıda ifade edilmiştir.

1. Sonlu Farklar Yöntemi
2. Sonlu Elemanlar Yöntemi
3. Sonlu Hacimler Yöntemi
4. Ağsız Yöntemler

Bu tez çalışmasında ısıl etki altındaki farklı sınır şartlarına sahip dikdörtgen levhanın ısıl gerilme analizini gerçekleştiren yazılım Python programlama dilinde sonlu farklar yöntemi kullanılarak geliştirilmiştir. Sonlu farklar yöntemi kullanılarak termoelastik gerilme analizi yapılan çalışmalara rastlanmakla birlikte ele alınan problemin geometrisinin dikdörtgen olması ve düzgün bir ızgara sistemi oluşturulabilmesi sonlu farklar yönteminin kullanılmasında etkili olmuştur.

3.1. Sonlu Farklar Yöntemi

Sonlu farklar yönteminde diferansiyel denklemleri oluşturan türev ifadelerinin çözülebilmesi için ilk olarak diferansiyel denklemin ayrıklaştırılması gerekir. Ayrıklaştırma işlemi için Taylor serisi kullanılır. Bir *f(x)* fonksiyonunun $x=x_o$ noktasındaki Taylor serisi açılımı aşağıda ifade edilmiştir.

$$f(x) = f(x_o) + f'(x_o)(x - x_0) + \frac{f''(x_o)}{2!}(x - x_0)^2 + ... + \frac{f^n(x_o)}{n!}(x - x_0)^n \tag{3.1}$$

Sonlu farklar yönteminde kullanılan üç farklı yaklaşım vardır. Bu yaklaşımlar ileri yönde, geri yönde ve merkezi farktır. Birinci ve ikinci dereceden merkezi fark ifadelerinin Taylor serilerine göre elde edilişi aşağıdaki eşitliklerde gösterilmiştir.

Birinci dereceden ileri fark formülasyonunu elde etmek için *f(x)* fonksiyonunun *(x+Δx)* noktasındaki değeri Taylor seri açılımı ile yazılır.

$$f(x+\Delta x) = f(x) + f'(x)(\Delta x) + f''(x)\frac{(\Delta x)^2}{2!} + \ldots + f^n(x)\frac{(\Delta x)^n}{n!} \tag{3.2}$$

Eş. 3.2'den birinci dereceli türev çekilirse,

$$f'(x) = \left(\frac{f(x+\Delta x) - f(x)}{\Delta x} - f''(x)\frac{\Delta x}{2!} - f'''(x)\frac{(\Delta x)^2}{3!} - \ldots - f^n(x)\frac{(\Delta x)^{n-1}}{n!} \right) \tag{3.3}$$

Eş. 3.3'deki yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{3.4}$$

ileri yönde fark formülasyonu elde edilir.

Birinci dereceden geri fark formülasyonunu elde etmek için $f(x)$ fonksiyonunun $(x - \Delta x)$ noktasındaki değeri Taylor seri açılımı ile yazılır.

$$f(x-\Delta x) = f(x) - f'(x)(\Delta x) + f''(x)\frac{(\Delta x)^2}{2!} - f'''(x)\frac{(\Delta x)3}{3!} + \ldots \tag{3.5}$$

Eş. 3.5'ten birinci dereceli türev çekilirse,

$$f'(x) = \left(\frac{f(x) - f(x-\Delta x)}{\Delta x} + f''(x)\frac{\Delta x}{2!} - f'''(x)\frac{(\Delta x)^2}{3!} + \ldots \right) \tag{3.6}$$

Eş. 3.6'dan yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f'(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x} + O(\Delta x) \tag{3.7}$$

geri yönde fark formülasyonu elde edilir.

Birinci dereceden merkezi fark formülasyonunu elde etmek için Eş. 3.2'den Eş. 3.5 çıkartılır.

$$f(x+\Delta x) - f(x-\Delta x) = 2\Delta x f'(x) + 2\frac{(\Delta x)^3}{3!} f'''(x) + \ldots \tag{3.8}$$

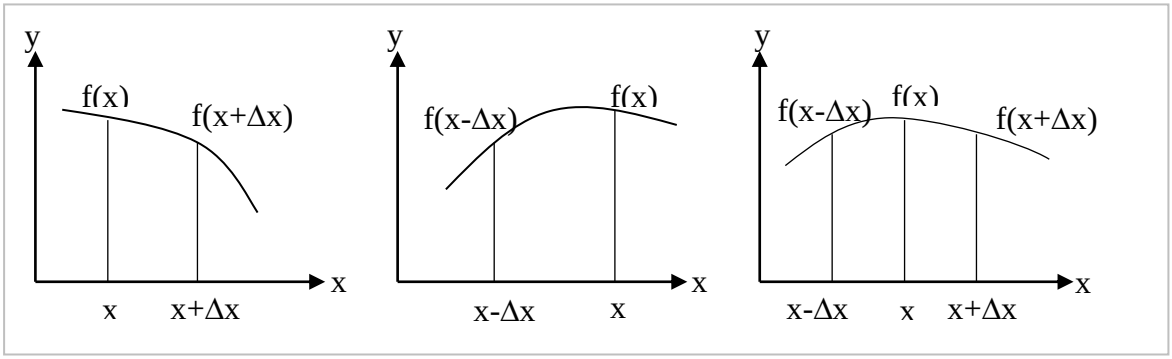
Eş. 3.8'den birinci dereceli türev çekilirse,

$$f'(x) = \left(\frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} - f'''(x)\frac{(\Delta x)^2}{3!} + \ldots \right) \tag{3.9}$$

Eş. 3.9'da yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + O(\Delta x)^2 \tag{3.10}$$

olarak merkezi fark formülasyonu elde edilir. Birinci dereceden ileri yönde, geri yönde ve merkezi fark formülasyonlarında kullanılan ağ noktaları aşağıda gösterilmiştir.



Şekil 3.1. Sonlu fark yaklaşımları

İkinci dereceden ileri fark formülasyonunu elde etmek için $f(x)$ fonksiyonunun $(x + 2\Delta x)$ noktasındaki değeri Taylor seri açılımı ile yazılır.

$$f(x+2\Delta x)=f(x)+f'(x)(2\Delta x)+f''(x)\frac{(2\Delta x)^2}{2!}+\ldots+f^n(x)\frac{(2\Delta x)^n}{n!} \tag{3.11}$$

Eş.3.11'den Eş.3.2 2 ile çarpılıp çıkartılırsa,

$$f(x+2\Delta x)-2f(x+\Delta x)=-f(x)+f''(x)(\Delta x)^2+f'''(x)(\Delta x)^3+\ldots \tag{3.12}$$

Eş. 3.12'den ikinci dereceli türev çekilirse ve yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f''(x)=\frac{f\left(x+2\Delta x\right)-2f(x+\Delta x)+f\left(x\right)}{\left(\Delta x\right)^2}+O\left(\Delta x\right) \tag{3.13}$$

ileri yönde fark formülasyonu elde edilir.

İkinci dereceden geri fark formülasyonunu elde etmek için $f(x)$fonksiyonunun $(x-2\Delta x)$ noktasındaki değeri Taylor seri açılımı ile yazılır.

$$f(x-2\Delta x)=f(x)-f'(x)(2\Delta x)+f''(x)\frac{(2\Delta x)^2}{2!}-f'''(x)\frac{(2\Delta x)^3}{3!}+\ldots \tag{3.14}$$

Eş.3.14'ten Eş.3.5 2 ile çarpılıp çıkartılırsa,

$$f(x-2\Delta x)-2f(x-\Delta x)=-f(x)+f''(x)(\Delta x)^2-f'''(x)(\Delta x)^3+\ldots \tag{3.15}$$

Eş. 3.15'ten ikinci dereceli türev çekilirse ve yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f''(x)=\frac{f\left(x-2\Delta x\right)-2f(x-\Delta x)+f\left(x\right)}{\left(\Delta x\right)^2}+O\left(\Delta x\right) \tag{3.16}$$

geri yönde fark formülasyonu elde edilir.

İkinci dereceden merkezi fark formülasyonunu elde etmek için Eş. 3.2 ve Eş. 3.5 toplanır.

$$f(x+\Delta x)+f(x-\Delta x)=2f(x)+f''(x)(\Delta x)^2+... \tag{3.17}$$

Eş. 3.17'den ikinci dereceli türev çekilirse ve yüksek dereceli türevler kesme hatası terimi olarak gösterilirse,

$$f''(x)=\frac{f\left(x+\Delta x\right)-2f(x)+f\left(x-\Delta x\right)}{\left(\Delta x\right)^2}+O(\Delta x)^2 \tag{3.18}$$

merkezi fark formülasyonu elde edilir.

Bu çalışmada kullanılan diferansiyel denklemler birinci ve ikinci dereceden merkezi fark ifadeleri kullanılarak ayrıklaştırılmıştır. Merkezi fark ifadeleri daha hassas sonuç elde edebilmek için tercih edilmiştir. İleri ve geri farklar yaklaşımlarında kesme hataları, yani seri toplamına dahil edilmeyen toplam ile ilgili hata Δx mertebelidir. Merkezi fark yaklaşımında ise kesme hatası Δx^2 mertebelidir, daha doğru türev sonuçları elde edilir ve kesme hatası ileri ve geri fark ifadelerine göre daha azdır.

Diferansiyel denklemde bulunan türev ifadeleri yerine yukarıda elde edilen ilgili fark ifadeleri yerleştirildiğinde diferansiyel denklem ayrıklaştırılmış olur. Bu ayrıklaştırma sonucu cebirsel denklem sistemi oluşturmaya uygun bir denklem elde edilir. Çözüm alanı düzgün ızgaralara ayrılarak çözüm ağı oluşturulur. Izgaraların kesim noktaları düğüm noktası olarak ifade edilir. Çözüm ağında bulunan her bir düğüm noktası için sonlu farklar denklemi yazıldığında bir cebirsel denklem sistemi elde edilir. Bu denklem sistemi çözüldüğünde her bir düğüm noktası için ilgili hesaplanılması istenen büyüklük elde edilmiş olur.

Sonlu farklar yöntemi kullanılarak gerçekleştirilen analizlerin çözüm aşamasında oluşturulan lineer denklem sisteminin çözümü gerekir. Denklem sistemi doğrudan veya iteratif yöntemlerden biri kullanılarak çözülebilir. Doğrudan yöntemlerden en bilinenleri Gauss yok etme, Gauss–Jordan ve LU ayrıştırma yöntemleridir. Jacobi ve Gauss-Seidel ise en sık kullanılan iteratif yöntemlerdir. Bu çalışmada denklem sisteminin çözümünde Jacobi yöntemi kullanılmıştır. Yöntem ilk olarak tahmini başlangıç değerlerini kullanarak bir sonraki adımdaki bilinmeyen değerleri hesaplar daha sonra elde edilen değerleri kullanarak tekrar yeni değerler elde eder. Bu tekrar eden işlem istenilen hata toleransına ulaşıncaya

kadar devam eder. İstenilen hata toleransına ulaşıldığında işlem sonlandırılır ve bilinmeyen değerler elde edilmiş olur.

3.2. Isı İletim Denkleminin Sonlu Farklar Açılımları

Isı iletimi denklemi, elastik değişikliklerin ısı etkisi yaratmadığı kabulüyle herhangi bir elastik terim içermemektedir. Sıcaklıkta zamana bağlı bir değişim olmadığı kararlı ısı iletim durumu için ısı iletim denklemi Eş. 3.19'da ifade edilmiştir. Isı iletim denkleminin merkezi sonlu farklar ifadeleri kullanılarak ayrıklaştırılması sonucu elde edilen sıcaklık denklemi Eş. 3.22'de gösterilmiştir.

$$T_{,ii} = \nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{3.19}$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta X^2} \tag{3.20}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta Y^2} \tag{3.21}$$

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta X^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta Y^2} = 0 \tag{3.22}$$

3.3. Navier Denklemlerinin Sonlu Farklar Açılımları

Navier denklemi içerisinde sıcaklık ifadeleri yer aldığı için ilk olarak ısı iletim denklemi çözülmüştür. Düzlem gerilme durumu için yer değiştirme denklemleri bir önceki bölümde Eş. 2.34 ve Eş. 2.35'te ifade edilmiştir. Yer değiştirme denklemindeki benzer türev ifadelerinin ortak paranteze alınmasıyla Eş. 3.23 ve Eş. 3.24 elde edilmiştir.

$$\left(\frac{2\mu\lambda}{\lambda+2\mu} + 2\mu\right)\frac{\partial^2 u}{\partial x^2} + \mu\frac{\partial^2 u}{\partial y^2} + \left(\frac{2\mu\lambda}{\lambda+2\mu} + \mu\right)\frac{\partial^2 v}{\partial x \partial y} = \frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha\frac{\partial(T-T_0)}{\partial x} \tag{3.23}$$

$$\left(\frac{2\mu\lambda}{\lambda+2\mu}+2\mu\right)\frac{\partial^2 v}{\partial y^2}+\mu\frac{\partial^2 v}{\partial x^2}+\left(\frac{2\mu\lambda}{\lambda+2\mu}+\mu\right)\frac{\partial^2 u}{\partial x\partial y}=\frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha\frac{\partial(T-T_0)}{\partial y} \tag{3.24}$$

Aşağıda yer değiştirme denkleminde bulunan türev ifadelerinin merkezi fark açılımları ifade edilmiştir.

$$\frac{\partial^2 u}{\partial x^2}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta X^2} \tag{3.25}$$

$$\frac{\partial^2 u}{\partial y^2}=\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta Y^2} \tag{3.26}$$

$$\frac{\partial^2 v}{\partial x\partial y}=\frac{v_{i+1,j+1}-v_{i-1,j+1}-v_{i+1,j-1}+v_{i-1,j-1}}{4\Delta X\Delta Y} \tag{3.27}$$

$$\frac{\partial(T-T_0)}{\partial x}=\frac{(T-T_0)_{i+1,j}-(T-T_0)_{i-1,j}}{2\Delta X} \tag{3.28}$$

$$\frac{\partial^2 v}{\partial y^2}=\frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{\Delta Y^2} \tag{3.29}$$

$$\frac{\partial^2 v}{\partial x^2}=\frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{\Delta X^2} \tag{3.30}$$

$$\frac{\partial^2 u}{\partial x\partial y}=\frac{u_{i+1,j+1}-u_{i-1,j+1}-u_{i+1,j-1}+u_{i-1,j-1}}{4\Delta X\Delta Y} \tag{3.31}$$

$$\frac{\partial(T-T_0)}{\partial y}=\frac{(T-T_0)_{i,j+1}-(T-T_0)_{i,j-1}}{2\Delta Y} \tag{3.32}$$

Navier denkleminin merkezi sonlu fark ifadeleri kullanılarak ayrıklaştırılması sonucu elde edilen yer değiştirme denklemi Eş. 3.33 ve Eş. 3.34'de gösterilmiştir.

$$\left(\frac{4\mu\lambda+4\mu^2}{\lambda+2\mu}\right)\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta X^2}+\mu\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta Y^2}+\left(\frac{3\mu\lambda+2\mu^2}{\lambda+2\mu}\right)$$
$$\frac{v_{i+1,j+1}-v_{i-1,j+1}-v_{i+1,j-1}+v_{i-1,j-1}}{4\Delta X\Delta Y}=\frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha\frac{(T-T_0)_{i+1,j}-(T-T_0)_{i-1,j}}{2\Delta X} \quad (3.33)$$

$$\left(\frac{4\mu\lambda+4\mu^2}{\lambda+2\mu}\right)\frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{\Delta Y^2}+\mu\frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{\Delta X^2}+\left(\frac{3\mu\lambda+2\mu^2}{\lambda+2\mu}\right)$$
$$\frac{u_{i+1,j+1}-u_{i-1,j+1}-u_{i+1,j-1}+u_{i-1,j-1}}{4\Delta X\Delta Y}=\frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha\frac{(T-T_0)_{i,j+1}-(T-T_0)_{i,j-1}}{2\Delta Y} \quad (3.34)$$

3.4. Gerilme Denklemlerinin Sonlu Farklar Açılımları

Gerilme denklemleri Lame sabitleri, yer değiştirme ve sıcaklık ifadelerinden oluşmuştur. Bu yüzden ısı iletim denklemi ve Navier denklemi çözüldükten sonra gerilme denklemi çözülmüştür. Düzlem gerilme durumu için gerilme denklemleri bir önceki bölümde Eş. 2.39, Eş. 2.41 ve Eş. 2.42'de ifade edilmiştir. Aşağıda gerilme denkleminde bulunan türev ifadelerinin merkezi fark açılımları ifade edilmiştir.

$$\frac{\partial u}{\partial x}=\frac{u_{i+1,j}-u_{i-1,j}}{2\Delta X} \quad (3.35)$$

$$\frac{\partial v}{\partial y}=\frac{v_{i,j+1}-v_{i,j-1}}{2\Delta Y} \quad (3.36)$$

$$\frac{\partial u}{\partial y}=\frac{u_{i,j+1}-u_{i,j-1}}{2\Delta Y} \quad (3.37)$$

$$\frac{\partial v}{\partial x}=\frac{v_{i+1,j}-v_{i-1,j}}{2\Delta X} \quad (3.38)$$

Gerilme denkleminin merkezi sonlu farklar ifadeleri kullanılarak ayrıklaştırılması sonucu elde edilen gerilme denklemleri Eş. 3.39, Eş. 3.40 ve Eş. 3.41'de gösterilmiştir.

$$\sigma_x = \left(\frac{2\mu\lambda}{\lambda+2\mu}+2\mu\right)\frac{u_{i+1,j}-u_{i-1,j}}{2\Delta X}+\left(\frac{2\mu\lambda}{\lambda+2\mu}\right)\frac{v_{i,j+1}-v_{i,j-1}}{2\Delta Y}-\frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha(T-T_0)_{i,j} \quad (3.39)$$

$$\sigma_y = \left(\frac{2\mu\lambda}{\lambda+2\mu}+2\mu\right)\frac{v_{i,j+1}-v_{i,j-1}}{2\Delta Y}+\left(\frac{2\mu\lambda}{\lambda+2\mu}\right)\frac{u_{i+1,j}-u_{i-1,j}}{2\Delta X}-\frac{2\mu}{\lambda+2\mu}(3\lambda+2\mu)\alpha(T-T_0)_{i,j} \quad (3.40)$$

$$\tau_{xy} = \mu\left(\frac{u_{i,j+1}-u_{i,j-1}}{2\Delta Y}+\frac{v_{i+1,j}-v_{i-1,j}}{2\Delta X}\right) \quad (3.41)$$

4. ABAQUS YAZILIMI İLE ISIL GERİLME ANALİZİ

Matematikte ve mühendislikte karşılaşılan birçok problemin kapalı formda analitik çözümünün elde edilememesi sayısal yöntemlerin çıkış noktasını oluşturmuştur. En yaygın sayısal yöntemlerden biri olan sonlu elemanlar yöntemi, modeli üzerinde hesap yapılabilecek küçük sonlu elemanlara bölerek analiz yapma esasına dayanır. Tarihi gelişimi 1940'lı yılların başlarına dayanan sonlu elemanlar yöntemi bilgisayar teknolojisinin gelişmesiyle önemini koruyarak günümüze kadar gelmiştir. Günümüzde sonlu elemanlar yöntemini kullanan ABAQUS, Ansys, Nastran ve Hypermesh gibi sonlu eleman yazılımları geliştirilmiştir. Bu tez kapsamında geliştirilen yazılımın ısıl gerilme analiz sonuçlarını karşılaştırmak için ABAQUS [18] sonlu eleman yazılımı kullanılmıştır.

4.1. ABAQUS Sonlu Elemanlar Yazılımı

ABAQUS, basit doğrusal analizlerden zorlu doğrusal olmayan benzeştirmelere kadar geniş yelpazede problemleri ele alabilen, sonlu elemanlar yöntemine dayanan mühendislik simülasyon programıdır. ABAQUS, neredeyse her geometriyi modelleyebilen geniş bir eleman kütüphanesini içerir. Metal, kauçuk, polimer, kompozit, betonarme, kırılabilir veya esnek köpük gibi malzemeler dâhil olmak üzere çoğu tipik mühendislik malzemesinin davranışını benzeştirebilen eşit derecede kapsamlı bir malzeme model listesine sahiptir. Genel amaçlı bir simülasyon aracı olarak tasarlanan ABAQUS, yapısal problemlerden daha fazlasını incelemek için kullanılır. Isı transferi, kütle difüzyonu, elektrikli bileşenlerin ısıl yönetimi, akustik, zemin mekaniği, piezoelektrik analiz, elektromanyetik analiz ve akışkan dinamiği gibi farklı alanlarda problemler analiz edebilir [19].

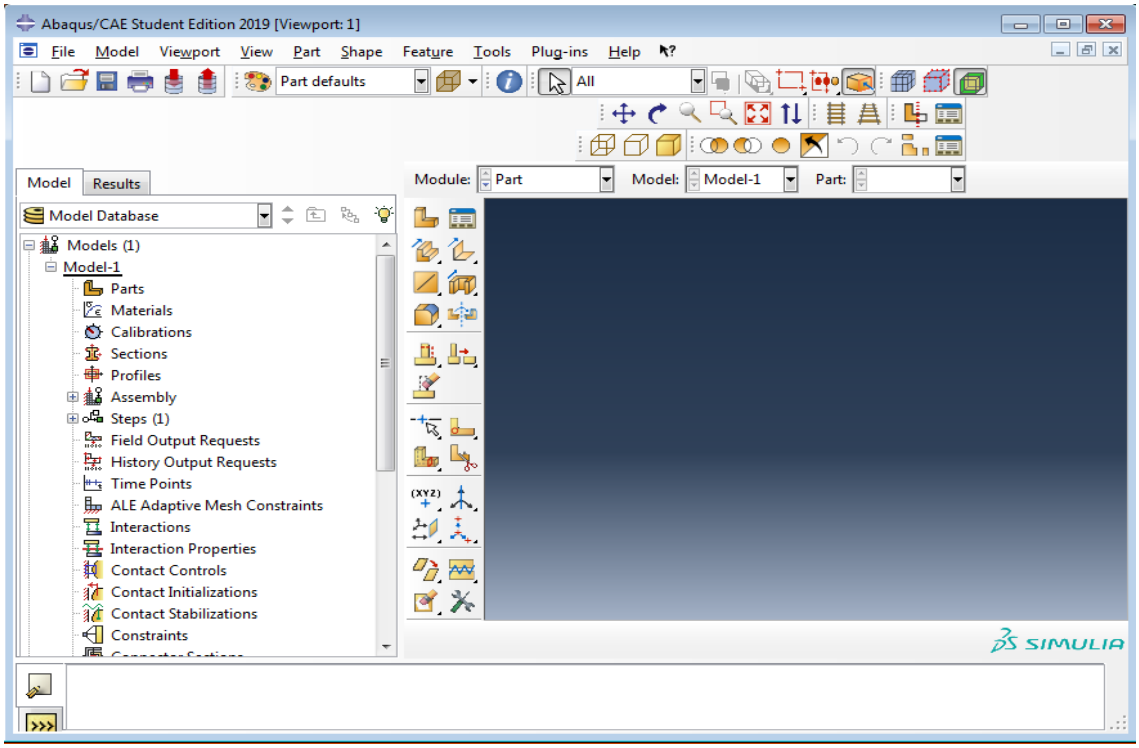
ABAQUS, analiz edilecek yapının geometrisini üreterek veya dışarıdan alarak modellerin hızlı ve kolay bir şekilde oluşturulmasını ve geometrinin çözüm ağı oluşturulabilir bölgelere ayrılmasını sağlar. Oluşturulan geometriye malzeme özellikleri, yükler ve sınır şartları atanır. ABAQUS, geometriyi sonlu elemanlar ağına bölmek ve elde edilen modeli doğrulamak için çok güçlü seçenekler içerir. Model tamamlandığında analiz işlemleri yapılabilir, izlenebilir ve kontrol edebilir. Görselleştirme modülü daha sonra sonuçları yorumlamak için kullanılır [19].

4.2. Dikdörtgen Levhanın Isıl Gerilme Analizi

Bu bölümde ABAQUS sonlu elemanlar yazılımı kullanılarak ısıl gerilme analizinin nasıl yapıldığı anlatılmak için lineer sıcaklık dağılımı etkisi altındaki Bölüm 6.1'de tanımlanan Python-1 örneği seçilmiştir.

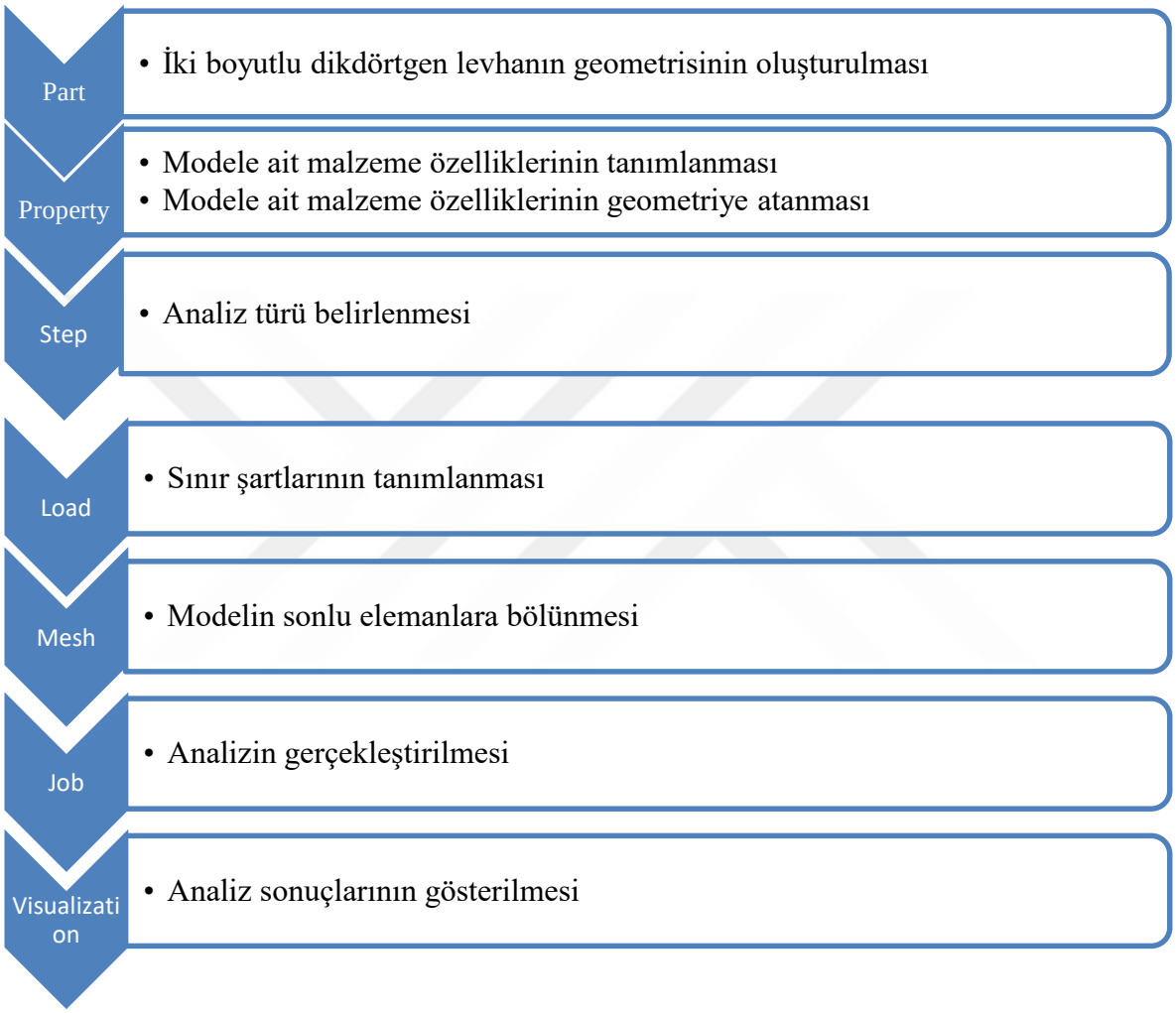
Python-1 örneğinde yer alan ortam düşey kenarların *x* yönünde, yatay kenarların *y* yönünde yer değiştirmelerinin engellendiği, 2000 mm uzunluğundaki ve 1000 mm genişliğindeki dikdörtgen bir levhadır. Lineer elastik malzeme özelliklerine sahip levhanın Elastisite modülü $2\times10^{5}\,MPa$, Poisson oranı 0,3, ısıl genleşme katsayısı $1,3\times10^{-5}C^{-1}$'dir. Kararlı halde levha içerisinde üniform bir sıcaklık dağılımı oluşturmak için, sol ve sağ kenardan $30^{\circ}C$ sıcaklık uygulanmıştır. Levhanın alt ve üst kenarları izole kenar seçilmiştir. Ortamın başlangıç sıcaklığı $0^{\circ}C$'dir.

Şekil 4.1'de gösterilen ABAQUS ana ekranındaki model ağacı kullanılarak veya modül menüsü ile birlikte araç kutusu bölgesi kullanılarak istenilen model oluşturulur. Modülleri takip edilerek analizi gerçekleştirebilmek program kullanıcılarına büyük kolaylık sağlar.



Şekil 4.1. ABAQUS açılış penceresi

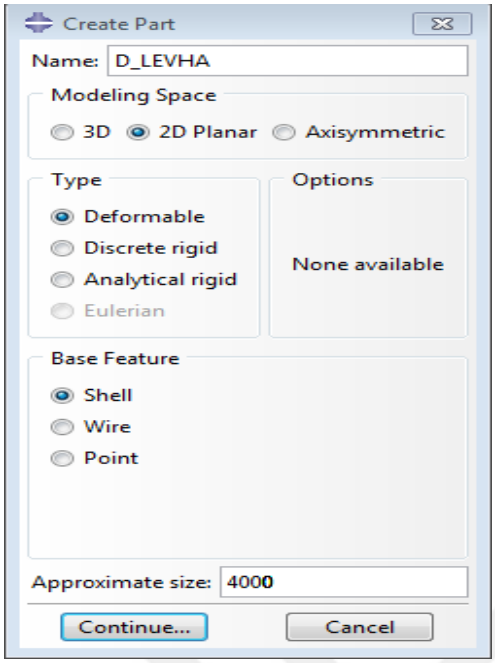
Isıl etki altındaki dikdörtgen levhanın ısıl gerilme analizinin ABAQUS sonlu elemanlar yazılımında gerçekleştirilebilmesi için gerekli işlem adımları Şekil 4.2'de gösterilmiştir. Model tek bir parçadan oluştuğu için işlem adımlarında "Assembly" ve "Interaction" modülleri gösterilmemiştir.



Şekil 4.2. ABAQUS işlem adımları

4.2.1. Parça (Part) modülünün kullanılması

Dikdörtgen levha geometrisinin modellenmesi "Parts" modülü ile gerçekleştirilmiştir. "Parts" modülünde "Create Part" diyalog kutusunda seçilen radio butonlar Şekil 4.3'te gösterilmiştir. Türkçe karakterler kullanmadan ve üzerinde çalıştığımız parçayı nitelendirecek bir parça ismi verilmiştir. Model iki boyutlu olduğu için uzayda kapladığı boyutu 2D, şekil değiştirebildiği için modelin tipi şekil değiştirebilir, üçüncü boyutu çok küçük olduğu için model özelliği kabuk seçilmiştir.



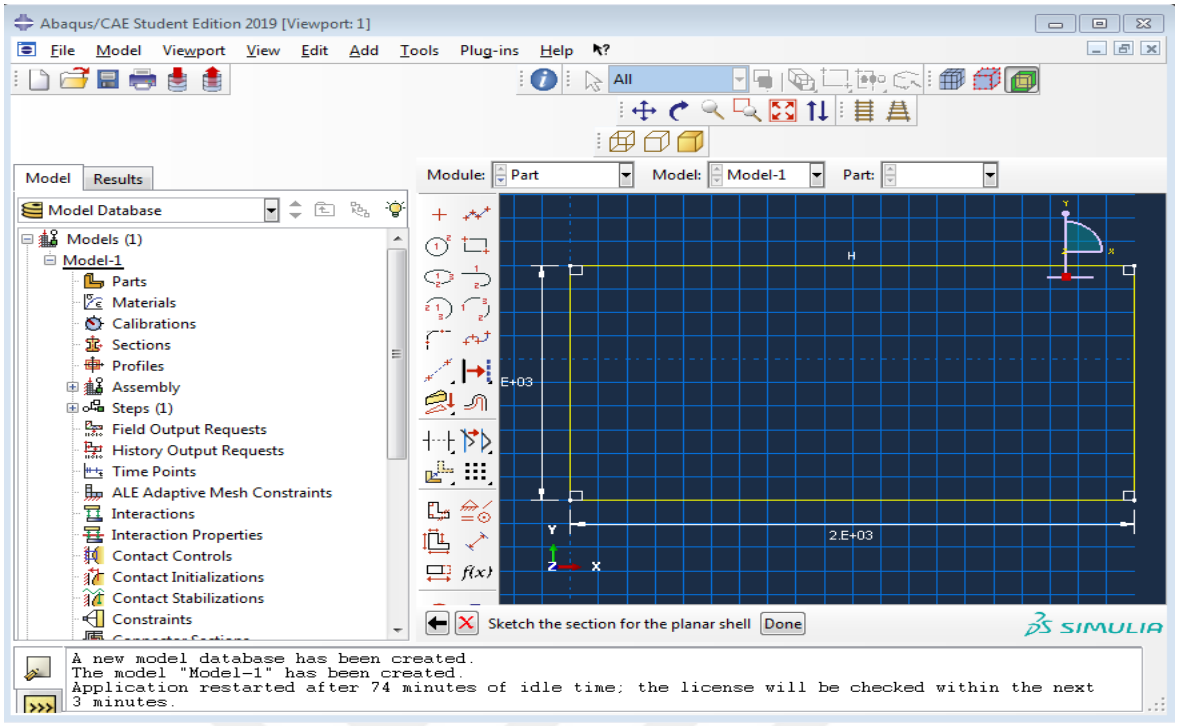
Şekil 4.3. ABAQUS 2D model oluşturma

ABAQUS'un belirli bir birim sistemi yoktur. Bu yüzden ABAQUS'a veri girerken birbiriyle uyumlu birimler kullanılmalıdır. ABAQUS içinde kullanılabilecek birim sistemleri Çizelge 4.1'de verilmiştir [19]. Bu çalışmada SI(mm) birim sistemi kullanılmıştır.

Çizelge 4.1. ABAQUS uyumlu birimler

Quantity	SI	SI (mm)	US Unit (ft)	US Unit (inch)
Length	m	mm	ft	in
Force	N	N	lbf	lbf
Mass	kg	tonne (10^3 kg)	slug	lbf s^2/in
Time	s	s	s	s
Stress	Pa (N/m^2)	MPa (N/mm^2)	lbf/ft^2	psi (lbf/in^2)
Energy	J	mJ (10^{-3} J)	ft lbf	in lbf
Density	kg/m^3	$tonne/mm^3$	$slug/ft^3$	lbf s^2/in^4

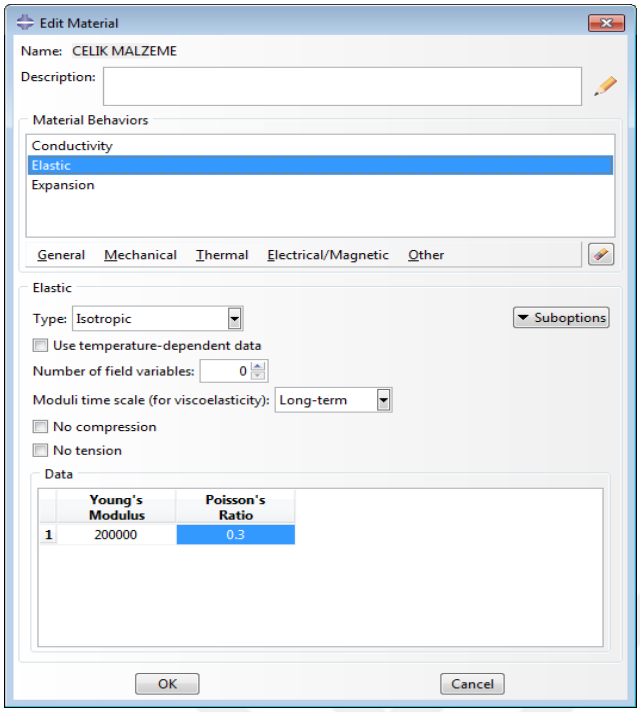
Parçanın isimlendirilmesi ve modelin boyutu, tipi ve özelliklerinin belirlenmesinin ardından "rectangle" aracı ile dikdörtgen çizilmiş, araç kutusundan "Add Dimension" aracı ile dikdörtgen levhanın boyutlandırması yapılmıştır. Levha uzunluğu 2000 mm, levha genişliği 1000 mm olarak seçilmiştir.



Şekil 4.4. ABAQUS çizim alanı

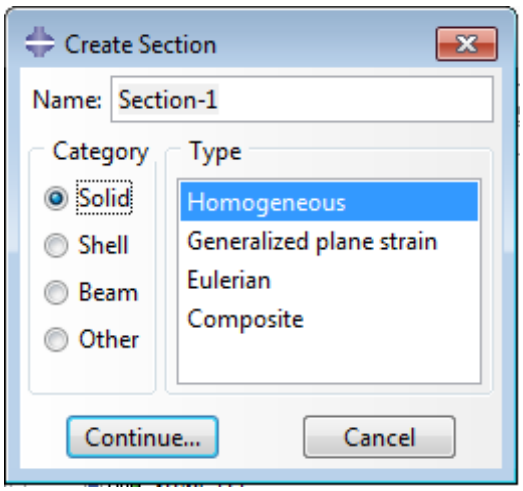
4.2.2. Özellik (Property) modülünün kullanılması

Ele alınan levha için lineer-elastik malzeme oluşturulmuştur. Modele ait malzeme özelliklerinin tanımlanması "Property" modülünde "Create Material" ile gerçekleştirilmiştir. Malzeme ismi, Mechanical – Elasticity – Elastic yolu izlenerek Young modulü ve Poisson oranı sırasıyla 200000 MPa ve 0,3 olarak girilmiştir. Mechanical – Expansion yolu izlenerek malzemenin lineer ısıl genleşme katsayısı 1,3E-05 C^{-1} olarak girilmiş, Thermal – Conductivity yolu izlenerek malzemenin iletkenliği 50 olarak girilmiştir.



Şekil 4.5. ABAQUS malzeme girişi

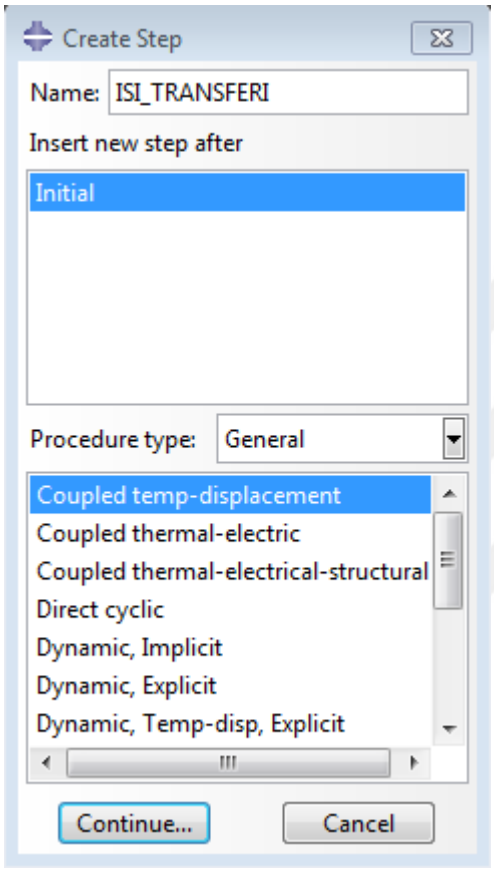
Modele ait kesit özelliklerinin tanımlanması “Property” modülü ile gerçekleştirilmiştir. “Property” modülünde “Create Section” diyalog kutusunda geometriye atanacak kesit özellikleri belirlendikten sonra levhanın kalınlığı 0,01 mm olarak girilmiştir.



Şekil 4.6. ABAQUS kesit özellikleri tanımlama

4.2.3. Adım (Step) modülünün kullanılması

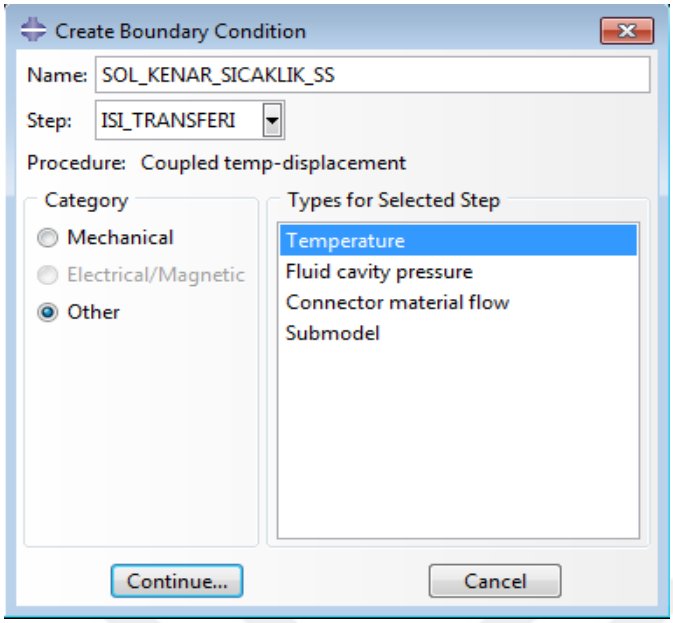
Analiz türünün belirlenmesi “Step” modülü ile gerçekleştirilmiştir. Şekil 4.7’de gösterilen “Step” modülündeki “Create Step” diyalog kutusu ile analiz türü ve “Steady State” seçimleri yapılmıştır.



Şekil 4.7. ABAQUS analiz türünü belirleme

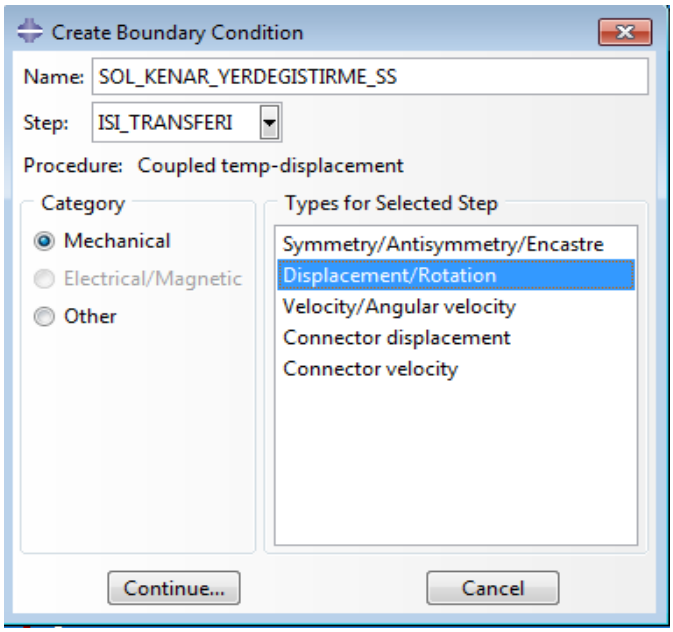
4.2.4. Yük (Load) modülünün kullanılması

Python-1 örneğinde kullanılan sıcaklık sınır şartları ve yer değiştirme sınır şartlarının modelde tanımlanması “Load” modülü ile gerçekleştirilmiştir. Buradaki “Create Boundary Condition” diyalog kutusu Şekil 4.8’de gösterilmiştir. Sınır şartı türü belirlendikten sonra sınır şartının uygulanacağı levha kenarları seçilerek ilgili sıcaklık sınır şartı 30℃ olarak girilmiştir.



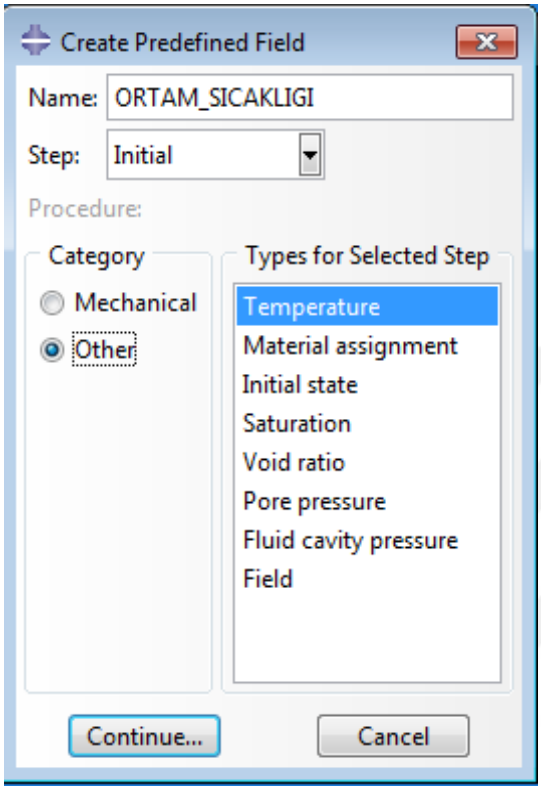
Şekil 4.8. ABAQUS sıcaklık sınır şartı belirleme

Ayrıca yer değiştirme sınır şartları da burada tanımlanmıştır. Şekil 4.9'da gösterilen diyalog kutusunda yer alan yer değiştirme ile ilgili seçenek ve sınır şartının uygulanacağı levha kenarlarının seçimi sonrasında, sol ve sağ kenarlar için U1= 0, alt ve üst kenarlar için U2=0 girilmiştir.



Şekil 4.9. ABAQUS yer değiştirme sınır şartı belirleme

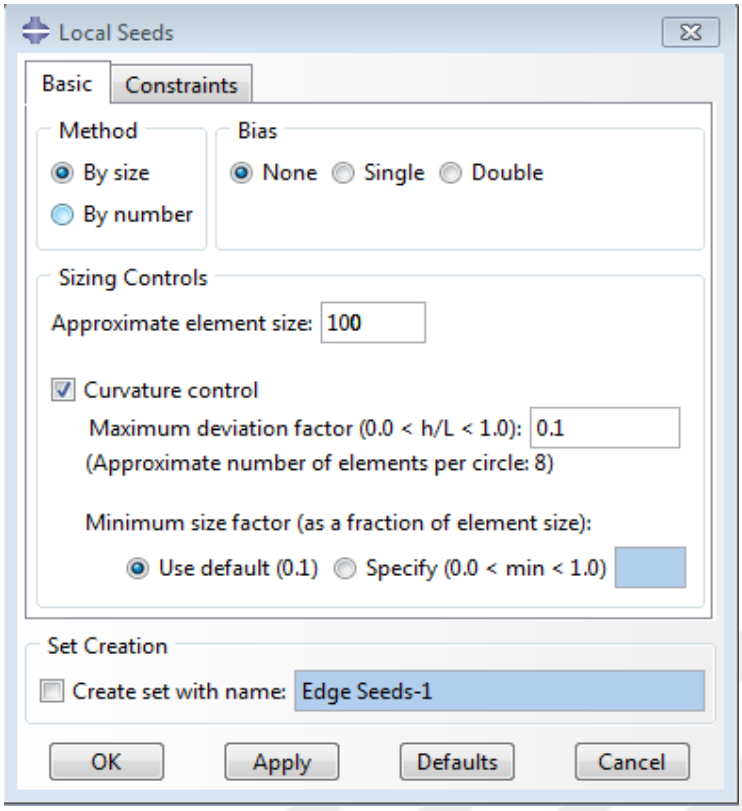
Son olarak başlangıç sıcaklık şartının belirlenmesi aşamasında “Load” modülünde “Create Predefined Field” diyalog kutusunda ilgili sıcaklık seçeneğinin seçilmesi ve uygulanacak alanın belirlenmesi ardından, başlangıç sıcaklık değeri 0℃ olarak girilmiştir.



Şekil 4.10. ABAQUS ortam sıcaklığı sınır şartını belirleme

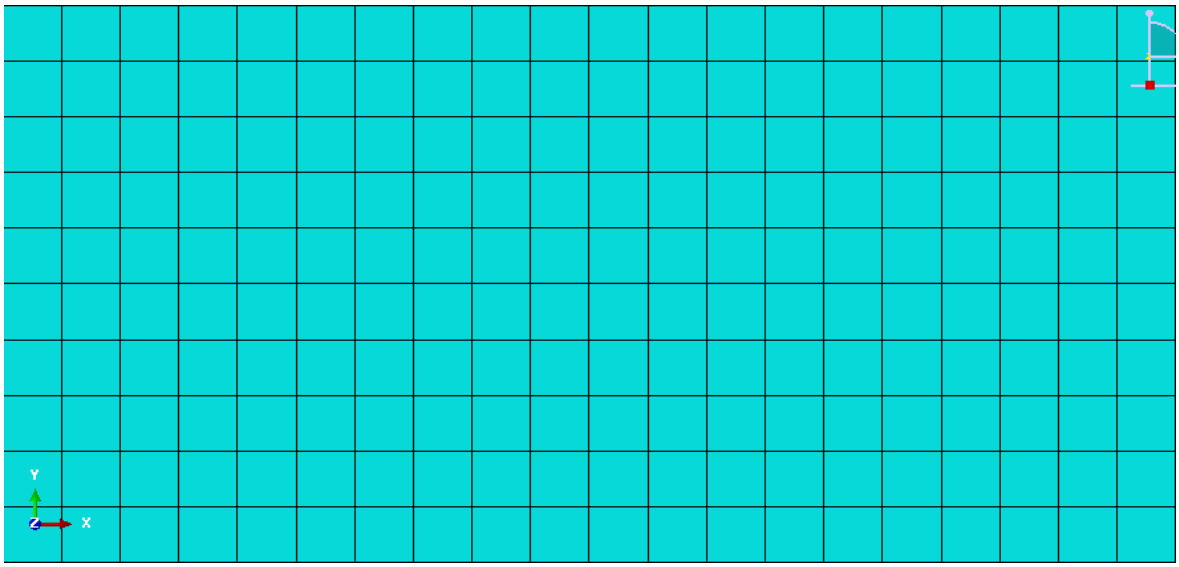
4.2.5. Ağ (Mesh) modülünün kullanılması

Sonlu elemanlar ağının oluşturulması “Mesh” modülü ile gerçekleştirilmiştir. Şekil 4.11’de gösterilen “Seed” menüsünde kenarlar üzerinde yer alacak düğüm sayısının belirlenmesi veya oluşturulacak bir elemanın yaklaşık boyutunun belirlenmesi ile ilgili seçenekler yer almaktadır. Burada yaklaşık eleman boyutu 100 girilerek 21x11’lik bir çözüm ağı oluşturulmuştur.



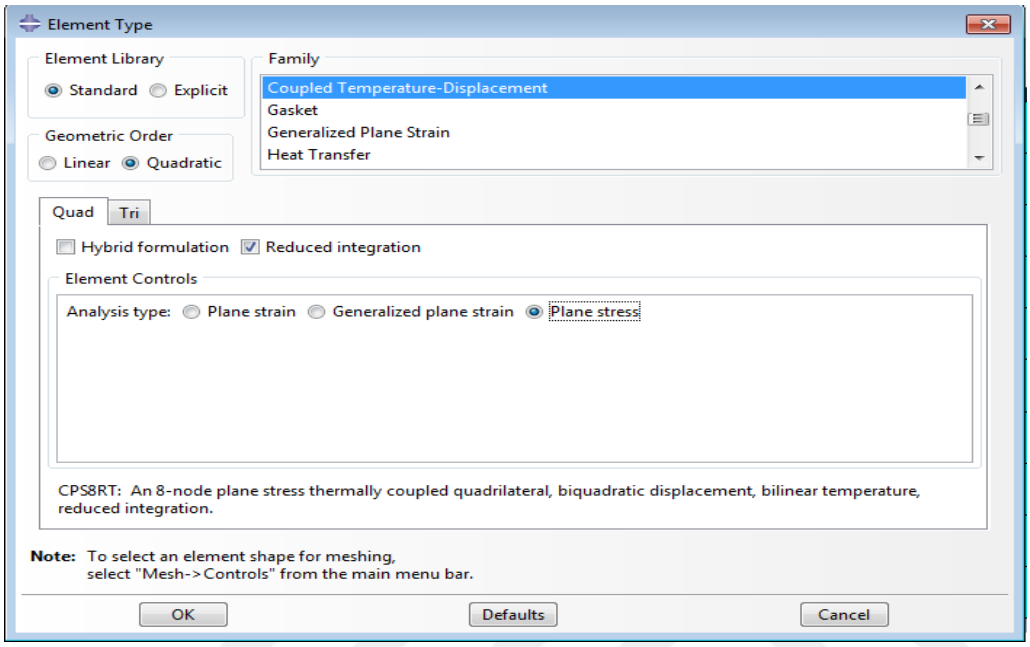
Şekil 4.11. Abaqus çözüm ağı oluşturma

"Mesh" modülünden "Mesh Part" ile çözüm ağı oluşturulması tamamlanmıştır. Oluşturulan bu çözüm ağı Şekil 4.12'de gösterilmiştir.



Şekil 4.12. Abaqus çözüm ağı oluşturulmuş dikdörtgen levha

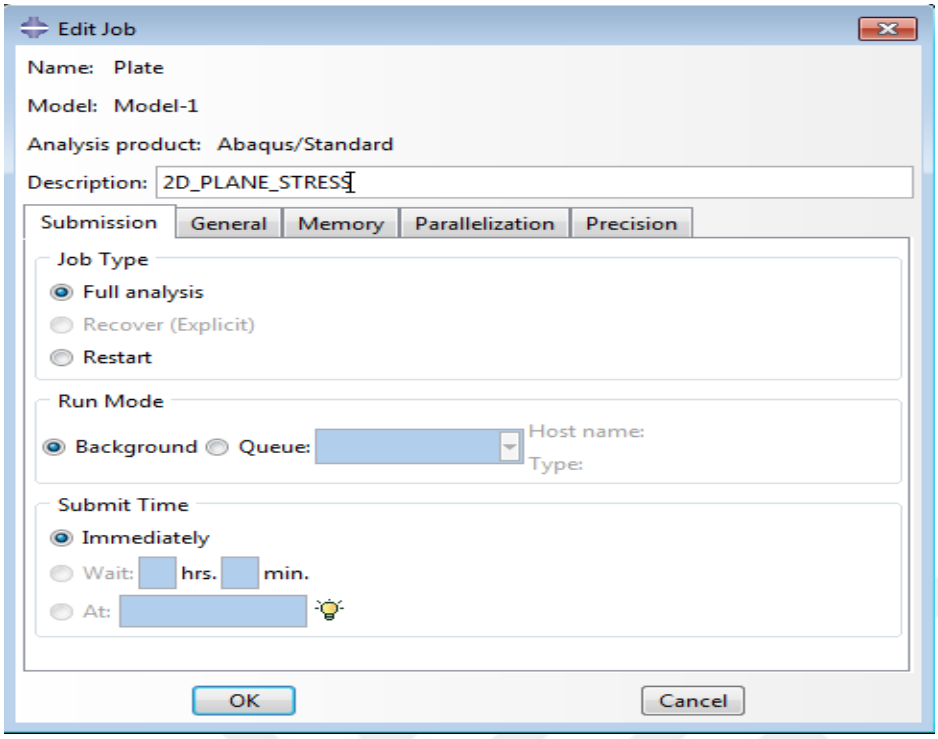
Şekil 4.13'te sonlu eleman ağında kullanılacak eleman tipinin belirlenmesi için yine "Mesh" modülünde yer alan "Assign Element Type" diyalog kutusu ve analizde kullanılan sonlu eleman tipi gösterilmiştir.



Şekil 4.13. ABAQUS eleman tipi belirleme

4.2.6. İş (Job) modülünün kullanılması

Geometrinin, malzeme özelliklerinin, sonlu elemanlar ağı özelliklerinin, yükleme ve sınır şartlarının belirlenmesinin ardından çözüme hazır hale getirilen modelin analizi "Job" modülü ile gerçekleştirilmiştir. Şekil 4.14'te gösterilen "Job" modülünde yer alan "Create Job" diyalog kutusunda çözüm için tanımlanması gereken seçenekler yer almaktadır.

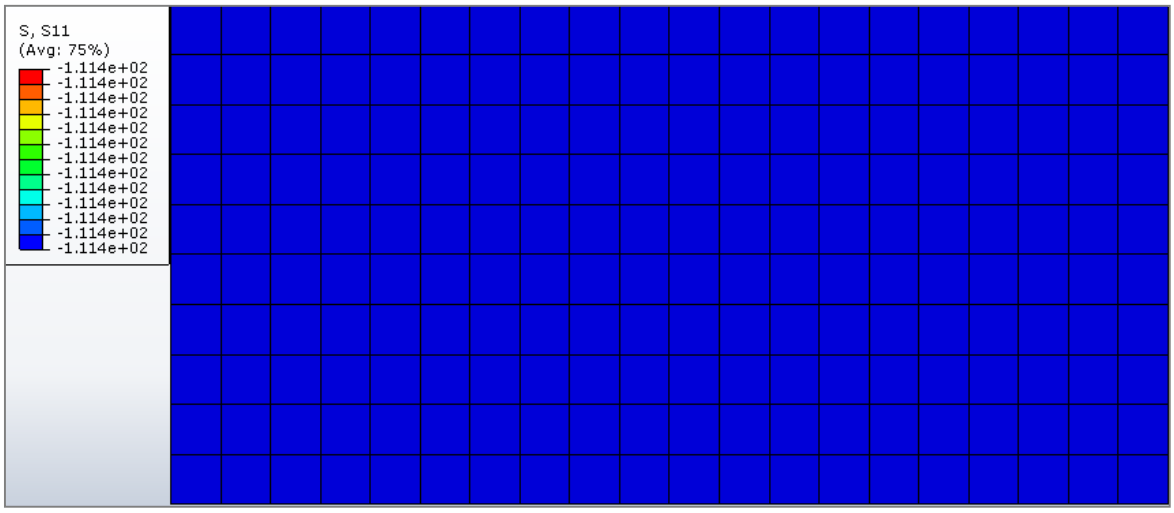


Şekil 4.14. ABAQUS iş modülü

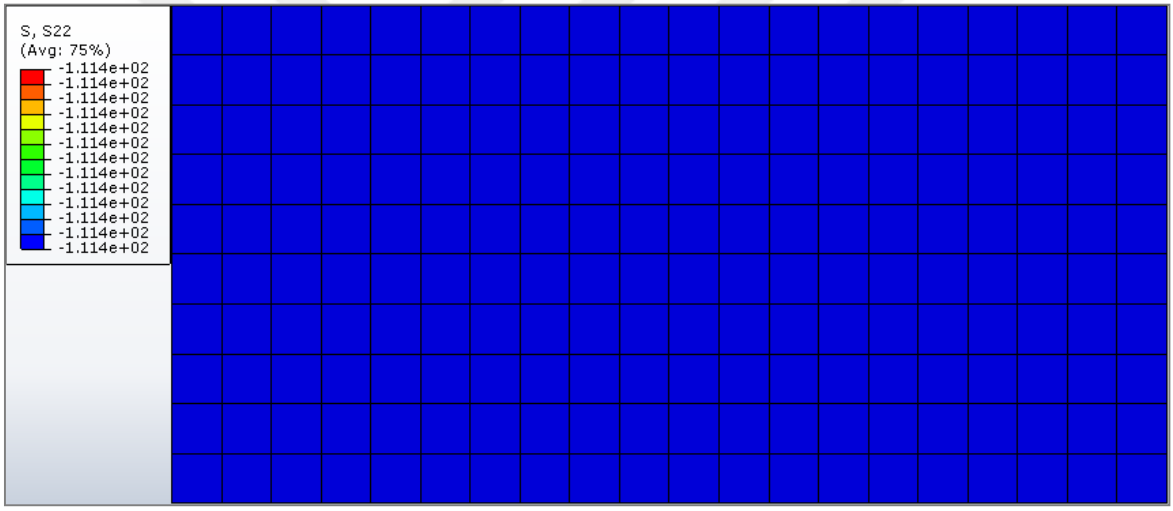
Modelin analizi başlatılmadan önce "Job" modülünde yer alan "Job Manager" ile veriler kontrol edilmiştir. Sonrasında modelin çözümü başlatılmış, başarılı bir şekilde tamamlandıktan sonra sonuçları görüntülemek için "Results" seçeneği kullanılmıştır.

4.2.7. Görselleştirme (Visualization) modülünün kullanılması

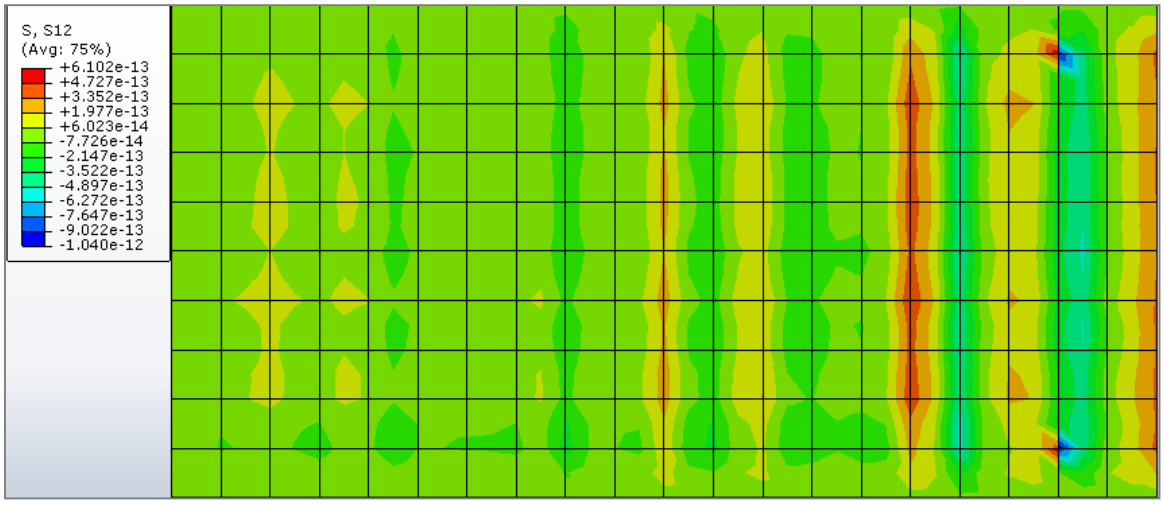
İş başarıyla tamamlandıktan sonra sonuçların görselleştirilmesi "Visualization" modülü ile yapılmaktadır. Ele alınan levhada zamana bağlı sıcaklık değişimi olmadığı durumda levhanın tüm noktalarında sıcaklık değeri 30℃'ye ulaşmıştır. Python-1 örneğinin üniform sıcaklık dağılımı için ABAQUS sonlu elemanlar yazılımı kullanılarak elde edilen ısıl gerilme sonuçları *x* doğrultusundaki normal gerilme için Şekil 4.15'te, *y* doğrultusundaki normal gerilme için Şekil 4.16'da ve xy kesme gerilmesi için Şekil 4.17'de gösterilmiştir.



Şekil 4.15. ABAQUS x doğrultusundaki normal gerilme sonuçları



Şekil 4.16. ABAQUS y doğrultusundaki normal gerilme sonuçları



Şekil 4.17. ABAQUS xy kesme gerilmesi sonuçları

5. PYTHON PROGRAMLAMA DİLİ VE GELİŞTİRİLEN YAZILIM

5.1. Python Programlama Dili

Python, 1980'lerin sonlarında, Hollanda'daki Centrum Wiskunde & Informatica'daki (CWI) Guido Van Rossum tarafından, ABC dilinin halefi olarak geliştirilmeye başlanmıştır. Sadeliği ve öğrenme kolaylığını güçlü yeteneklerle ve performanslarla birleştiren Python, dünyadaki en popüler programlama dillerinden biridir [20]. Python sade bir dil yapısına sahip olmasından dolayı öğrenmesi ve kodlanması kolay, girintili kod yazımı sayesinde okunması basit bir programlama dilidir. Python'un yorumlanan kod yapısı ve işletim sistemleri arası çalışması ile hızlı uygulama geliştirme için ideal bir dildir. Python içerisinde nesne yönelimli programlama, klasik yapısal programlama veya fonksiyonel programlama yapılabilmektedir. Python başlangıç seviyesinde programlama yapacak kişilerin kolaylıkla adapte oldukları bir dil olarak görülmektedir.

5.1.1. Python kullanım alanları

Python çok çeşitli alanlarda kullanılan bir programlama dilidir. Bunların başında veri analizi, makine öğrenimi ve yapay zekâ, web geliştirme, ara yüz geliştirme, ağ ve soket programcılığı, örümcek türü yazılımlar gelir. Son yıllarda artan veri bilimi uygulamaları ile birlikte Python analiz ve istatistik alanlarında sıklıkla kullanılmaya başlanmıştır. Python için geliştirilen yapay zekâ kütüphaneleri onu günümüzün popüler konularından olan yapay zekâ alanında öne çıkarmıştır.

Python programlama dili için geliştirilmiş önemli sayıda kullanılabilir kütüphane bulunmaktadır. Tez çalışmasında bu kütüphanelerden Numpy ve Matplotlib kullanılmıştır. Numpy ve Matplotlib kütüphanelerini Python içerisinde kullanabilmek için Pythton'a aktarılması gereklidir. Karmaşık matematik işlem ve uygulamaları Numpy kütüphanesi ile kolaylıkla kodlanabilmekte ve Python listelerinden daha hızlı sonuç üretilebilmektedir. Matplotlib kütüphanesi ise elde edilen verilerin görselleştirilmesi ve ihtiyaç duyulan grafiklerin oluşturulmasının rahat bir şekilde gerçekleştirilmesini sağlayan bir kütüphanedir.

5.1.2. PyCharm

Bilgisayar programlarının geliştirilmesi için gerekli ortamı sağlayan yazılım geliştirme ortamı (IDE) olarak tanımlanmaktadır. PyCharm [21], Python ile yazılım geliştirilmesi için JetBrains tarafından tasarlanmış tümleşik bir geliştirme ortamıdır. Yazılan kodun analizi ve grafiksel hata ayıklayıcı yeteneklerini barındırır. Tez kapsamında oluşturulan yazılım Pycharm kullanılarak kodlanmıştır.

5.1.3. Python'da veri tipleri ve değişken tanımlama

Değişken herhangi bir değeri saklamak için ayrılmış bir hafıza alanıdır. Başka bir deyişle, Python programındaki bir değişken, bilgisayara işleme için veri verir. Python'daki her değişkenin bir veri türü vardır. Karakter, sayı, liste, sözlük, matris (karmaşık veri tipi) Python'daki veri türlerinden bazılarıdır. Değişkenleri tanımlarken veri tipi verilmek zorunda değildir. Python, değişkenin ilk atamasında veri tipini belirler. Python'da değişkenin veri tipi dinamik olarak değiştirilebilir.

Python'da değişken tipi tanımlarken aşağıdaki ifadelere dikkat edilmelidir.

1. Değişken ismi birden fazla kelimeden oluşuyorsa aralarında boşluk bırakılmaz.
2. Değişken içinde sadece_ sembolü kullanılır.
3. Değişken ismi sayı ile başlayamaz.
4. Python'da tanımlı kelimeler değişken ismi olarak tanımlanamaz.

Karakter ifadeleri (stringler)

Python'da çift tırnak veya tek tırnak içine yazılan ifadeler karater ifadelerdir. Örneğin;
str_var1 = "Karakter Verisi"
str_var2 = 'Karakter Verisi'
şeklinde karakter veri tipindeki değişken tanımı yapılabilir.

Python'da sayısal veri tipleri (int, float)

Python'da sayı tanımlamak için iki veri tipi bulunur. Tamsayı için "int" veri tipi, ondalıklı sayı için "float" veri tipi kullanılır. Değişkenin türü ilk atanan değerin türü ile belirlenir.

Örnek olarak,

sayi1 = 0
değişkeni int (tamsayı) tipinde tanımlanırken,
sayi2 = 0.0
değişkeni float (ondalıklı) olarak tanımlanır.

Python'da liste veri tipi

Python'da liste veri tipi tanımlamak için [] operatörü kullanılır. Listelerde farklı veri tiplerinden elemanlar saklanabilir. Liste elemanlarına doğrudan erişilebilir ve doğrudan değer ataması yapılabilir.

Örnek olarak, sayılardan oluşan liste2 değişkeni aşağıdaki gibi ifade edilir.

liste2 = [100, 85, 30, 10, 6]
Listenin 0. elemanına ulaşmak için;
liste2[0] yazılarak 0. eleman değeri 100'e ulaşabilir.
Listenin 0. elemanına değer atanması için;
liste2[0] = 105 yazılarak listenin 0. elemana 105 değeri atanabilir.

5.1.4. Karşılaştırma operatörleri

Python'da kullanabilecek olan karşılaştırma operatörleri Çizelge 5.1'de verilmiştir.

Çizelge 5.1. Python karşılaştırma operatörleri

Operatör	Görevi
= =	İki değer birbirine eşitse True, değilse False döner.
! =	İki değer birbirine eşit değilse True, eşitse False döner.
>	Soldaki değer sağdaki değerden büyükse True, değilse False döner.
<	Soldaki değer sağdaki değerden küçükse True, değilse False döner.
> =	Soldaki değer sağdaki değerden büyükse veya eşitse True, değilse False döner.
<=	Soldaki değer sağdaki değerden küçükse veya eşitse True, değilse False döner.

5.1.5. Mantıksal bağlaçlar

Python'da iki veya daha fazla koşul ifadesini birbirine bağlamak için and veya or operatörü kullanılır.

And operatörü

Tüm karşılaştırma işlemlerinin kendi içinde sonucu True ise genel sonuç True, diğer durumlarda sonuç False çıkar.

Or operatörü

Tüm karşılaştırma işlemlerinden en az birinin kendi içinde sonucu True ise genel sonuç True, diğer durumlarda sonuç False çıkar.

Not operatörü

Mantıksal değeri tam tersi duruma çevirir. Not operatörü True olan sonucu False, False olan sonucu True sonucuna çevirir.

5.1.6. Kontrol deyimleri

Her programlama dilinde olduğu gibi Python'da da program akışını kontrol edebilmek için kontrol deyimleri bulunmaktadır. Python kontrol deyimlerinin nasıl kullanılacağı hakkında kısa bilgiler aşağıda verilmiştir.

If deyimi

"If" deyimi belli bir şartın kontrol edilmesi için kullanılır. Örnek olarak eğer dayanım 30'dan büyükse ekrana "basınç dayanımı yeterli" yazılması için If deyimi kullanılmalıdır.

If deyiminin söz dizimi aşağıdaki gibidir:

```
If şartlar:
        Yapılacaklar
```

Bu ifadeden anlaşılacağı üzere, eğer şartlar yerine gelirse yapılacaklar satırına yazılanlar yapılır.

Örnek olarak;

```
a= 32
If a > 30:
        Print (“basınç dayanımı yeterli”)
```

Elif ve else deyimi

Elif deyimi ile kod satırında If ifadesinde gerçekleşmemiş bir durum yeniden ele alınır. Else deyimi ile if ifadesi yada if-elif ifadesi gerçekleşmediği durumlarda çalışan deyimlerdir.

Örnek olarak sayı 90-100 arasında ise A, 80-89 arasında ise B diğer durumlarda C şeklindeki şartlı durum için if, elif ve else ifadesi kullanılır.

```
a = 70
if a => 90 and a <= 100:
        print(“A”)
elif a => 80 and a <= 89:
        print(“B”)
else:
        print(“C”)
```

5.1.7. Döngüler

Programlarda belirli koşullarda işlemlerin sürekli tekrar etmesi döngülerle sağlanır.

For döngüsü

For döngüsü, stringler, listeler, demetler veya sözlükler gibi veri tiplerinin üzerinde dolaşmamızı sağlayan döngü türüdür. For döngüsünün yapısı aşağıda belirtilmiştir.

```
for i in veri_turu:
        Yapılacak işlemler
```

Örnek olarak liste2 listesindeki tüm elemanlar aşağıdaki gibi for döngüsü kullanılarak ekrana yazdırılabilir.

```
liste2 = [100, 85, 30, 10, 6]

for i in liste2:
        print(i)
```

Burada i değişkeni her döngüde liste2'nin bir elemanı olur. For döngüsü aşağıdaki gibi "range" kullanılarak belirli bir aralık üzerinde yapılabilir. Burada dikkat edilmesi gereken husus döngünün ilk sayıdan başlayıp, son sayının bir eksiği ile bitişidir. Yani aşağıdaki örnekte program ekrana en son 9 yazıp çıkar.

```
for i in range(1,10):
        print (i)
```

<u>While döngüsü</u>

While döngüsünde şart kısmındaki durum doğru oldukça yapılacak işlemler yapılmaya devam eder. Şart kısmındaki durum False olduğunda döngü sonlanır. Bununla birlikte döngüden şartın doğruluğu sonlanmadan çıkmak için break anahtar kelimesi kullanılır. While döngüsünün yapısı aşağıda belirtilmiştir.

```
while şart:
        Yapılacak işlemler
```

Örnek olarak aşağıdaki döngü a değişkenini azaltarak ekrana yazar. Değişken 0'dan büyük oldukça bu işlem yapılır. Dikkat edilmesi gereken kısım, değişkenin döngü içerisinde sürekli azaltılmasıdır. Eğer bu işlem yapılmazsa sonsuz döngü oluşur.

```
a = 10
while a >= 0:
```

```
    print(a)
    a = a - 1
```

5.1.8. Fonksiyonlar

Fonksiyonlar, bir kere tanımlandıktan sonra çağrıldıklarında çalışan kod parçalarıdır. Aynı işi birden fazla defa yapmak için kodlanırlar. Fonksiyonlar def anahtar kelimesi ile tanımlanırlar. İsimleri ve varsa parametreleri yazılarak çağrılırlar. Fonksiyonlara parametre gönderilebilir ve fonksiyonlar sonuç olarak bir veri dönebilirler. Fonksiyonun veri döndürülmesi istenirse fonksiyon kodunun sonunda return anahtar kelimesi istenen değişkenden önce yazılarak değişken değeri fonksiyonu çağırana aktarılır.

Fonksiyonun yapısı aşağıda belirtilmiştir.

```
def fonksiyon_adı (paremetre1,  paremetre2, …..):
    Yapılacak işlemler
```

Sayının karesini hesaplayan örnek fonksiyon aşağıdaki gibi yazılıp çağrılabilir. Bu fonksiyon hem parametre alır hem de çağırana değer döndürür. Print fonksiyonu kare_al fonksiyonundan geri dönen değeri ekrana yazar.

```
def kare_al(x):
    return x*x
print(kare_al(4))
```

Python'un hata yakalama mekanizması ile programda oluşan beklenmedik hatalar yakalanarak yazılımın doğru çalışması sağlanır. Örnek olarak, sayı girilmesi gereken bir alana kullanıcı sayı yerine karakter girebilir. Bu durum beklenmedik olduğundan yazılım hata ile kapanır. Hata yakalama mekanizması ile beklenmedik hata yakalanıp kullanıcıya "Sadece Sayı Girmelisiniz" mesajı verilebilir. Python'da hata yakalamak içine try ve except anahtar kelimeler kullanılır. Hata yakalama örneği aşağıda gösterilmiştir. print (e) satırında Python'dan gelen hata mesajı ekrana yazdırılmaktadır.

```
sayi = 3
karakter = "hosgeldiniz"
```

```
try:
        sayi = karakter

except Exception as e:
        print(“Hatali atama yaptiniz.”)
        print (e)
```

5.1.9. Sınıflar ve nesne yönelimli programlama

Python dili nesne yönelimli programlamayı ve sınıfları destekler. Sınıflar nesneleri oluştururken nesnelerin özelliklerinin ve metotlarının tanımlandığı bir yapıdır. Nesneler sınıflardan örnekler alınarak oluşturulur. Nesne yönelimli yaklaşım sınıfların yazılım kodları içerisinde kullanılmasıdır. Python’da yeni bir sınıf tanımı class anahtar kelimesi ile yapılır. Örneğin; sayı türünde veri tutacak bir özelliği olan ve ekrana “Merhaba” yazacak bir metodu bulunan sınıf Python’da aşağıdaki gibi yazılır.

```
class Ornek:
        sayi = 0
        def ornek_methot(self):
                print(“Merhaba”)
```

Sınıf tanımları doğrudan çalıştırılabilir kodlar değillerdir. Bu tanımlardan örnek çıkarılarak kodların gereken kısımlarında çıkarılan örnekler üzerinden metotlar çağrılmalıdır. Ornek sınıfındaki ornek_method’u kullanabilmek için aşağıdaki kod satırları kullanılmalıdır.

```
sinif_ornek_nesne = Ornek()
sinif_ornek_nesne.ornek_method()
```

5.1.10. Modüller

Python’un açık kaynak kodlu olması ona topluluk tarafından geliştirilmiş birçok kütüphane kazandırmıştır. Çalışma kapsamında matris işlemleri için Numpy, veri görselleştirme için Matplotlib Kütüphanelerinden yararlanılmıştır.

Numpy Kütüphanesi

Numpy.array

Bu fonksiyon matris oluşturmak için kullanılır. Bir boyutlu ve 3 elemanlı matris aşağıdaki gibi oluşturulabilir.

```
mat = numpy.array([1, 2, 3])
```

Numpy ile bir yüzeyi ifade etmek için aşağıdaki iki boyutlu matris tanımı kullanılır.

```
mat2d = numpy.array([[1, 2, 3], [3, 4, 6]])
```

Matris boyutları bir değişkene bağlı dinamik olarak verilebilir. Örnek olarak x ve y değişkenlerinin değerlerine göre 2 boyutlu matris aşağıdaki gibi oluşturulabilir. Bu tanımda x,y ikilisi ayrıca parantez içinde yazılmalıdır.

```
x = 5
y = 3
mat2d = numpy.array((x, y))
```

Numpy matrisine veri tipi belirtilebilir. Aşağıdaki örnekte veri tipi float olan matris tanımı gösterilmiştir.

```
x = 5
y = 3
mat2d = numpy.array((x, y), dtype= ‘float ‘)
```

Numpy.equal

Matrisleri karşılaştırmak için kullanılan fonksiyondur. Karşılaştırılan iki matris aynı ise doğru (True), aynı değilse yanlış (False) değeri döndürür. Karşılaştırması yapılan matrislerin boyutlarının aynı olması gerekir. Bu fonksiyonun örnek kullanımı aşağıda gösterilmiştir. Matrisler birbirine eşit ise “esit_mi” değişkeninin değeri “True” olur, yoksa “False” olur.

```
x = 5
y = 3
mat1_2d = numpy.array((x, y), dtype= ‘float ‘)
mat2_2d = numpy.array((x, y), dtype= ‘float ‘)
esit_mi = numpy.equal(mat1_2d, mat2_2d)
```

Numpy.zeros

Yeni matris oluştururken tüm elemanları 0 olan matris oluşturmak için kullanılır. Fonksyonda veri tipi ayrıca belirtilebilir. Örnek kullanımı aşağıda gösterilmiştir.

```
x = 5
y = 3
mat_2d_zero = numpy.zeros((x, y), dtype= ‘float ‘)
```

Numpy.ones

Yeni matris oluştururken tüm elemanları 1 olan matris oluşturmak için kullanılır. Fonksiyonda veri tipi ayrıca belirtilebilir. Örnek kullanımı aşağıda gösterilmiştir.

```
x = 5
y = 3
mat_2d_one = numpy.ones((x, y), dtype= ‘float ‘)
```

Numpy.full

Oluşturulmuş bir matrisin tüm elemanlarına belirtilen değerin atanmasını sağlar. Örnek kullanımı aşağıda gösterilmiştir. “mat_2d” matrisinin tüm elemanları 128 değerine sahip olur.

```
x = 5
y = 3
mat_2d = numpy.full((x, y), 128, dtype= ‘float’)
```

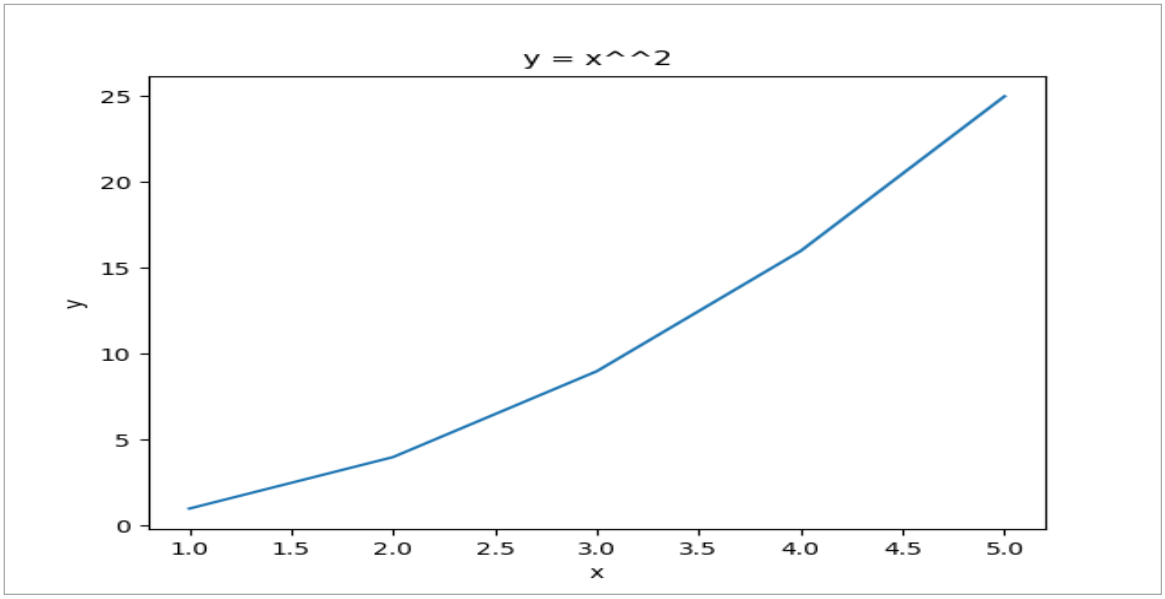
Matplotlib Kütüphanesi

Matplotlib kütüphanesi Python ortamında veri görselleştirmek için yaygın olarak kullanılan bir kütüphanedir. Az sayıda işlem adımıyla grafik oluşturmayı sağlar. Giriş olarak aşağıdaki örnekte $y=x^2$ grafiğinin çizilmesi gösterilmiştir.

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [1, 4, 9, 16, 25]
plt.title('y = x^^2')
plt.xlabel("x")
plt.ylabel("y")
plt.plot(x, y)
plt.show()
```

x'e bağlı bir y fonksiyonunun grafiğini oluşturmak için x ve y ekseni değerlerine ihtiyaç vardır. Yukarıdaki örnekte x ve y değerleri liste veri tipinde tanımlanmıştır. plt.plot (x, y) fonksiyonu çizgi (line) grafiği oluşturur. Fonksiyon birincisi x ve ikincisi y olmak üzere en az iki parametre ile çalışır. Şekil 5.1 bu kodların çalışması ile elde edilen grafiği gösterir.



Şekil 5.1. Matplotlib'de çizilen örnek grafik

Çizilen grafiğin türüne göre bu eksenleri oluşturan veri tipi değişmektedir. Örneğin çizgi grafiği için eksenlerde kullanılacak değerler liste olmalıyken, kontur (contour) grafiği için matris olmalıdır. Kontur grafikte x ve y eksenleri için oluşturulan matrislerin z eksenindeki matris boyutları ile uyumlu olması gereklidir. Aksi halde grafik çizimi başarılamaz. Aşağıdaki bir kontur grafiği örneği gösterilmiştir.

```
import numpy as np
import matplotlib.pyplot as plt
x = 5
y = 3
x_ar, y_ar = np.meshgrid(np.arange(0, x), np.arange(0, y))
z_ar = np.arange(15).reshape(3,5)
plt.contourf(x_ar, y_ar, z_ar)
plt.show()
```

Kullanılan fonksiyonların açıklamaları şöyle ifade edilebilir.

np.arange(0, x): Verilen sayılar arasında numpy.array oluşturur.

np.meshgrid(np.arange(0, x), np.arange(0, y)): Verilen listeleri mesh yapıp matris veri türü olarak geriye döner. Kaç adet liste varsa o kadar matris oluşturur.

np.arange(15).reshape(3,5): Bu kod parçasında iki adet fonksiyon peş peşe çağrılmıştır. Birinci fonksiyon üstte açıklandığı gibi, istenen sayıya kadar liste verisi oluşturur. Arkasından çağrılan reshape fonksiyonu ise oluşan listeyi 5x3 formatında matrise dönüştürür.

Kontur grafiği 3 eksenden oluşan matrisleri kullanarak çizim yaptığından oluşturulan matrisler sırasıyla x, y, z olarak çizim fonksiyonuna gönderilir. plt.show() fonksiyonu çizilen grafiğin ekrana çıkmasını sağlar.

5.2. Uygulamanın Geliştirilmesi

Uygulamada sıcaklık hesabının yapıldığı sıcaklık modülü, yer değiştirme hesabının yapıldığı yer değiştirme modülü, gerilme hesabının yapıldığı gerilme modülü ve ara yüzün tasarlandığı ara yüz modülü olmak üzere dört tane modül bulunmaktadır.

5.2.1. Sıcaklık modülü

Python kütüphanelerinden Numpy, Matplotlib ve Datetime "sicaklik" modülünde kullanılmıştır. Bu kütüphaneleri sicaklik modülünde kullanabilmek için import anahtar sözcüğü ile içeri aktarılmıştır. Matris hesaplamaları Numpy kütüphanesi kullanılarak yapılmıştır. Hesaplanan sıcaklık değerlerinin görselleştirilmesinde Matplotlib kütüphanesi kullanılmış ve kontur grafikleri oluşturulmuştur. Sıcaklık hesabının ne kadar sürede yapıldığını kayıt altına almak için Datetime Kütüphanesi kullanılmıştır. Yukarıda bahsedilen üç kütüphane aynı amaçlar için yer değiştirme ve gerilme modüllerinde de kullanılmıştır.

"SicaklikAnalizi" sınıfında sınıf değişkenlerinin veri tipleri belirlenmiştir. Dikdörtgen levha yüzeyi SicaklikAnalizi sınıfında sicaklik_mat olarak tanımlanmıştır. Sınıf değişkenlerinin kullanıcıdan aldığı verilerin ilk ataması __init__özel fonksiyonu ile yapılmıştır. Grafik çizme grafik_ciz, sıcaklık hesabı sicaklik_hesapla sınıf fonksiyonuyla gerçekleştirilmiştir.

Sıcaklık matrisinin elemanları çözüm ağındaki düğüm noktalarının sıcaklık değerlerini temsil etmektedir. Başlangıçta sıcaklık matrisi elemanlarına kullanıcıdan alınan ilk ortam sıcaklığı değeri atanmıştır. Levhanın sıcaklık uygulanan kenarlarına karşılık gelen matrisin ilgili elemanlarına sınır değer atamaları yapılmıştır. Levhanın izole kenara sahip olması durumunda matrisin kenar elemanları için sıcaklık sınır şartları tanımlanmıştır. Hesaplanmak istenen matris elemanı sonlu farklar denkleminde bilinen sıcaklık değerleri kullanılarak hesaplanmıştır. While döngüsü kurularak matris değerleri birbirine eşit oluncaya kadar matris elemanlarının sıcaklık değerleri hesaplanmıştır. Elde edilen sonuçlar görsel olarak ifade edildikten sonra yer değiştirme ve gerilme modüllerinde kullanılmak üzere program içerisinde değişkenlere atanmıştır.

Levha kenarlarının izole olması durumunda sıcaklık değerleri hesaplanırken sınır koşulları kullanılmıştır. Sol kenar ve alt kenarda sınır şartları tanımlanırken negatif indeksler oluşmuştur. Numpy'de negatif indeks sınır dışı olarak kabul edilmez. Bu yüzden negatif indeksler ile hesaplama yapılacağı zaman önceden belirlenmiş pozitif indeksteki değerle işlem yapılır. Örneğin; levhanın sol kenarı üzerindeki (0,1) noktası için sınır şartı $T_{-1,1} = T_{1,1}$ olarak tanımlanmıştır. $T_{-1,1}$ negatif index olduğu için bu değer önceden belirlenmiş sınır şartı gereği $T_{1,1}$ dönüştürülmüştür. Bu durumu ifade eden kod parçası aşağıda gösterilmiştir.

```
if i - 1 < 0:
    n_sol = self.sicaklik_mat[i + 1, j]
else:
    n_sol = self.sicaklik_mat[i - 1, j]
```

Ara yüzde kullanıcının seçtiği sınır şartlarına göre sıcaklık matrisinin hesaplanan kısımları değişmektedir. Bu durumu ifade edebilmek için her kenar için bir değişken tanımlanmıştır. Kenarın izole olması durumunda değişken değeri 0 atanarak izole kenarların üzerindeki noktaların hesaplanması sağlanmıştır. Kenardan ısı uygulanması durumunda değişken değeri 1 atanmış ve ısı uygulanan kenar üzerindeki noktaların hesaplanmaması sağlanmıştır. Bu durumu ifade eden kod parçası aşağıda gösterilmiştir.

```
for i in range(self.k_sol, self.x - self.k_sag):
    for j in range(self.k_alt, self.y - self.k_ust):
```

5.2.2. Yer değiştirme modülü

"YerDegistirmeAnalizi" sınıfında sınıf değişkenlerinin veri tipleri belirlenmiştir. Dikdörtgen levha üzerindeki noktaların yer değiştirmesi "YerDegistirmeAnalizi" sınıfında u ve v matrisleri olarak tanımlanmıştır. Sınıf değişkenlerinin kullanıcıdan aldığı verilerin ilk ataması __init__özel fonksiyonu ile yapılmıştır. Grafik çizme grafik_ciz, yer değiştirme hesabı yer_degistirme_hesapla fonksiyonları ile gerçekleştirilmiştir. Çözüm ağındaki düğüm noktalarının yer değiştirme değerleri u ve v matrisinin elemanları ile temsil edilmektedir. Başlangıçta u ve v matrisi elemanlarına sıfır değeri atanmıştır.

Levhanın serbest veya yer değiştirmesi engellenmiş kenara sahip olması durumunda matrisin ilgili elemanları için yer değiştirme sınır şartları tanımlanmıştır. Hesaplanmak istenen matris elemanı, sonlu farklar denkleminde bilinen yer değiştirme değerlerini kullanarak hesaplanmıştır. While döngüsü kurularak matris elemanlarının yer değiştirme değerleri hesaplanmıştır. Elde edilen sonuçlar görsel olarak ifade edildikten sonra gerilme modülünde kullanılmak üzere program içerisinde değişkenlere atanmıştır.

5.2.3. Gerilme modülü

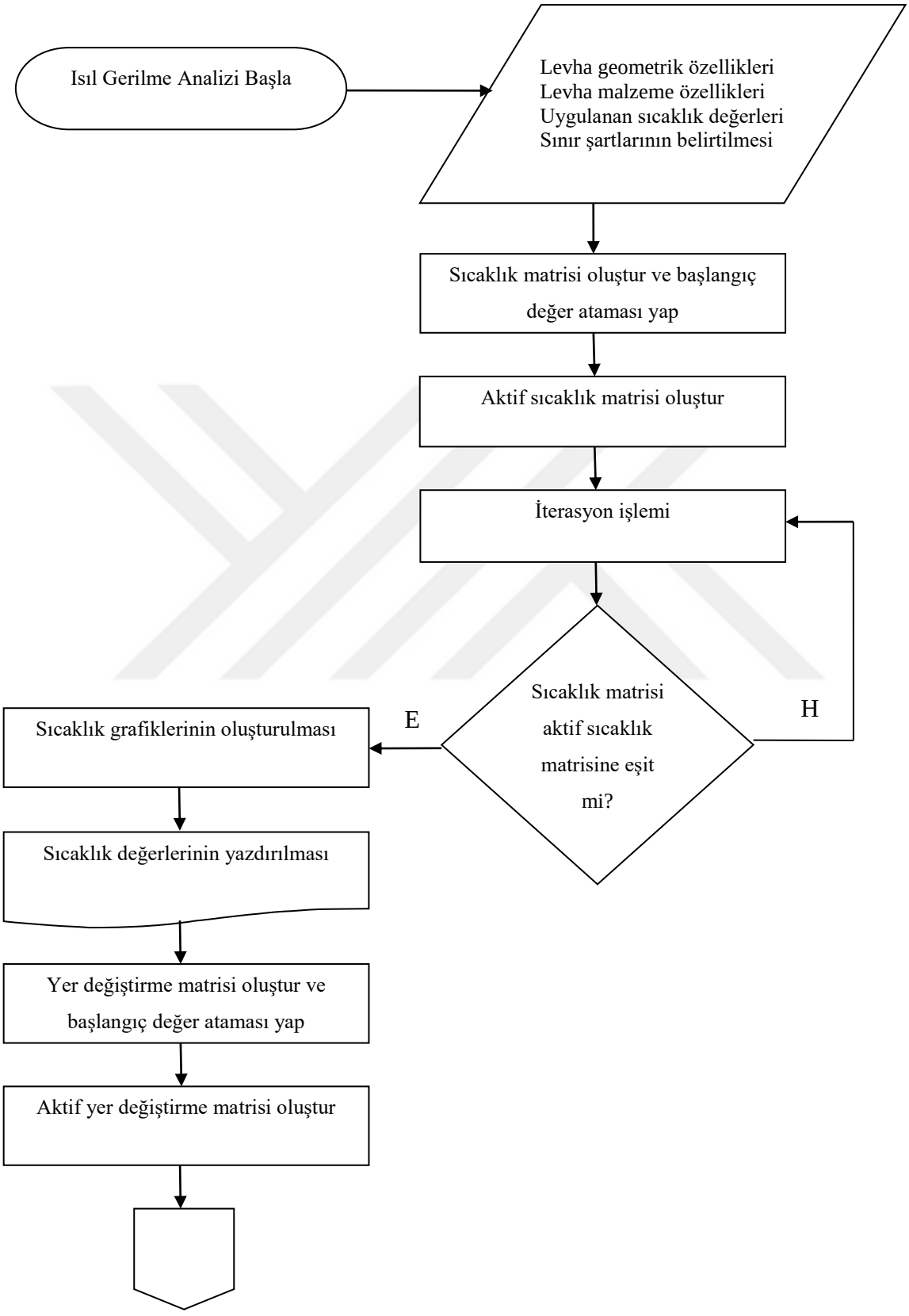
"GerilmeAnalizi" sınıfında sınıf değişkenlerinin veri tipleri belirlenmiştir. Gerilme matrisleri "GerilmeAnalizi" sınıfında xx, yy ve xy matrisi olarak tanımlanmıştır. Sınıf değişkenlerinin kullanıcıdan aldığı verilerin ilk ataması __init__özel fonksiyonu ile yapılmıştır. Grafik çizme grafik_ciz, gerilme hesabı xx_gerilme_hesapla, yy_gerilme_hesapla ve xy_gerilme_hesapla sınıf fonksiyonlarıyla gerçekleştirilmiştir. Çözüm ağındaki düğüm noktalarının gerilme değerlerini xx, yy ve xy matrisi temsil etmektedir. Başlangıçta xx, yy ve xy matris elemanlarına sıfır değeri atanmıştır. Levhanın serbest veya yer değiştirmesi engellenmiş kenara sahip olması durumunda matrisin kenar elemanları için yer değiştirme sınır şartları tanımlanmıştır. Hesaplanmak istenen matris elemanları sonlu farklar denkleminde bilinen yer değiştirme değerlerini kullanarak hesaplanmıştır. Elde edilen sonuçlar görsel olarak ifade edilmiştir.

5.3. Uygulamanın Kullanımı

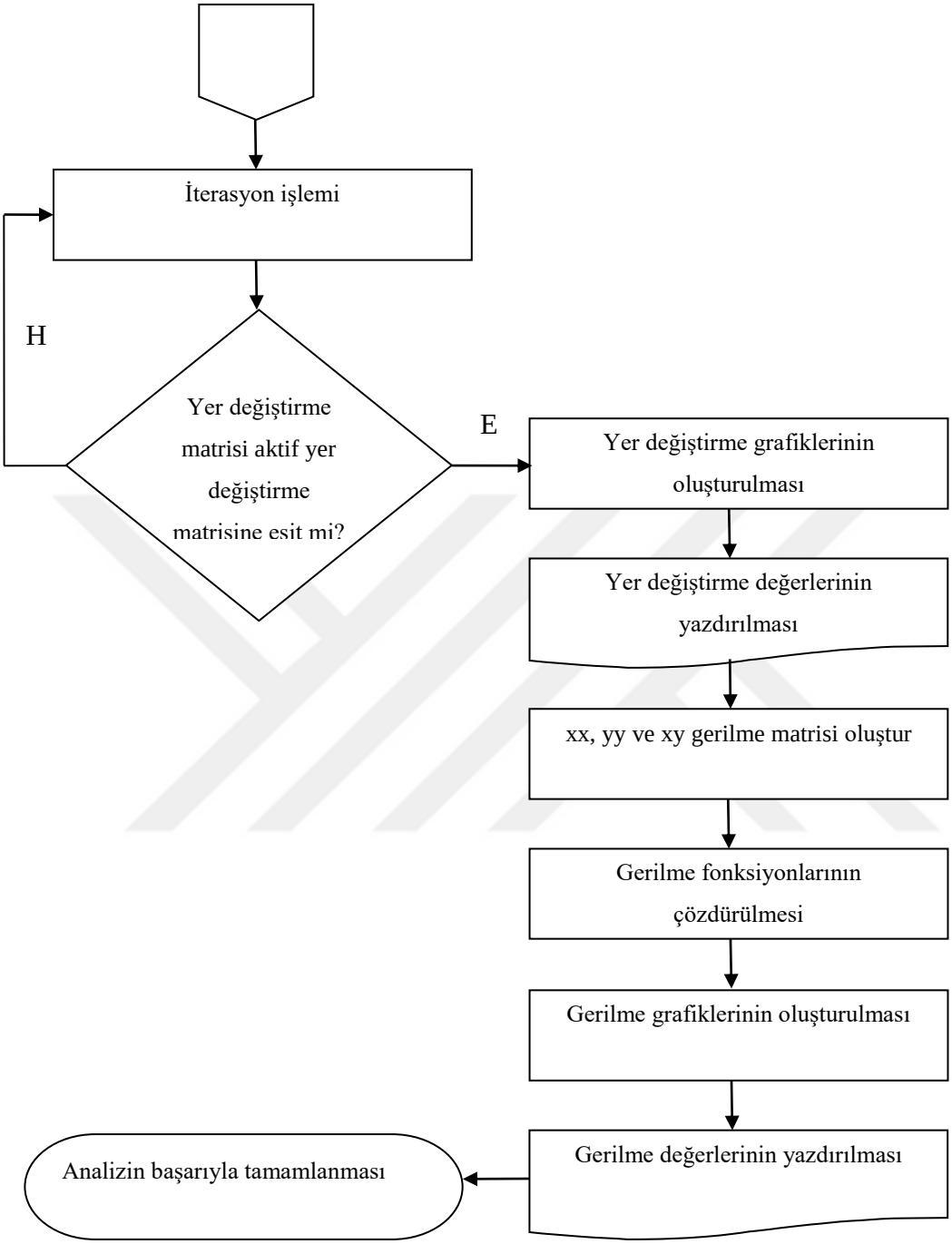
Isıl Gerilme Analizi uygulaması kullanıcıdan veri alan bir ara yüz ile çalışmaktadır. Kullanıcı analiz yapacağı verileri ara yüzde bulunun ilgili alanlardan girer ve seçimlerini işaretler. Bu işlemler tamamlandıktan sonra kullanıcının yanlış yapmasına mahal vermeden, ara yüz kullanıcıyı yönlendirir.

5.3.1. Akış diyagramı

Yazılım içindeki işlemlere ait iş akış diyagramı Şekil 5.2'de gösterilmiştir.



Şekil 5.2. IGAU'na ait iş akış diyagramı



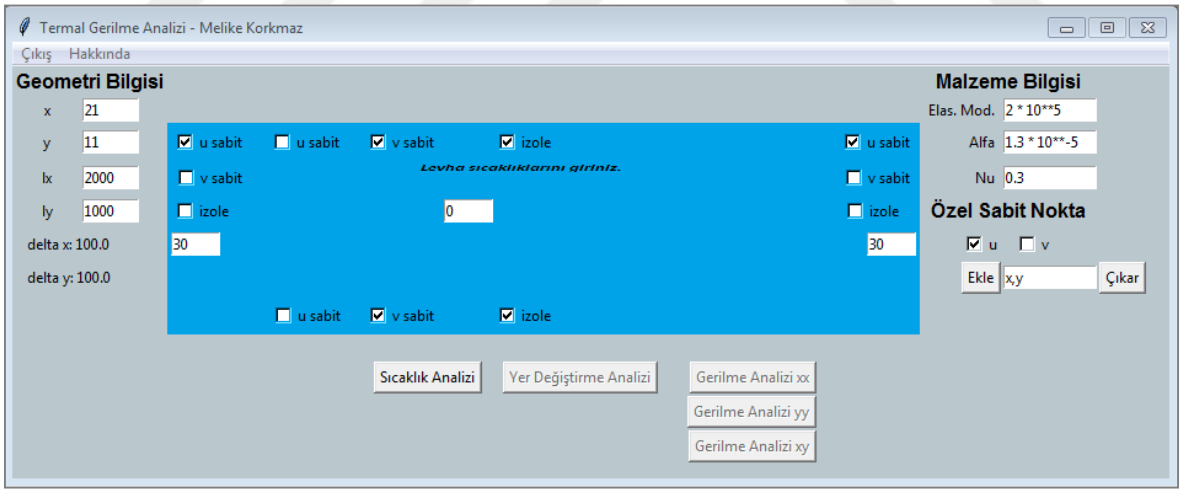
Şekil 5.2. (devam) IGAU'na ait iş akış diyagramı

5.3.2. Veri girişi

Isıl Gerilme Analizi uygulamasında analiz yapılabilmesi için kullanıcı tarafından bazı bilgilerin girilmesi gereklidir. Kullanıcıya ara yüzde bu bilgilerin girişi için metin kutuları ve işaret kutuları sağlanmıştır. Örneğin; çözüm ağı sayıları, levha uzunluğu, levha genişliği, uygulanan sıcaklıklar, ortam sıcaklığı ve malzeme bilgileri bilgi girilmesi

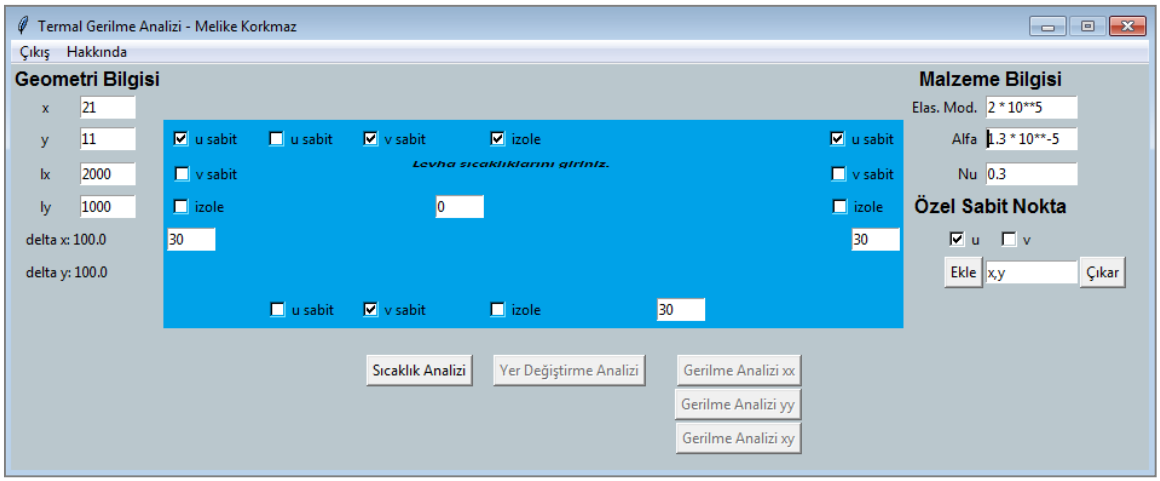
gereken metin kutularıdır. Sıcaklık ve yer değiştirme sınır şartları ise bilgi girilmesi gereken işaret kutularıdır.

Kullanıcıyı yönlendirmesi için bilgi girilmesi gereken alanlara başlangıçta bazı değerler yazılmıştır. Yazılan değerler Python1 örneğinin üniform sıcaklık dağılımı için ısıl gerilme analizini gerçekleştiren değerlerdir. Dikdörtgen levhanın geometri bilgisi kısmına çözüm ağı 21x11, uzunluğu 2000 mm ve genişliği 1000 mm olacak şekilde başlangıç değerleri yazılmıştır. Dikdörtgen levhanın malzeme bilgisi kısmına çelik türü malzemenin özellikleri başlangıç değeri olarak yazılmıştır. Levhanın başlangıç sıcaklığı 0°C, sol ve sağ kenardan uygulanan sıcaklık 30°C olacak şekilde başlangıç değerleri yazılmıştır. Levhanın düşey kenarların *x* yönünde yer değiştirmeleri engellendiği için u sabit işaret kutuları, yatay kenarların *y* yönünde yer değiştirmeleri engellendiği için v sabit işaret kutuları başlangıç sınır şartı olarak işaretlenmiştir. Levhanın alt ve üst kenarlarında ısı akışı olmadığı için izole işaret kutuları başlangıç sınır şartı olarak işaretlenmiştir. Şekil 5.3'te uygulama başlarken açılan açılış penceresi ekran görüntüsüne yer verilmiştir.



Şekil 5.3. IGAU açılış penceresi

Levhanın herhangi bir kenarı için ısı akışı engellenmişse yani kenar üzerinde izole işaret kutusu işaretlenmişse, uygulama bu kenar için var olan ısı giriş formunu kapatarak ısı değerinin girilmesine izin vermez. Örnek olarak üst kenarın izole seçilmesi durumunda uygulamanın nasıl davrandığı Şekil 5.4'te gösterilmiştir.

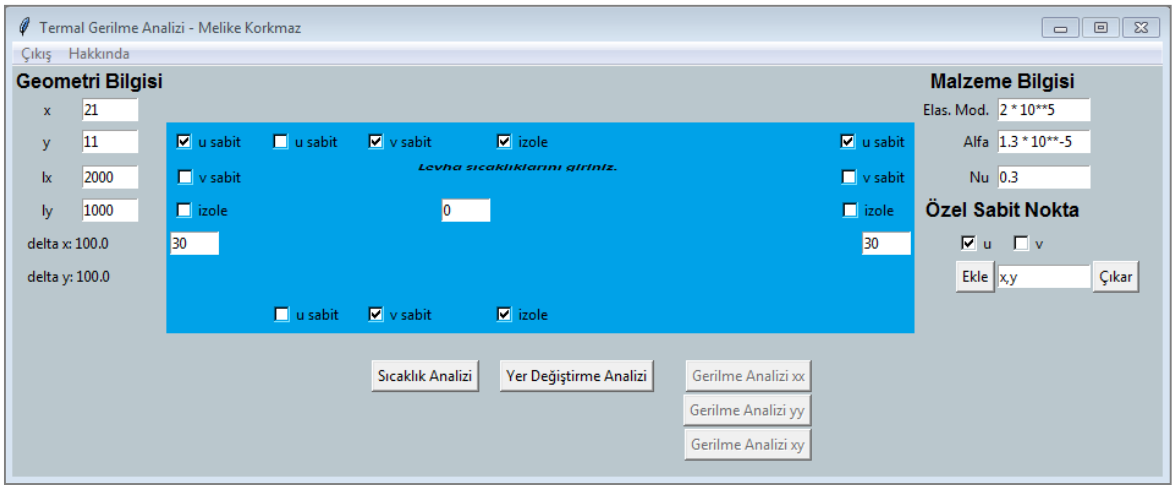


Şekil 5.4. IGAU izole kenar seçimi

Isıl Gerilme Analizi uygulaması çalıştığında ilk olarak geometri bilgileri, malzeme bilgileri ve sıcaklık değerleri metin kutularına girilmelidir. Daha sonra dikdörtgen levhanın sıcaklık ve yer değiştirme sınır şartları işaret kutuları kullanarak belirlenmelidir. Sıcaklık sınır şartı olarak levhanın kenarlarında ısı akışının serbest veya izole olması durumuna göre iki seçenek mevcuttur. Yer değiştirme sınır şartı olarak levhanın kenarlarında yer değiştirmenin olması veya engellenmesi durumuna göre iki seçenek mevcuttur.

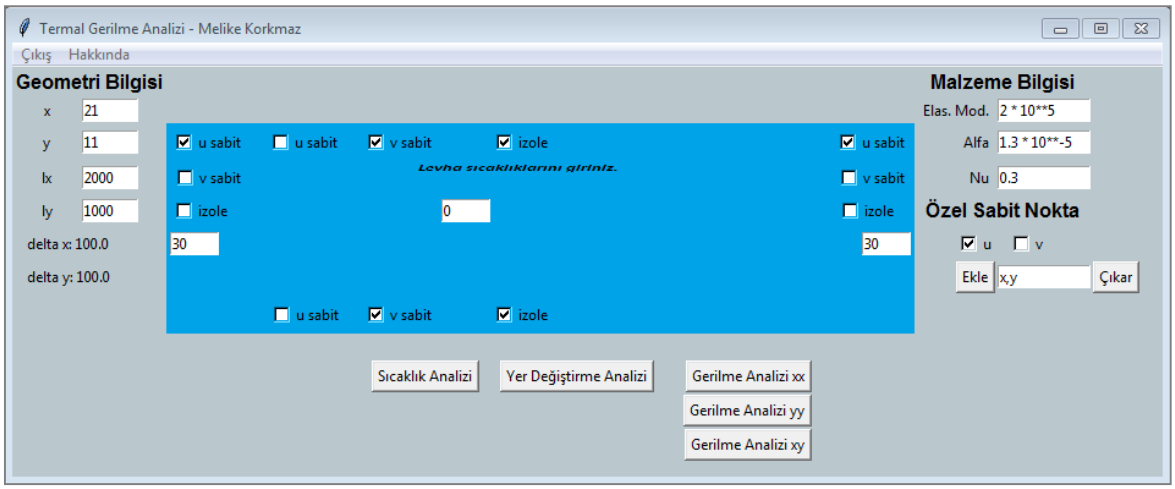
5.3.3. Analizlerin yapılması

Gerekli veri girişleri ve sınır şartı seçimleri kullanıcılar tarafından belirlendikten sonra ara yüzde sağlanan butonlar ile analiz başlatılır. Şekil 5.3'te gösterildiği üzere uygulama sıcaklık analizi butonu hariç diğer butonları pasif duruma getirerek kullanıcının diğer analizleri yapmasına izin vermez, böylelikle kullanıcıyı yönlendirmiş olur. Yazılımda sıcaklık analizi yapılmadan yer değiştirme ve gerilme analizi yapılamaz. Sıcaklık analizi tamamlandıktan sonra sıcaklık dağılımı grafiği oluşturulur ve sonuçlar PyCharm konsoluna Excel'e aktarılabilir olarak yazılır. Yazılım arka planda sıcaklık analizinden elde edilen sonuçları yer değiştirme analizi yapmak için kullanır. Bu sırada "Yer Değiştirme Analizi" butonu aktive edilir ve kullanıcı yer değiştirme analizini gerçekleştirebilir. Şekil 5.5'te sıcaklık analizini başarıyla gerçekleştirdiği gösterilmiştir. "Sıcaklık Analizi" ve "Yer Değiştirme Analizi" butonları aktif durumdadır.



Şekil 5.5. IGAU'nda sıcaklık analiz sonrası pencerenin durumu

Yer değiştirme analizinin yapılabilmesi için öncesinde sıcaklık analizinin yapılmış olması gereklidir. Yer değiştirme analizi tamamlandıktan sonra *x* ve *y* yönündeki yer değiştirmeleri gösteren kontur grafiği aynı ekranda oluşturulur, sonuçlar PyCharm ekranına Excel'e aktarılabilir olarak yazılır. Yazılım arka planda yer değiştirme analizinden elde edilen sonuçları gerilme analizi yapmak için kullanır. Bu sırada "Gerilme Analizi xx" "Gerilme Analizi yy" ve "Gerilme Analizi xy" butonları aktive edilir ve kullanıcı gerilme analizini gerçekleştirebilir. Şekil 5.6'da uygulamanın yer değiştirme analizini başarıyla gerçekleştirdiği gösterilmiştir. "Sıcaklık Analizi", "Yer Değiştirme Analizi" ve "Gerilme Analizi xx", "Gerilme Analizi yy" ve "Gerilme Analizi xy" butonları aktif durumdadır.



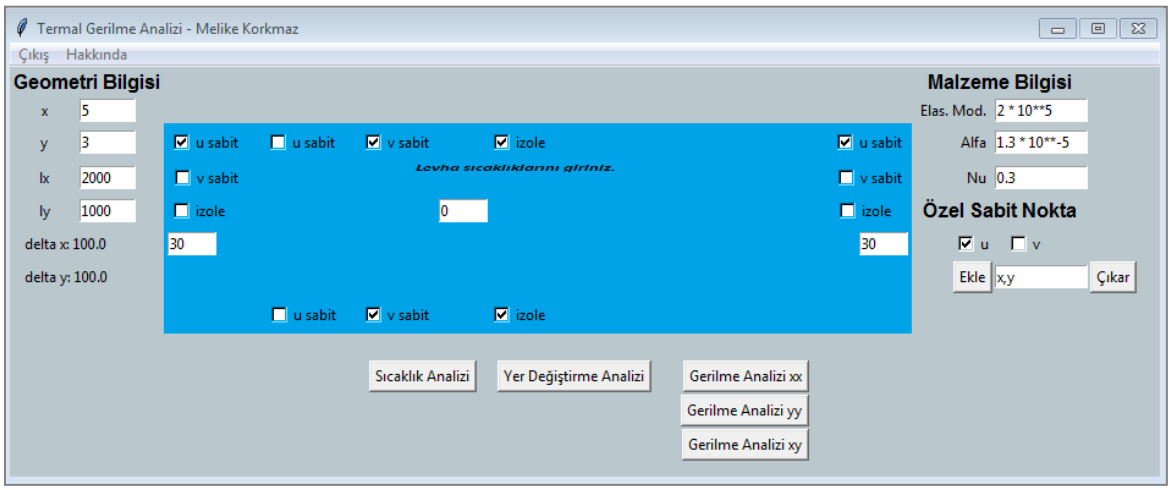
Şekil 5.6. IGAU yer değiştirme analizi sonrası pencerenin durumu

Gerilme analizinin yapılabilmesi için öncesinde sıcaklık ve yer değiştirme analizinin yapılmış olması gereklidir. Yazılımda gerilme analizini gerçekleştirebilmek için üç adet buton kullanılmıştır. Gerilme hesapları birbirine bağlı olmadığından, yer değiştirme hesabı yapılınca üç buton aynı anda aktif hale gelmektedir. Gerilme analizi tamamlandıktan sonra *x* doğrultusundaki normal gerilme, *y* doğrultusundaki normal gerilme ve *xy* düzlemindeki gerilmeleri gösteren kontur grafikleri ayrı ayrı oluşturulur, sonuçlar PyCharm konsol ekranında Excel'e aktarılabilir olarak yazılır.

5.3.4. Örnek analiz

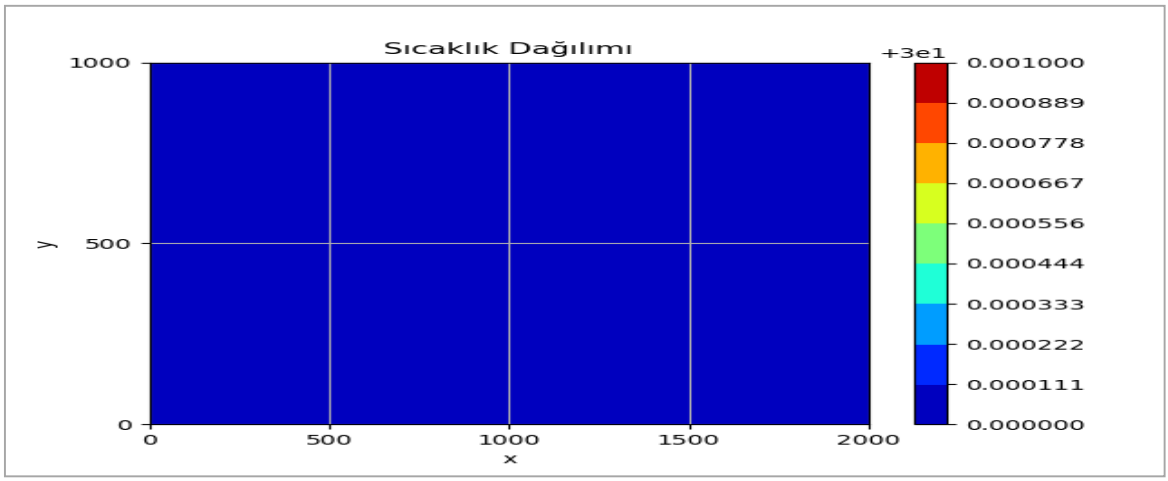
Bu bölümde Isıl Gerilme Analizi uygulaması kullanılarak, 5x3'lük küçük bir çözüm ağı üzerinden sıcaklık, yer değiştirme ve gerilme analizinin nasıl gerçekleştirildiği anlatılmıştır. Örnek analiz için küçük bir çözüm ağının tercih edilme sebebi konsol ekranında basılan sıcaklık, yer değiştirme ve gerilme sonuçlarının gösterilmek istenmesidir.

Isıl Gerilme Analizi uygulaması çalıştığında bilgilerin girişi için metin kutularına girilen değerler ve işaret kutularını kullanarak seçilen sınır şartları Şekil 5.7'de gösterilmiştir.



Şekil 5.7. Örnek analiz veri ve sınır şartı girişi

Gerekli veri girişleri ve sınır şartı seçimleri kullanıcılar tarafından belirlendikten sonra aktif olan "Sıcaklık Analizi" butonuna basılır. Sıcaklık analizi tamamlandıktan sonra ekrana Şekil 5.8'de görülen sıcaklık dağılımı grafiği çizilir.



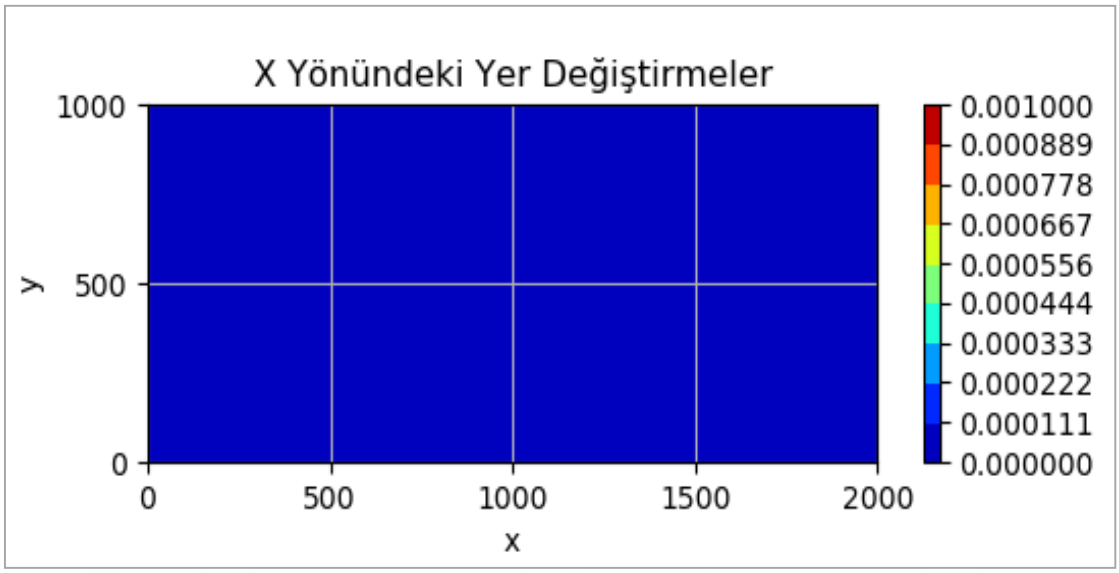
Şekil 5.8. Örnek analiz sıcaklık dağılımı grafiği

Şekil 5.9'da görülen PyCharm konsol ekranına iterasyonun kaç adımda gerçekleştiği, iterasyon süresi ve düğüm noktalarının sıcaklık değerleri basılır.

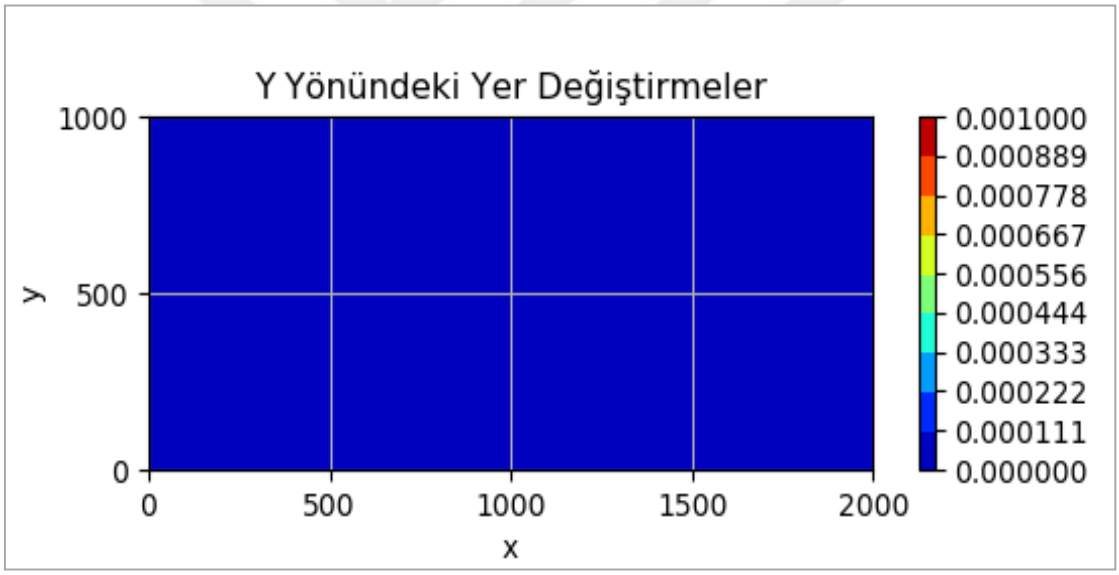
```
Sicaklik          ==>> X: 5, Y: 3 | 131    iterasyon | sure: 0:00:00.046800
Sicaklik
T(0,0)  30,0
T(0,1)  30,0
T(0,2)  30,0
T(1,0)  29,999999974905847
T(1,1)  29,999999974905847
T(1,2)  29,999999974905847
T(2,0)  29,99999996451151
T(2,1)  29,99999996451151
T(2,2)  29,99999996451151
T(3,0)  29,999999974905847
T(3,1)  29,999999974905847
T(3,2)  29,999999974905847
T(4,0)  30,0
T(4,1)  30,0
T(4,2)  30,0
```

Şekil 5.9. Örnek analiz sıcaklık değerlerini gösteren konsol ekran

Yazılım sıcaklık analizinden elde edilen sonuçları yer değiştirme analizi yapmak için ara yüze gönderir. Aktif olan "Yer Değiştirme Analizi" butonuna basılır. Yer değiştirme analizi tamamlandıktan sonra ekrana Şekil 5.10 ve Şekil 5.11'de görülen yer değiştirme dağılımı grafiği çizilir.



Şekil 5.10. Örnek analiz x yönündeki yer değiştirme dağılımı grafiği



Şekil 5.11. Örnek analiz y yönündeki yer değiştirme dağılımı grafiği

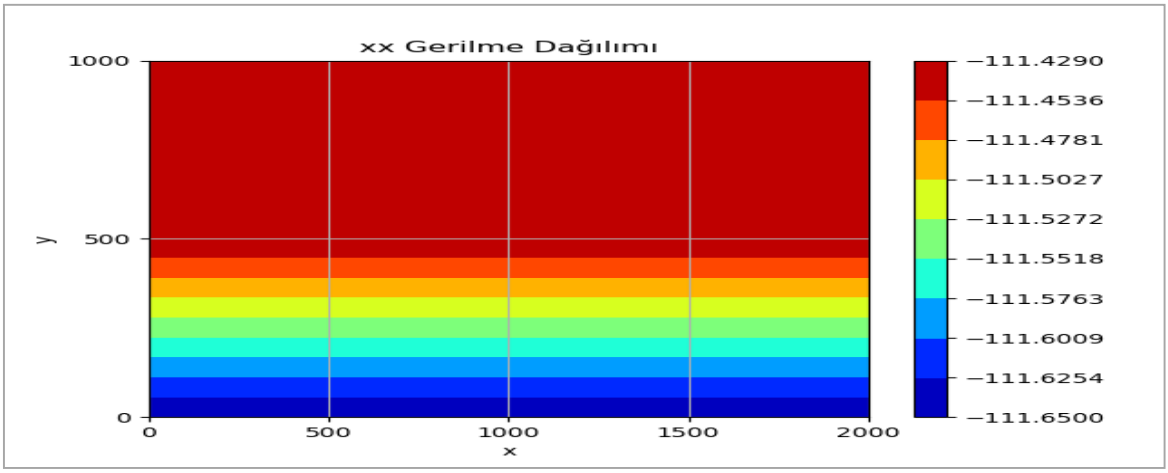
Şekil 5.12'de görülen PyCharm konsol ekranında iterasyonun kaç adımda gerçekleştiği, iterasyon süresi ve düğüm noktalarının yer değiştirme değerleri basılır.

```
Deplasman ==>>  X: 5, Y: 3 | sure: 0:00:00 | 13          iterasyon
u
u(0,0)  0,0
u(0,1)  0,0
u(0,2)  0,0
u(1,0)  7,496942899022074e-11
u(1,1)  7,496942899022074e-11
u(1,2)  7,496942899022074e-11
u(2,0)  0,0
u(2,1)  0,0
u(2,2)  0,0
u(3,0)  -7,496942899022074e-11
u(3,1)  -7,496942899022074e-11
u(3,2)  -7,496942899022074e-11
u(4,0)  0,0
u(4,1)  0,0
u(4,2)  0,0

v
v(0,0)  0,0
v(0,1)  0,0
v(0,2)  0,0
v(1,0)  0,0
v(1,1)  0,0
v(1,2)  0,0
v(2,0)  0,0
v(2,1)  0,0
v(2,2)  0,0
v(3,0)  0,0
v(3,1)  0,0
v(3,2)  0,0
v(4,0)  0,0
v(4,1)  0,0
```

Şekil 5.12. Örnek analiz yer değiştirme değerlerini gösteren konsol ekran

Yazılım yer değiştirme analizinden elde edilen sonuçları gerilme analizi yapmak için ara yüze gönderir. Aktif olan “Gerilme Analizi xx” butonuna basılır. Gerilme analizi tamamlandıktan sonra ekrana Şekil 5.13’te görülen x doğrultusundaki gerilme dağılımı grafiği çizilir. Aynı analizler “Gerilme Analizi yy” ve “Gerilme Analizi xy” için yapılır.

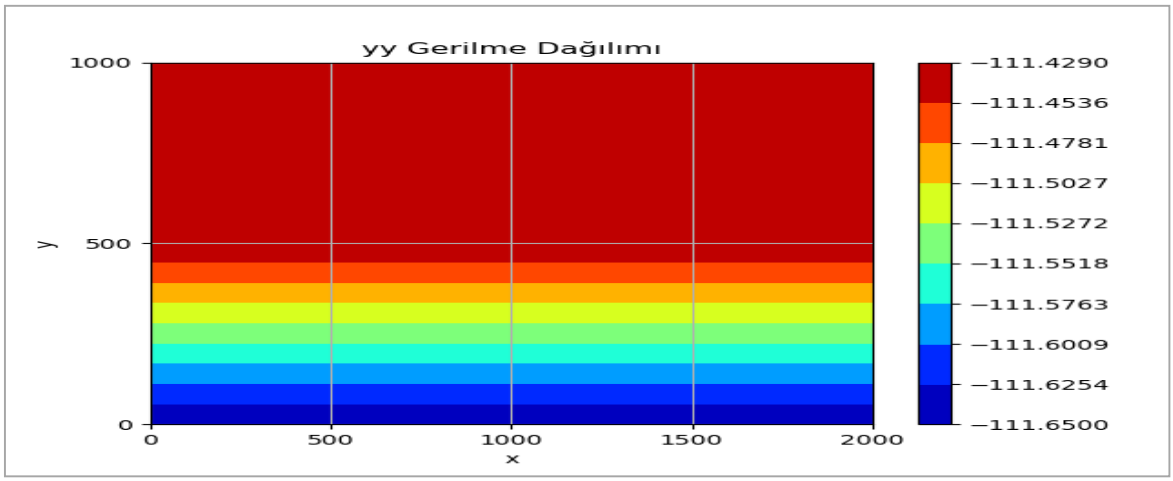


Şekil 5.13. Örnek analiz x doğrultusundaki normal gerilme dağılımı grafiği

Şekil 5.14'te görülen PyCharm konsol ekranında düğüm noktalarının *x* doğrultusundaki gerilme değerleri basılır.

```
gerilme xx
gxx(0,0)    -111,65142847804663
gxx(0,1)    -111,42857133537588
gxx(0,2)    -111,42812562109053
gxx(1,0)    -111,6514284780353
gxx(1,1)    -111,42857133536457
gxx(1,2)    -111,42812562107922
gxx(2,0)    -111,65142847237004
gxx(2,1)    -111,42857132971062
gxx(2,2)    -111,4281256154253
gxx(3,0)    -111,6514284780353
gxx(3,1)    -111,42857133536457
gxx(3,2)    -111,42812562107922
gxx(4,0)    -111,65142847804663
gxx(4,1)    -111,42857133537588
gxx(4,2)    -111,42812562109053
```

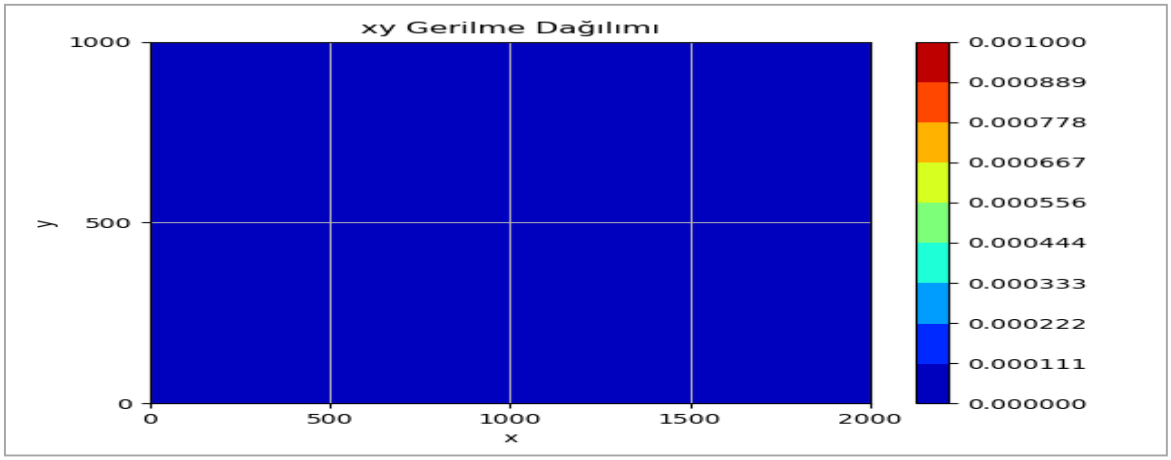
Şekil 5.14. Örnek analiz x doğrultusundaki normal gerilme değerlerini gösteren konsol ekran



Şekil 5.15. Örnek analiz y doğrultusundaki normal gerilme dağılımı grafiği

```
gerilme yy
gyy(0,0)     -111,65142847809285
gyy(0,1)     -111,428571335422
gyy(0,2)     -111,42812562113666
gyy(1,0)     -111,6514284780353
gyy(1,1)     -111,42857133536457
gyy(1,2)     -111,42812562107922
gyy(2,0)     -111,6514284492564
gyy(2,1)     -111,42857130664311
gyy(2,2)     -111,42812559235789
gyy(3,0)     -111,6514284780353
gyy(3,1)     -111,42857133536457
gyy(3,2)     -111,42812562107922
gyy(4,0)     -111,65142847809285
gyy(4,1)     -111,428571335422
gyy(4,2)     -111,42812562113666
```

Şekil 5.16. Örnek analiz y doğrultusundaki normal gerilme değerlerini gösteren konsol ekran



Şekil 5.17. Örnek analiz xy kesme gerilmesi dağılımı grafiği

```
gerilme xy
gxy(0,0)     0,0
gxy(0,1)     0,0
gxy(0,2)     0,0
gxy(1,0)     0,0
gxy(1,1)     0,0
gxy(1,2)     0,0
gxy(2,0)     0,0
gxy(2,1)     0,0
gxy(2,2)     0,0
gxy(3,0)     0,0
gxy(3,1)     0,0
gxy(3,2)     0,0
gxy(4,0)     0,0
gxy(4,1)     0,0
gxy(4,2)     0,0
```

Şekil 5.18. Örnek analiz xy kesme gerilmesi değerlerini gösteren konsol ekran

6. PYTHONDA GELİŞTİRİLEN YAZILIMDA TERMOELASTİK GERİME ANALİZİ

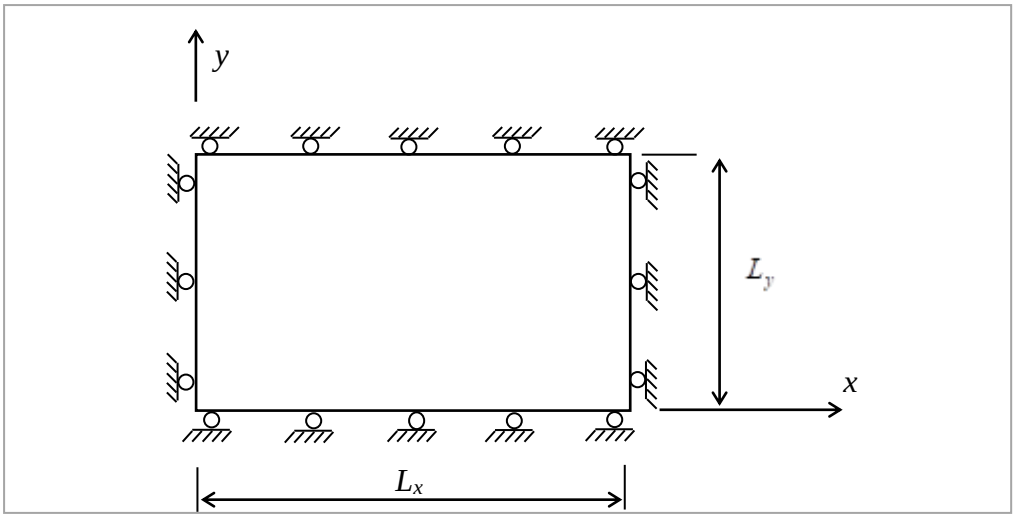
Sonlu farklar yöntemini kullanarak Python programlama dilinde oluşturulan yazılımdan elde edilen ısıl gerilme analizi sonuçları bu bölümde gösterilmiştir. Yazılımdan elde edilen sonuçların doğruluğunu göstermek için analitik çözüm ve ABAQUS sonlu elemanlar yazılımı kullanılmıştır.

Dikdörtgen bir levhanın farklı sınır şartları altında, kararlı hal ısı iletim durumunda ısıl gerilme analizlerinin sonuçları gösterilmiştir ve yorumlanmıştır. Her bir düğüm noktası için elde edilen sıcaklık, yer değiştirme ve gerilme değerlerinin daha kolay anlaşılabilir ve yorumlanabilir olması için sonuçlar grafiksel olarak da ifade edilmiştir.

Oluşturulan örnek modeller Python olarak adlandırılmıştır.

6.1. Python-1 Örneği

Bu örnekte düşey kenarların x yönünde, yatay kenarların y yönünde yer değiştirmelerinin engellendiği, yani sürtünmesiz bir duvar içine yerleştirilmiş, lineer elastik malzeme özelliklerine sahip dikdörtgen bir levha ele alınmıştır. L_x uzunluğundaki ve L_y genişliğindeki bu dikdörtgen levha Şekil 6.1'de gösterilmiştir.



Şekil 6.1. Dört kenarı kayıcı mafsal mesnetli levha

Dikdörtgen levhanın geometrik özellikleri Çizelge 6.1'de, malzeme özellikleri Çizelge 6.2.'de verilmiştir.

Çizelge 6.1. Dikdörtgen levhanın geometrik özellikleri

Dikdörtgen plağın uzunluğu	$L_x = 2000mm$
Dikdörtgen plağın genişliği	$L_y = 1000mm$

Çizelge 6.2. Dikdörtgen levhanın malzeme özellikleri

Elastisite modülü	$E = 2\times10^5 N/mm^2$
Poisson oranı	$\nu = 0.3$
Isıl genleşme katsayısı	$\alpha = 1.3\times10^{-5} C^{-1}$

Dikdörtgen levhanın sıcaklık sınır şartları Çizelge 6.3'te, yer değiştirme sınır şartları Çizelge 6.4.'te verilmiştir.

Çizelge 6.3. Dikdörtgen levhanın sıcaklık sınır şartı

$y = 0$	$\frac{\partial T}{\partial y} = 0$
$y = 1000mm$	$\frac{\partial T}{\partial y} = 0$

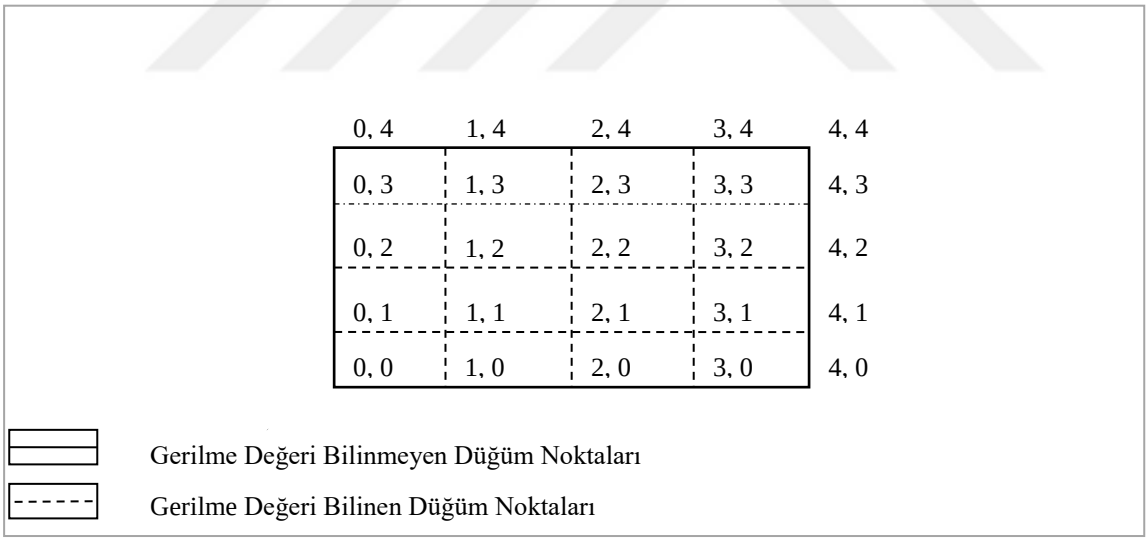
Çizelge 6.4. Python-1 için yer değiştirme sınır şartları

$x = 0$	$u = 0, \tau_{xy} = 0, \frac{\partial u}{\partial y} = 0, \frac{\partial v}{\partial x} = 0$
$x = 2000mm$	$u = 0, \tau_{xy} = 0, \frac{\partial u}{\partial y} = 0, \frac{\partial v}{\partial x} = 0$
$y = 0$	$v = 0, \tau_{xy} = 0, \frac{\partial v}{\partial x} = 0, \frac{\partial u}{\partial y} = 0$
$y = 1000mm$	$v = 0, \tau_{xy} = 0, \frac{\partial v}{\partial x} = 0, \frac{\partial u}{\partial y} = 0$

Çizelge 6.4'te ifade edilen sınır şartlarından $u = 0,\ v = 0,\ \tau_{xy} = 0$ doğrudan bilinen sınır şartlarıdır. $x = 0$ ve $x = 2000mm$ kenarı boyunca x yönündeki yerdeğiştirme sabit olduğu

için $\frac{\partial u}{\partial y}=0$, $y=0$ ve $y=1000mm$ kenarı boyunca y yönündeki yerdeğiştirme sabit olduğu için $\frac{\partial v}{\partial x}=0$ sınır şartı elde edilmiştir. $x=0$ ve $x=2000mm$ kenarı boyunca $\frac{\partial u}{\partial y}=0$, $\tau_{xy}=0$ olduğu için $\frac{\partial v}{\partial x}=0$, $y=0$ ve $y=1000mm$ kenarı boyunca $\frac{\partial v}{\partial x}=0$, $\tau_{xy}=0$ olduğu için $\frac{\partial u}{\partial y}=0$ sınır şartı elde edilmiştir.

Çizelge 6.4'te ifade edilen sınır şartlarını kullanarak levhadaki tüm düğüm noktalarının x ve y yönündeki yer değiştirme değerleri hesaplanmıştır. Levhanın iç kısımlarındaki normal gerilme değerleri de hesaplanan yer değiştirmeler kullanılarak elde edilmiştir. Levha kenarları üzerindeki normal gerilmeler ise hesaplanan iç kısımlardaki normal gerilme değerleri kullanılarak elde edilmiştir. Şekil 6.2'de levha üzerindeki normal gerilme değeri bilinen ve bilinmeyen düğüm noktaları gösterilmiştir.



Şekil 6.2. Gerilme değeri bilinen ve bilinmeyen düğüm noktaları

Köşe noktalar hariç sol kenar üzerindeki düğüm noktalarının normal gerilme hesabı Eş. 6.1 ve Eş. 6.2,

$$(\sigma_x)_{0,1}=(\sigma_x)_{1,1}+((\sigma_x)_{1,1}-(\sigma_x)_{2,1}) \tag{6.1}$$

$$(\sigma_y)_{0,1} = (\sigma_y)_{1,1} + ((\sigma_y)_{1,1} - (\sigma_y)_{2,1}) \quad (6.2)$$

köşe noktalar hariç sağ kenar üzerindeki düğüm noktalarının normal gerilme hesabı Eş. 6.3 ve Eş. 6.4,

$$(\sigma_x)_{4,1} = (\sigma_x)_{3,1} + ((\sigma_x)_{3,1} - (\sigma_x)_{2,1}) \quad (6.3)$$

$$(\sigma_y)_{4,1} = (\sigma_y)_{3,1} + ((\sigma_y)_{3,1} - (\sigma_y)_{2,1}) \quad (6.4)$$

köşe noktalar hariç alt kenar üzerindeki düğüm noktalarının normal gerilme hesabı Eş. 6.5 ve Eş. 6.6,

$$(\sigma_x)_{1,0} = (\sigma_x)_{1,1} + ((\sigma_x)_{1,1} - (\sigma_x)_{1,2}) \quad (6.5)$$

$$(\sigma_y)_{1,0} = (\sigma_y)_{1,1} + ((\sigma_y)_{1,1} - (\sigma_y)_{1,2}) \quad (6.6)$$

köşe noktalar hariç alt kenar üzerindeki düğüm noktalarının normal gerilme hesabı Eş. 6.7 ve Eş. 6.8,

$$(\sigma_x)_{1,4} = (\sigma_x)_{1,3} + ((\sigma_x)_{1,3} - (\sigma_x)_{1,2}) \quad (6.7)$$

$$(\sigma_x)_{1,4} = (\sigma_x)_{1,3} + ((\sigma_x)_{1,3} - (\sigma_x)_{1,2}) \quad (6.8)$$

kullanılarak hesaplanmıştır.

Köşe noktalar ise kenarlar üzerindeki düğüm noktalarının normal gerilme değerleri kullanılarak hesaplanmıştır. Eş. 6.9, Eş. 6.10, Eş. 6.11 ve Eş. 6.12'de köşe noktalarının nasıl hesaplandığı anlatılmıştır.

$$(\sigma_x)_{0,0} = (\sigma_x)_{0,1} \quad (6.9)$$

$$(\sigma_x)_{0,4} = (\sigma_x)_{0,3} \quad (6.10)$$

$$(\sigma_x)_{4,0} = (\sigma_x)_{4,1} \tag{6.11}$$

$$(\sigma_x)_{4,4} = (\sigma_x)_{4,3} \tag{6.12}$$

6.1.1. Üniform sıcaklık dağılımı durumu

Python-1 örneğinde üniform sıcaklık dağılımı elde etmek için levhanın alt ve üst kenarları izole kenar seçilmiş, sol ve sağ kenardan 30℃ sıcaklık uygulanmıştır. Ortamın ilk sıcaklığı 0℃'dir. $\Delta T = 30℃$ sıcaklık artışı oluşturulmuştur.

Çizelge 6.5. Python-1 ÜSD için sıcaklık sınır şartları

Ortam sıcaklığı	$T_{ortam} = 0\,℃$
Sol kenardan uygulanan sıcaklık	T_{sol}=30℃
Sağ kenardan uygulanan sıcaklık	$T_{sağ}$=30℃

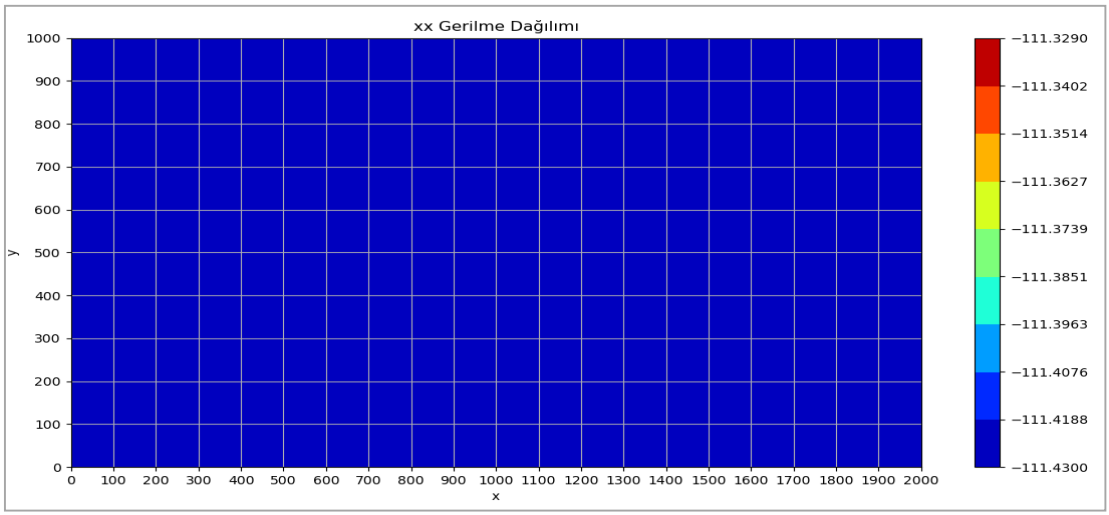
ΔT üniform sıcaklık artışı durumunda x doğrultusundaki normal gerilmelerin analitik ifadesi Eş.6.13'te, y doğrultusundaki normal gerilmelerin analitik ifadesi Eş.6.14'te verilmiştir [22].

$$\sigma_x = \frac{1+\nu}{1-\nu^2} E\alpha\Delta T \tag{6.13}$$

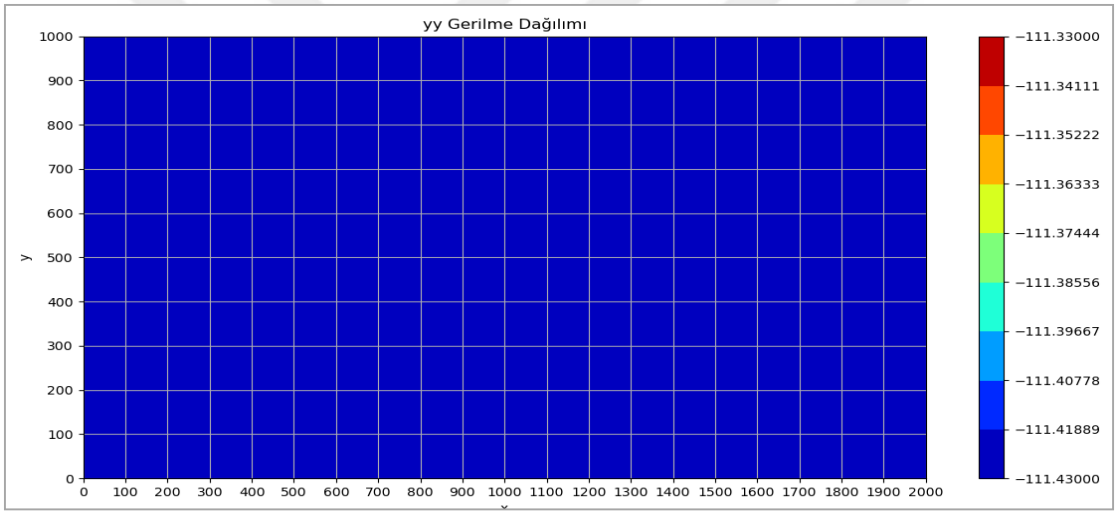
$$\sigma_y = \frac{1+\nu}{1-\nu^2} E\alpha\Delta T \tag{6.14}$$

Levhadaki sıcaklık farkı $\Delta T = 30℃$ olduğu durum için $\sigma_x = -111{,}43\ MPa$, $\sigma_y = -111{,}43\ MPa$ olarak hesaplanmıştır.

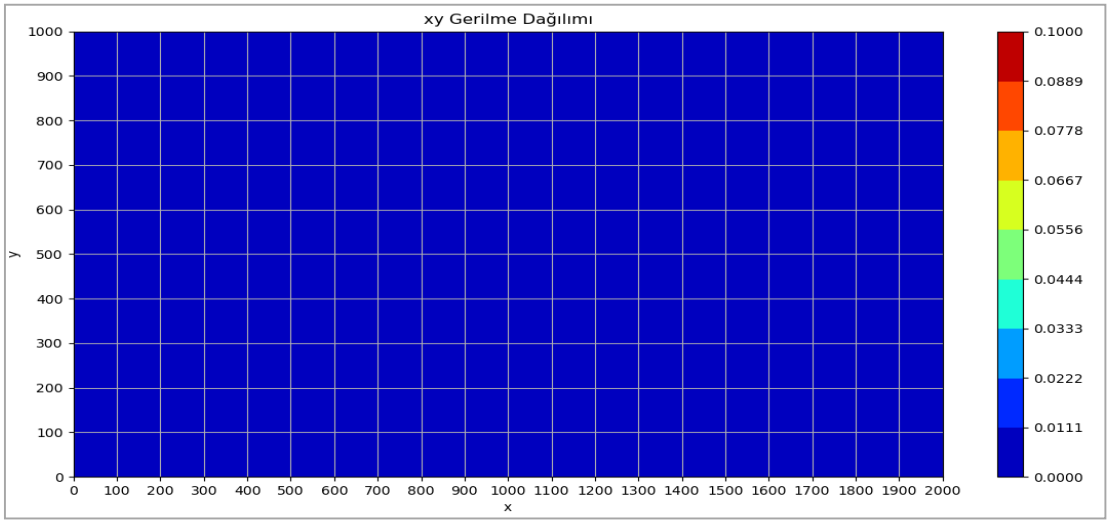
Bu örneğin Python ortamında oluşturulan yazılım ile çözülmesi için 21x11 çözüm ağı kullanılmıştır. Python-1 örneğinin üniform sıcaklık dağılımında yazılımla elde edilen ısıl gerilme sonuçları Şekil 6.3, Şekil 6.4 ve Şekil 6.5'te gösterilmiştir.



Şekil 6.3. Python-1 ÜSD için x doğrultusundaki normal gerilme sonuçları



Şekil 6.4. Python-1 ÜSD için y doğrultusundaki normal gerilme sonuçları



Şekil 6.5. Python-1 ÜSD için xy kesme gerilmesi sonuçları

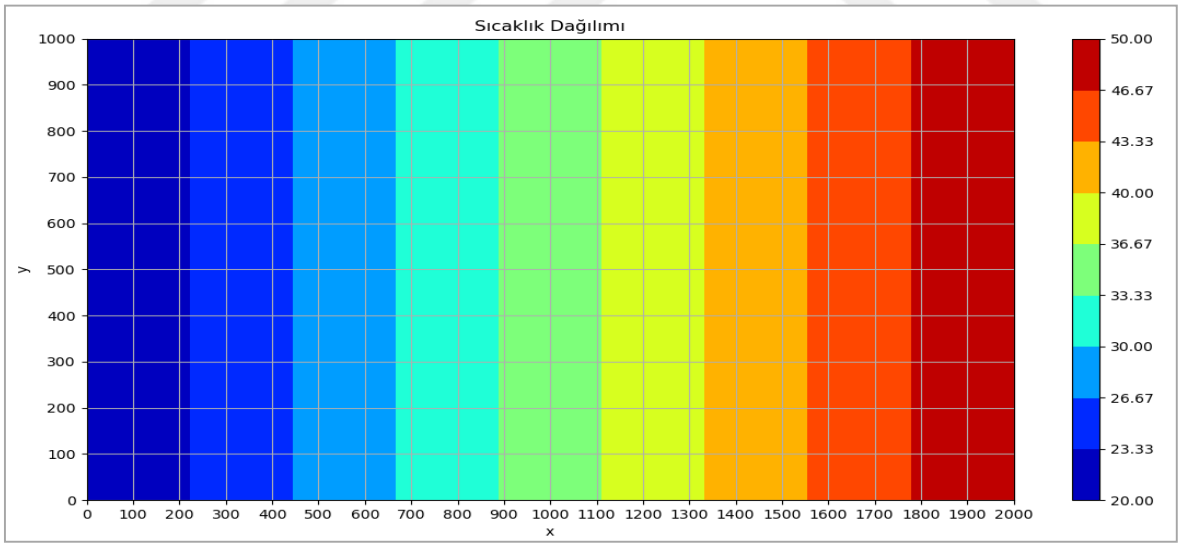
6.1.2. Lineer sıcaklık dağılımı durumu

Python-1 örneğinde verilen geometri ve yer değiştirme sınır şartlarının altında, lineer sıcaklık dağılımı elde etmek için levhanın alt ve üst kenarları izole kenar seçilmiş ve sol kenardan 20°C, sağ kenarından 50°C sıcaklık uygulanmıştır. Ortam ilk sıcaklığı 0°C'dir.

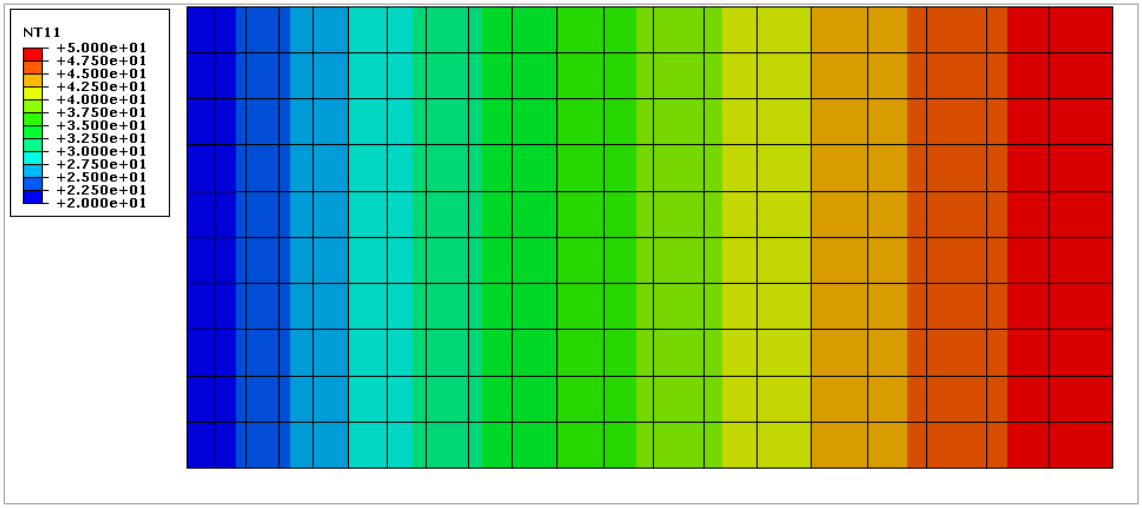
Çizelge 6.6. Python-1 LSD için sıcaklık sınır şartları

Ortam sıcaklığı	$T_{ortam} = 0\,^{\circ}C$
Sol kenardan uygulanan sıcaklık	$T_{sol} = 20^{\circ}C$
Sağ kenardan uygulanan sıcaklık	$T_{sağ} = 50^{\circ}C$

Ele alınan şartlar altında oluşacak sıcaklık dağılımı 21x11 çözüm ağı kullanılarak hesaplanmıştır. Yazılımdan elde edilen sıcaklık değerlerinin dağılımı Şekil 6.6'da, ABAQUS sonlu elemanlar yazılımından elde edilen sıcaklık değerlerinin dağılımı Şekil 6.7'de gösterilmiştir.

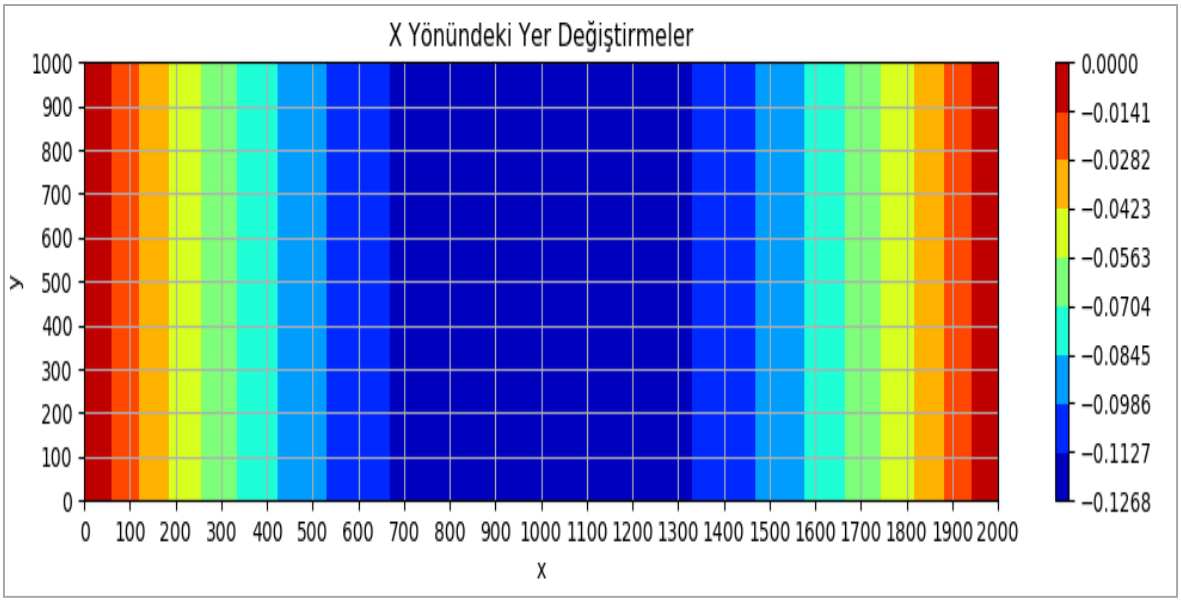


Şekil 6.6. Python-1 LSD için sıcaklık sonuçları

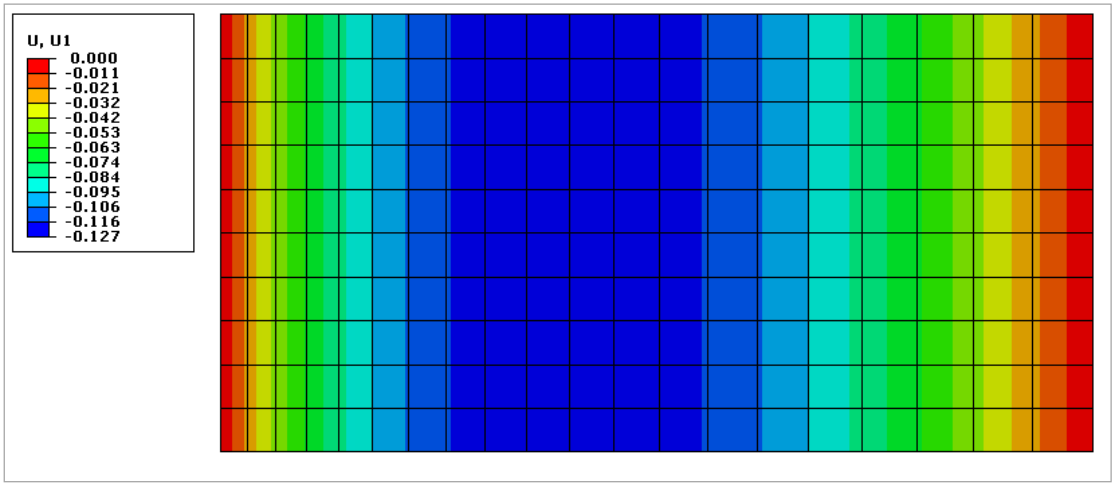


Şekil 6.7. Python-1 LSD için sıcaklık sonuçları (ABAQUS)

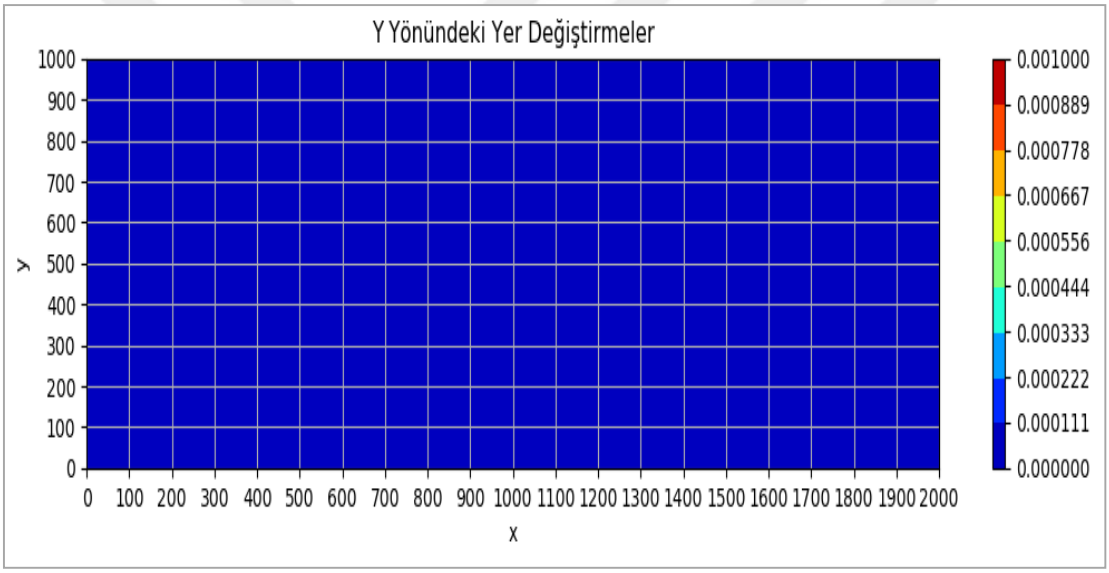
Sınır şartlarının ve yukarıda elde edilen sıcaklık dağılımının sonucunda oluşacak yer değiştirmeler Python-1 örneği için oluşturulan yazılım ile hesaplanmış ve Şekil 6.8 ve Şekil 6.10'da sırasıyla x ve y yönleri için gösterilmiştir. Aynı şartlar altında ABAQUS sonlu elemanlar yazılımından elde edilen yer değiştirme değerleri ise Şekil 6.9 ve Şekil 6.11'de verilmiştir.



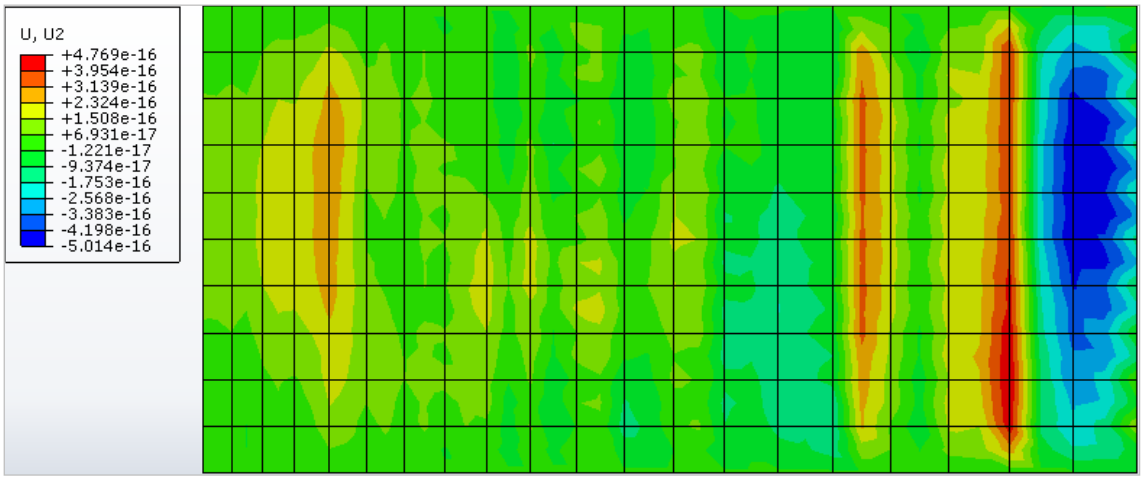
Şekil 6.8. Python-1 LSD için x yönündeki yer değiştirme sonuçları



Şekil 6.9. Python-1 LSD için x yönündeki yer değiştirme sonuçları (ABAQUS)

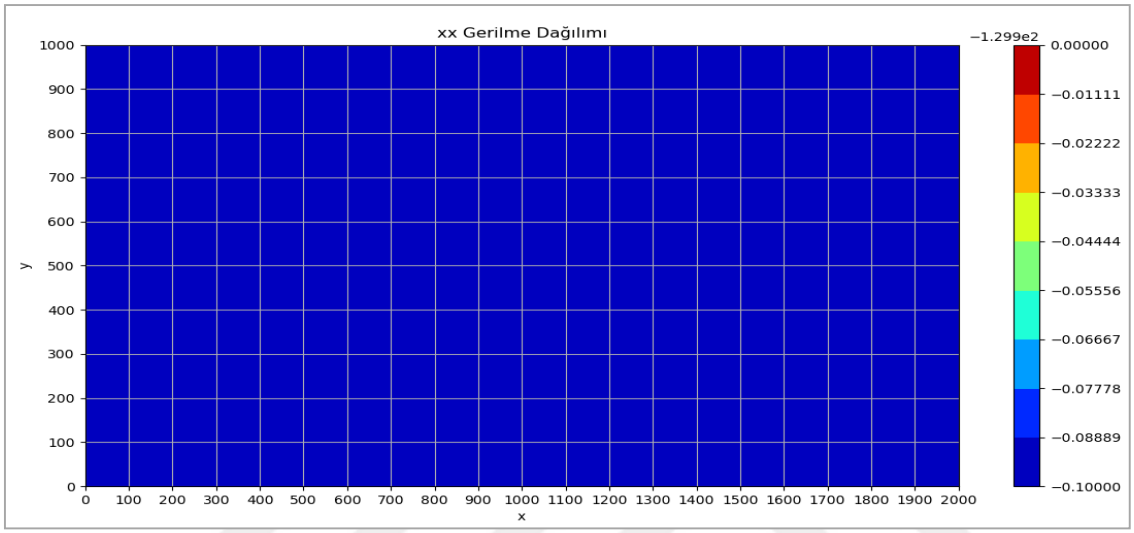


Şekil 6.10. Python-1 LSD için y yönündeki yer değiştirme sonuçları

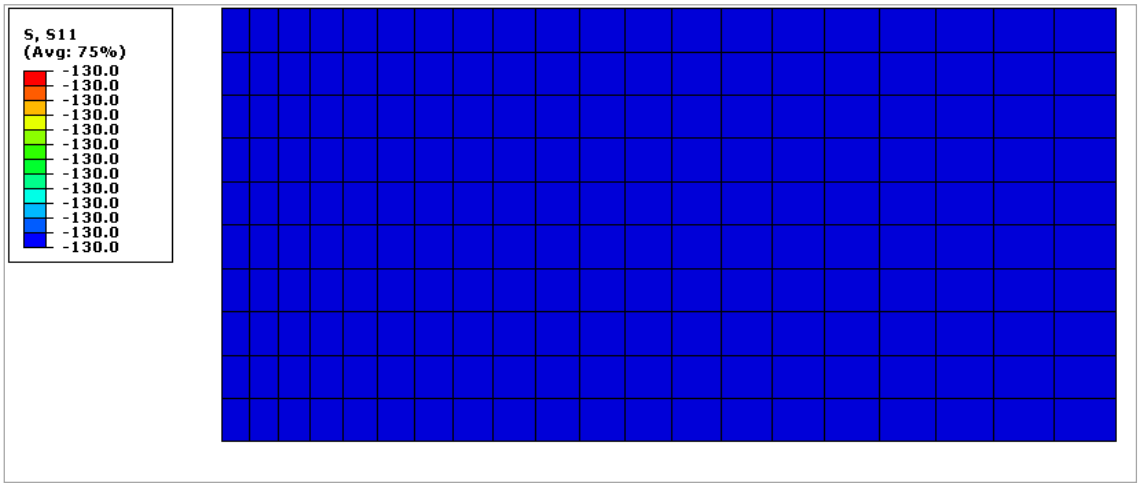


Şekil 6.11. Python-1 LSD için y yönündeki yer değiştirme sonuçları (ABAQUS)

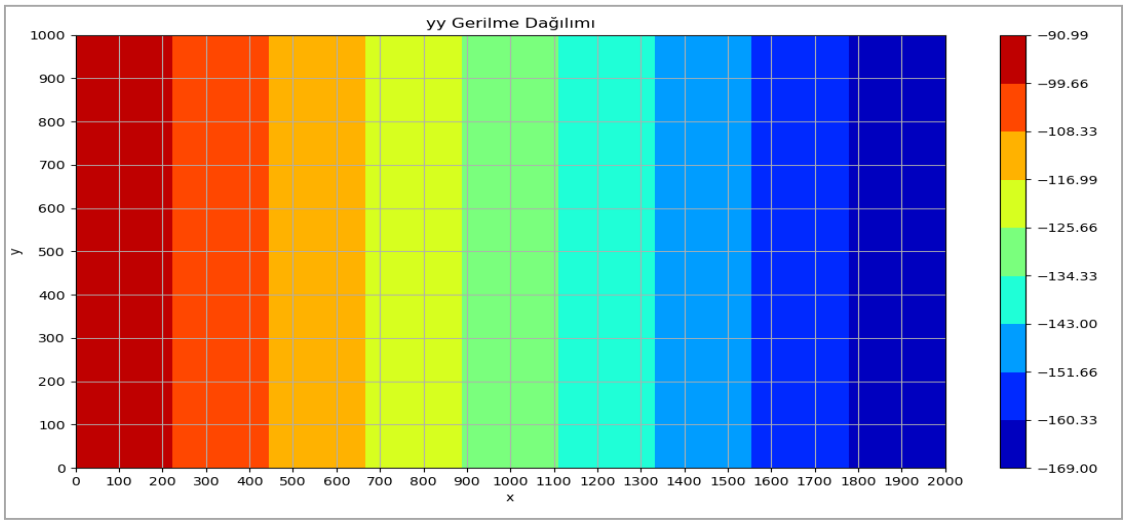
Şekil 6.6'da gösterilen sıcaklık dağılımı ve Şekil 6.8 ve 6.10'da gösterilen yer değiştime değerleri kullanılarak Python-1 örneği için yazılımdan elde edilen ısıl gerilme sonuçları Şekil 6.12 ve Şekil 6.14 ve Şekil 6.16'da gösterilmiştir. Aynı sınır şartları için ABAQUS sonlu elemanlar yazılımından elde edilen sonuçlar ise Şekil 6.13, Şekil 6.15 ve Şekil 6.17'de verilmektedir.



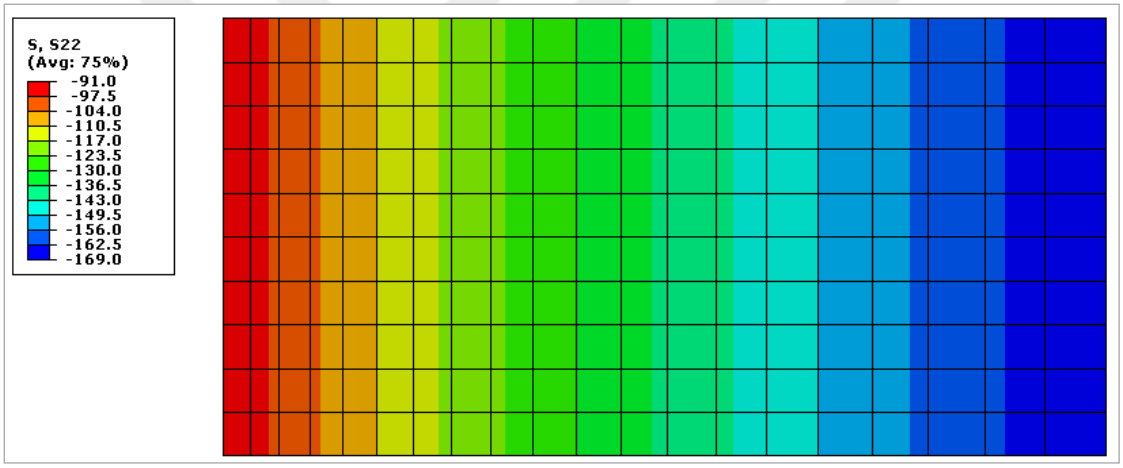
Şekil 6.12. Python-1 LSD için x doğrultusundaki normal gerilme sonuçları



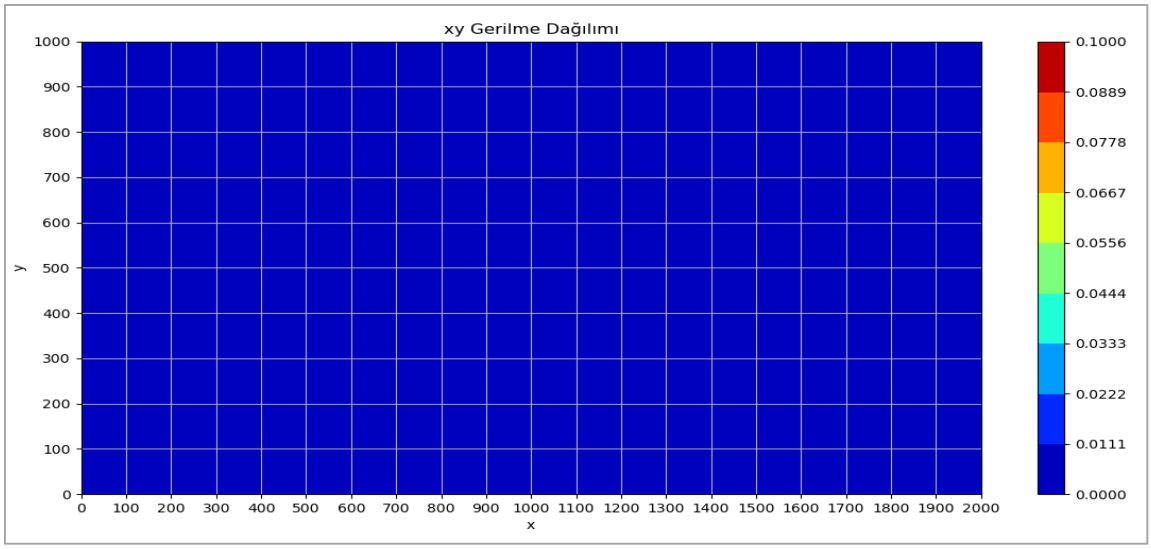
Şekil 6.13. Python-1 LSD için x doğrultusundaki normal gerilme sonuçları (ABAQUS)



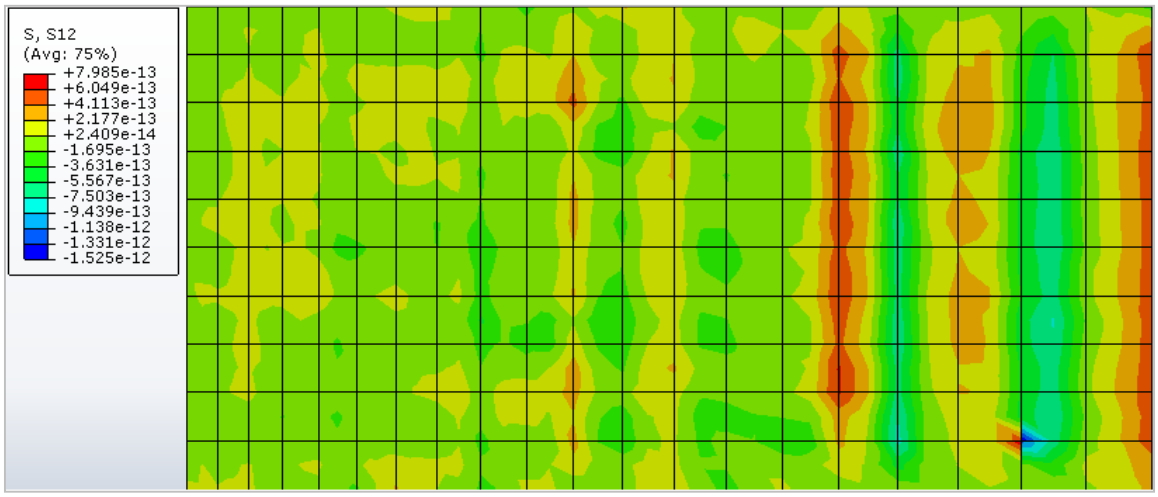
Şekil 6.14. Python-1 LSD için y doğrultusundaki normal gerilme sonuçları



Şekil 6.15. Python-1 LSD için y doğrultusundaki normal gerilme sonuçları (ABAQUS)

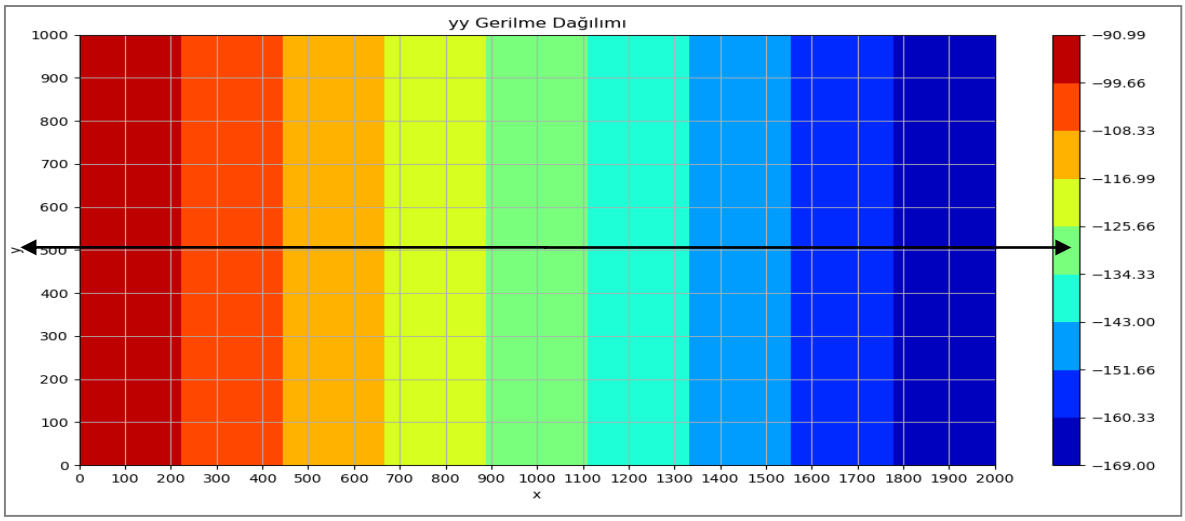


Şekil 6.16. Python-1 LSD için xy kesme gerilmesi sonuçları



Şekil 6.17. Python-1 LSD için xy kesme gerilmesi sonuçları(ABAQUS)

Şekil 6.18'de levhanın orta noktasından bir kesit alınmıştır. Kesit üzerindeki düğüm noktalarının yazılımdan ve ABAQUS sonlu elemanlar yazılımından elde edilen *y* doğrultusundaki normal gerilme değerleri Çizelge 6.7'de karşılaştırılarak gösterilmiştir. Python ortamında oluşturulan yazılımdan ve ABAQUS ile elde edilen değerler birbirlerine oldukça yakındır.



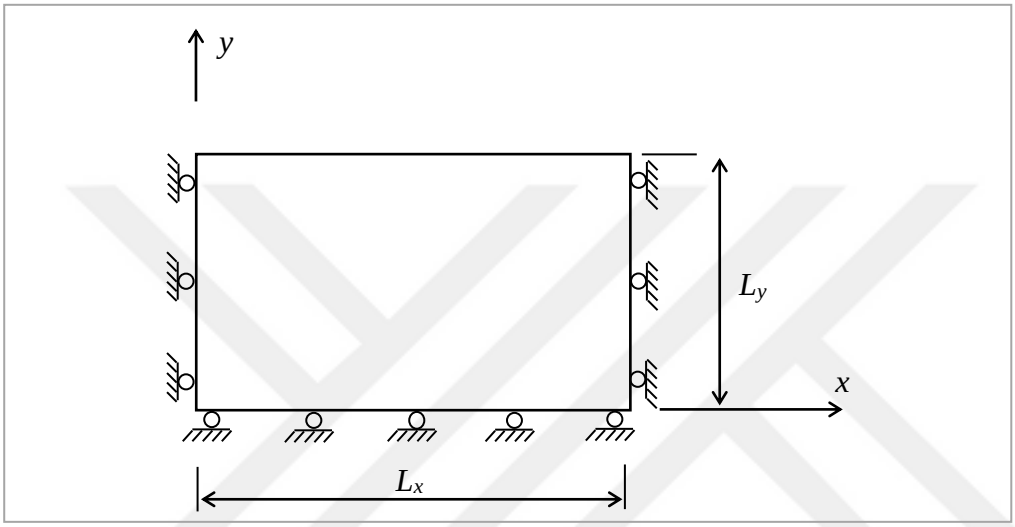
Şekil 6.18. Python-1 LSD için kesit alınan y doğrultusundaki normal gerilme grafiği

Çizelge 6.7. Python-1 LSD için y doğrultusundaki normal gerilme sonuçlarının karşılaştırılması

Düğüm Noktaları	Yazılım Sonuçları	ABAQUS Sonuçları	% Hata
$(\sigma_y)_{0,5}$	-90,99999935307278	-91	% $7{,}11\times10^{-7}$
$(\sigma_y)_{1,5}$	-94,89999903094336	-94,9	% $1{,}02\times10^{-6}$
$(\sigma_y)_{2,5}$	-98,79999870881394	-98,8	% $1{,}31\times10^{-6}$
$(\sigma_y)_{3,5}$	-102,69999840426456	-102,7	% $1{,}55\times10^{-6}$
$(\sigma_y)_{4,5}$	-106,59999812479425	-106,6	% $1{,}76\times10^{-6}$
$(\sigma_y)_{5,5}$	-110,49999787728453	-110,5	% $1{,}92\times10^{-6}$
$(\sigma_y)_{6,5}$	-114,3999976678299	-114,4	% $2{,}04\times10^{-6}$
$(\sigma_y)_{7,5}$	-118,29999750158775	-118,3	% $2{,}11\times10^{-6}$
$(\sigma_y)_{8,5}$	-122,19999738265159	-122,2	% $2{,}14\times10^{-6}$
$(\sigma_y)_{9,5}$	-126,09999731395003	-126,1	% $2{,}13\times10^{-6}$
$(\sigma_y)_{10,5}$	-129,99999729717467	-130	% $2{,}08\times10^{-6}$
$(\sigma_y)_{11,5}$	-133,89999733273862	-133,9	% $1{,}99\times10^{-6}$
$(\sigma_y)_{12,5}$	-137,7999974197662	-137,8	% $1{,}87\times10^{-6}$
$(\sigma_y)_{13,5}$	-141,69999755611443	-141,7	% $1{,}72\times10^{-6}$
$(\sigma_y)_{14,5}$	-145,59999773842603	-145,6	% $1{,}55\times10^{-6}$
$(\sigma_y)_{15,5}$	-149,49999796221184	-149,5	% $1{,}36\times10^{-6}$
$(\sigma_y)_{16,5}$	-153,3999982219615	-153,4	% $1{,}16\times10^{-6}$
$(\sigma_y)_{17,5}$	-157,2999985112792	-157,3	% $9{,}46\times10^{-7}$
$(\sigma_y)_{18,5}$	-161,1999988230409	-161,2	% $7{,}30\times10^{-7}$
$(\sigma_y)_{19,5}$	-165,09999914956995	-165,1	% $5{,}15\times10^{-7}$
$(\sigma_y)_{20,5}$	-168,99999947609902	-169	% $3{,}10\times10^{-7}$

6.2. Python-2 Örneği

Python-2 örneği olarak düşey kenarlarının *x* yönündeki yer değiştirmesi ve alt kenarın *y* yönündeki yer değiştirmesi engellenmiş lineer elastik malzeme özelliklerine sahip dikdörtgen bir levha ele alınmıştır. L_x uzunluğundaki ve L_y genişliğindeki dikdörtgen levha Şekil. 6.19'da gösterilmiştir.



Şekil 6.19. Sağ, sol ve alt kenarları kayıcı mafsal mesnetli levha

Dikdörtgen levhanın geometrik özellikleri, malzeme özellikleri ve sıcaklık sınır şartları Python-1 örneğiyle aynıdır. Dikdörtgen levhanın yer değiştirme sınır şartları ise Çizelge 6.8.'de verilmiştir.

Çizelge 6.8. Python-2 için yer değiştirme sınır şartları

$x = 0$	$u = 0,\ \frac{\partial u}{\partial x} = 0, \frac{\partial u}{\partial y} = 0, \frac{\partial v}{\partial x} = 0,\ \sigma_y = 0, \tau_{xy} = 0$
$x = 2000mm$	$u = 0,\ \frac{\partial u}{\partial x} = 0, \frac{\partial u}{\partial y} = 0, \frac{\partial v}{\partial x} = 0, \sigma_y = 0, \tau_{xy} = 0$
$y = 0$	$v = 0,\ \frac{\partial v}{\partial x} = 0, \frac{\partial u}{\partial x} = 0, \frac{\partial u}{\partial y} = 0,\ \sigma_y = 0, \tau_{xy} = 0$
$y = 1000mm$	$\frac{\partial v}{\partial x} = 0, \frac{\partial u}{\partial x} = 0, \frac{\partial u}{\partial y} = 0, \sigma_y = 0, \tau_{xy} = 0$

Çizelge 6.8'de ifade edilen sınır şartlarına ilaveten *y* yönündeki yer değiştirmenin *y* eksenindeki türevini ifade eden sınır şartının elde edilmesi aşağıda gösterilmiştir.

$$\sigma_y = \left(2\mu + \frac{2\mu\lambda}{\lambda+2\mu}\right)\left(\frac{\partial v}{\partial y}\right) + \frac{2\mu\lambda}{\lambda+2\mu}\left(\frac{\partial u}{\partial x}\right) - \frac{2\mu\beta}{\lambda+2\mu}(T - T_0) \tag{6.15}$$

$$\left(2\mu + \frac{2\mu\lambda}{\lambda+2\mu}\right)\left(\frac{\partial v}{\partial y}\right) = \frac{2\mu\alpha(3\lambda+2\mu)}{\lambda+2\mu}(T - T_0) \tag{6.16}$$

$$\left(\frac{\partial v}{\partial y}\right) = \frac{\alpha(3\lambda+2\mu)}{2(\lambda+\mu)}(T - T_0) \tag{6.17}$$

6.2.1. Üniform sıcaklık dağılımı durumu

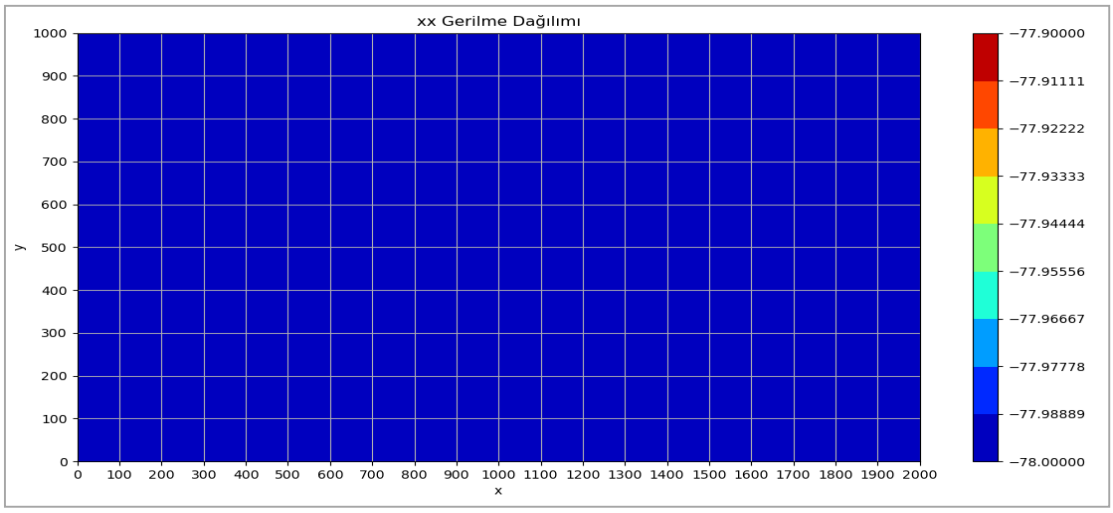
Python-2 örneğinde üniform sıcaklık dağılımı elde etmek için levhanın alt ve üst kenarları izole kenar seçilmiş, sol ve sağ kenardan 30℃ sıcaklık uygulanmıştır.

Ele alınan örnek için üniform sıcaklık dağılımı olması durumunda *x* yönündeki normal gerilme değeri analitik olarak Eş.6.18'de verildiği gibi hesaplanabilir [22].

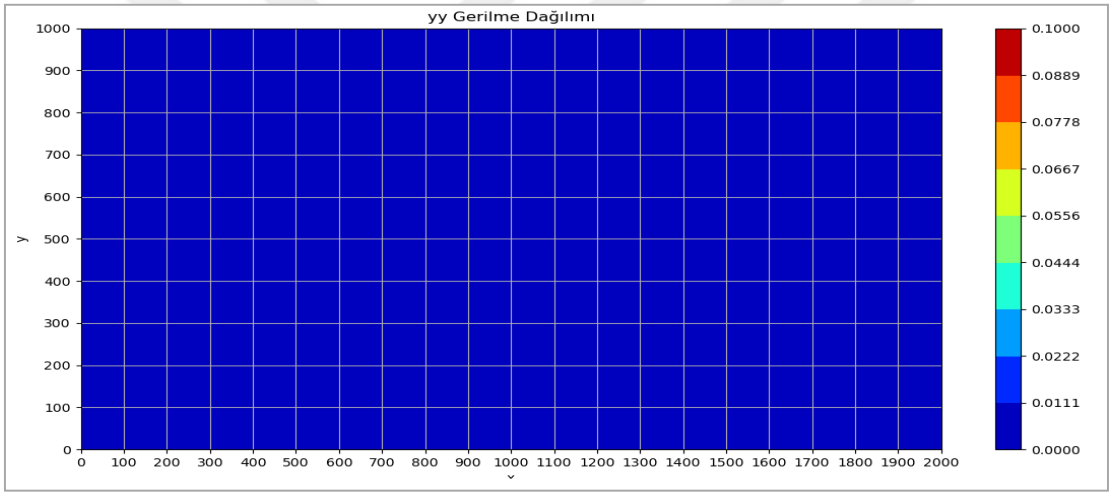
$$\sigma_x = E\alpha\Delta T \tag{6.18}$$

Levhadaki sıcaklık farkı $\Delta T = 30℃$ olduğu durumda σ_x normal gerilmesi -78MPa olarak hesaplanmıştır.

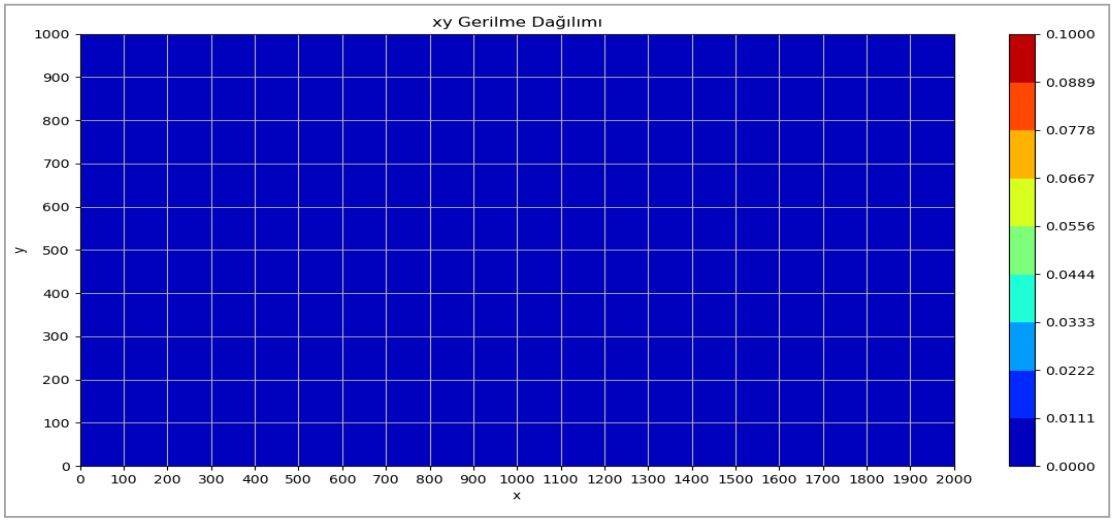
Üniform sıcaklık dağılımı için sonuçlar 21x11 çözüm ağı için hesaplanmıştır. Python-2 örneğinin üniform sıcaklık dağılımında yazılımla elde edilen ısıl gerilme sonuçları *x* doğrultusundaki normal gerilmeler için Şekil 6.20'de, *y* doğrultusundaki normal gerilmeler için Şekil 6.21'de ve xy kesme gerilmesi için 6.22'de gösterilmiştir.



Şekil 6.20. Python-2 ÜSD için x doğrultusundaki normal gerilme sonuçları



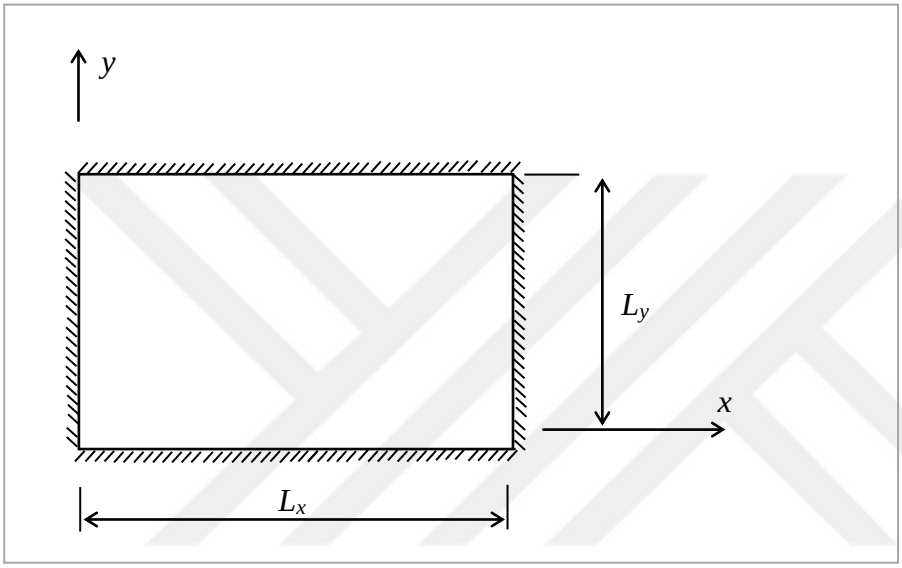
Şekil 6.21. Python-2 ÜSD için y doğrultusundaki normal gerilme sonuçları



Şekil 6.22. Python-2 ÜSD için xy kesme gerilmesi sonuçları

6.3. Python-3 Örneği

Bu örnekte hem düşey kenarların hem de yatay kenarların *x* ve *y* yönündeki yer değiştirmelerinin engellendiği, lineer elastik malzeme özelliklerine sahip dikdörtgen bir levha ele alınmıştır. L_x uzunluğundaki ve L_y genişliğindeki bu dikdörtgen levha Şekil 6.23'te gösterilmiştir.



Şekil 6.23. Dört kenarı sabit mesnetli levha

Dikdörtgen levhanın geometrik özellikleri, malzeme özellikleri ve sıcaklık sınır şartları Python-1 örneğiyle aynıdır. Dikdörtgen levhanın yer değiştirme sınır şartları ise Çizelge 6.9.'da verilmiştir.

Çizelge 6.9. Python-3 için yer değiştirme sınır şartları

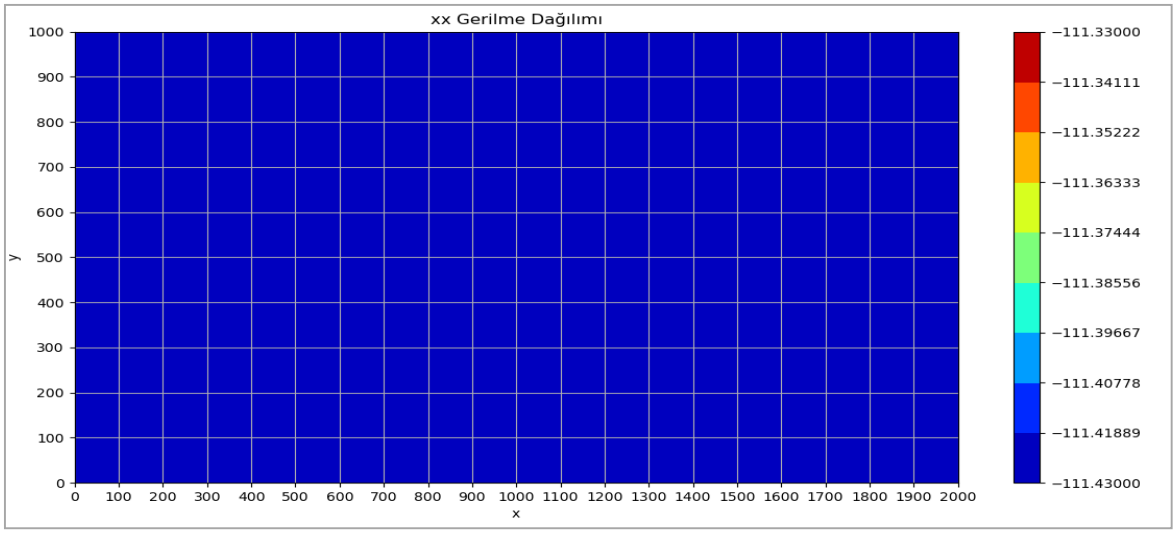
$x = 0$	$u = 0$, v=0, $\frac{\partial u}{\partial y} = 0$, $\frac{\partial v}{\partial y} = 0$
$x = 2000mm$	$u = 0$, v=0, $\frac{\partial u}{\partial y} = 0$, $\frac{\partial v}{\partial y} = 0$
$y = 0$	$u = 0$, v=0, $\frac{\partial u}{\partial x} = 0$, $\frac{\partial v}{\partial x} = 0$
$y = 1000mm$	$u = 0$, v=0, $\frac{\partial u}{\partial x} = 0$, $\frac{\partial v}{\partial x} = 0$

Çizelge 6.9'da ifade edilen sınır şartlarını kullanarak levhadaki tüm düğüm noktalarının *x* ve *y* yönündeki yer değiştirme değerleri hesaplanmıştır. Levhanın iç kısımlarındaki ve köşe noktalarındaki normal gerilme değerleri de hesaplanan yer değiştirmeler kullanılarak elde edilmiştir. Levhanın köşeleri hariç kenarlar üzerindeki normal gerilmeler ise Python-1 örneğindeki gibi hesaplanan iç kısımlardaki normal gerilme değerleri kullanılarak elde edilmiştir. (Bkz. Şekil 6.2)

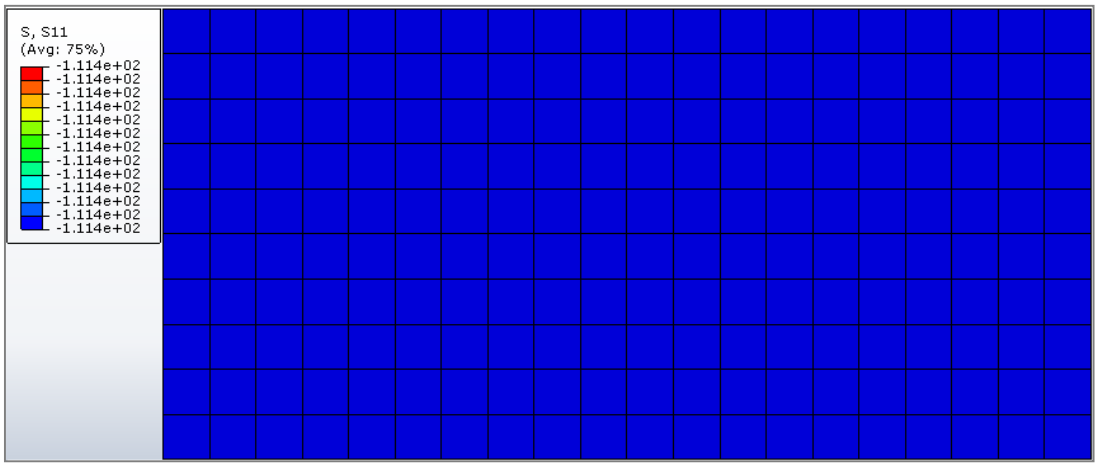
6.3.1. Üniform sıcaklık dağılımı durumu

Python-3 örneğinde üniform sıcaklık dağılımı elde etmek için levhanın alt ve üst kenarları izole kenar seçilmiş, sol ve sağ kenardan 30℃ sıcaklık uygulanmıştır. Ortamın ilk sıcaklığı 0℃'dir.

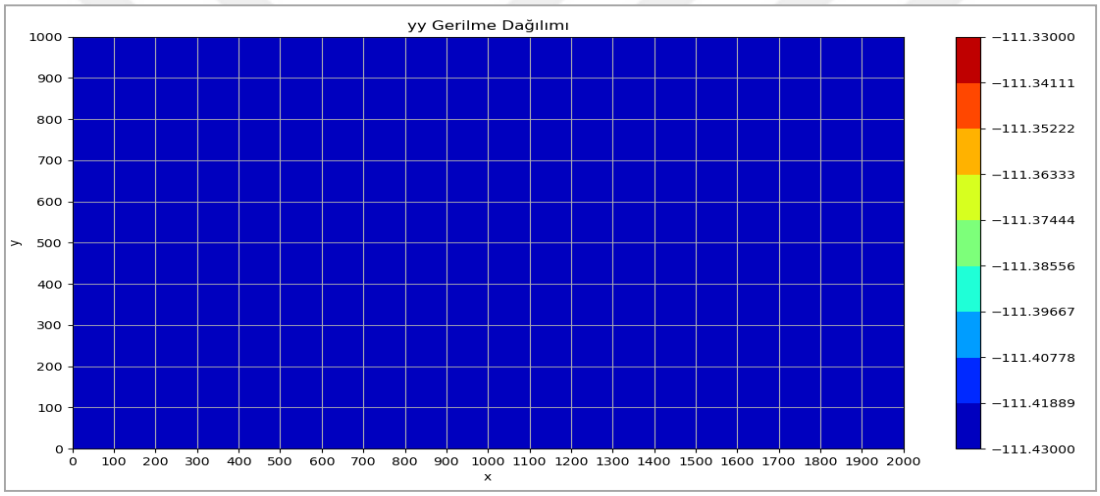
Yukarıda ifade edilen sıcaklık sınır şartları için 21x11 çözüm ağı kullanılarak sıcaklık dağılımı hesaplanmıştır. Bu sıcaklık dağılımı soncunda oluşan ve Python ortamında oluşturulan yazılımla elde edilen ısıl gerilme değerlerinin dağılımı Şekil 6.24, Şekil 6.26 ve Şekil 6.28'de, ABAQUS sonlu elemanlar yazılımından elde edilen gerilme değerlerinin dağılımı Şekil 6.25, Şekil 6.27 ve Şekil 6.29'da gösterilmiştir.



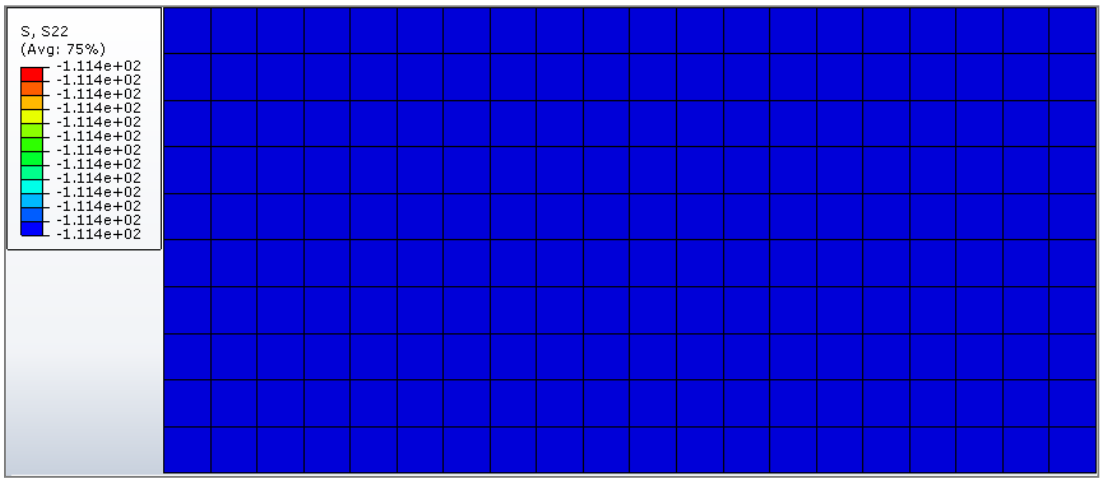
Şekil 6.24. Python-3 ÜSD için x doğrultusundaki normal gerilme sonuçları



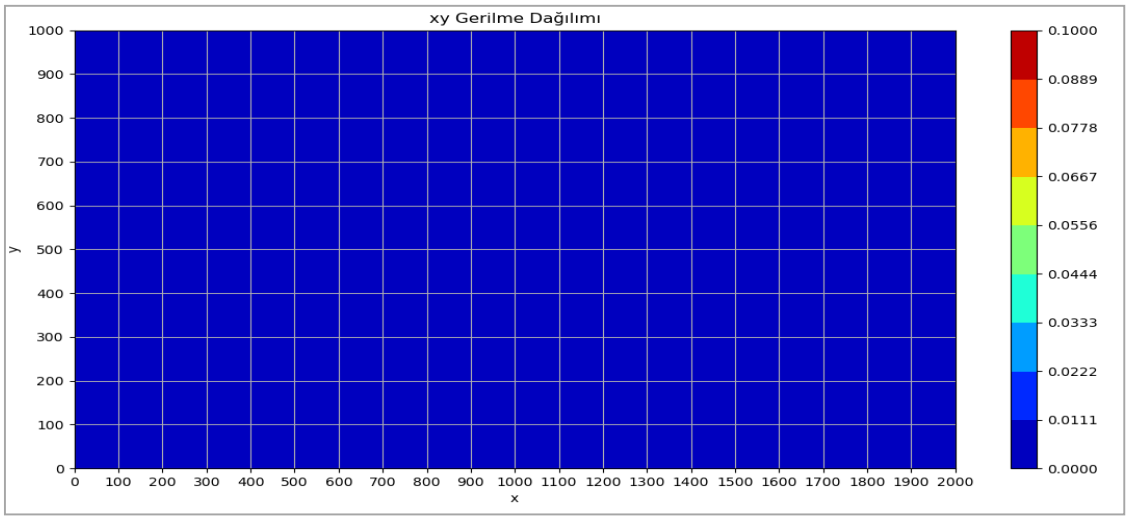
Şekil 6.25. Python-3 ÜSD için x doğrultusundaki normal gerilme sonuçları (ABAQUS)



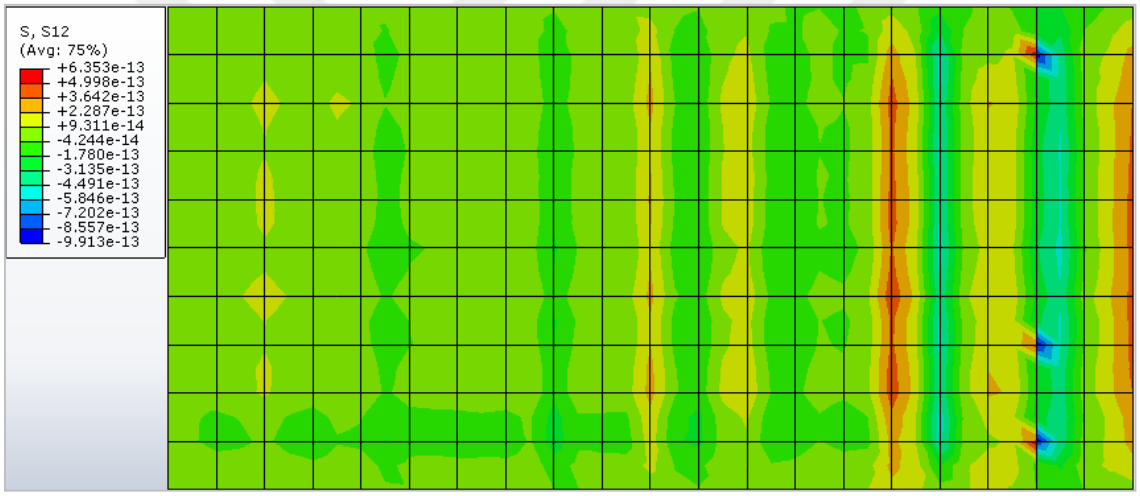
Şekil 6.26. Python-3 ÜSD için y doğrultusundaki normal gerilme sonuçları



Şekil 6.27. Python-3 ÜSD için y doğrultusundaki normal gerilme sonuçları (ABAQUS)



Şekil 6.28. Python-3 ÜSD için xy kesme gerilmesi sonuçları

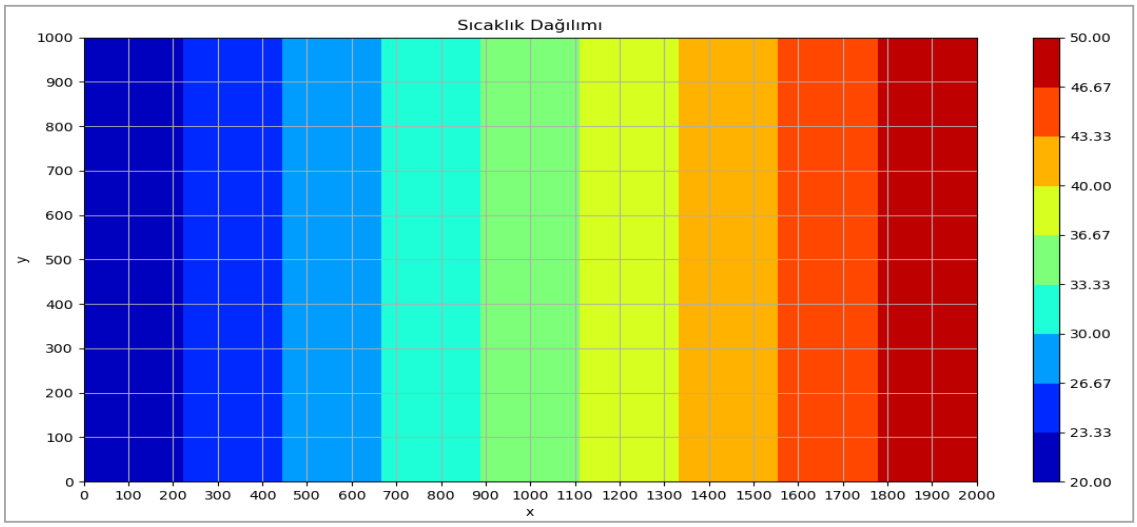


Şekil 6.29. Python-3 ÜSD için xy kesme gerilmesi sonuçları (ABAQUS)

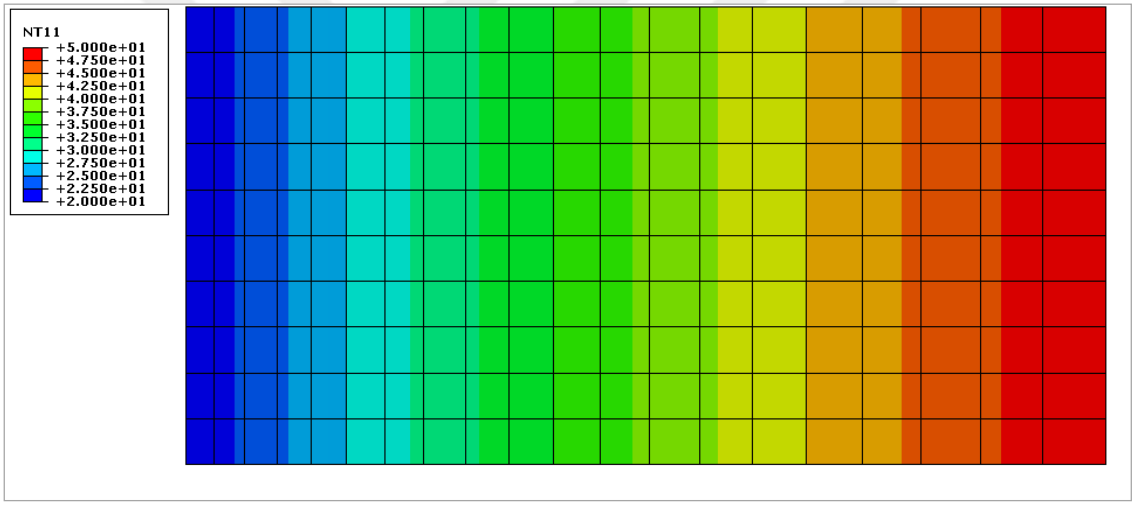
6.3.2. Lineer sıcaklık dağılımı durumu

Python-3 örneğinde lineer sıcaklık dağılımı elde etmek için levhanın alt ve üst kenarları izole kenar seçilmiştir. Sol kenardan 20℃, sağ kenarından 50℃ sıcaklık uygulanmıştır. Ortam ilk sıcaklığı 0℃'dir.

Yukarıda ifade edilen sıcaklık sınır şartları altında 21x11 çözüm ağı kullanılarak sıcaklık dağılımı hesaplanmıştır. Yazılımdan elde edilen sıcaklık değerlerinin dağılımı Şekil 6.30'da, ABAQUS sonlu elemanlar yazılımından elde edilen sonuçlar ise Şekil 6.31'de gösterilmiştir.

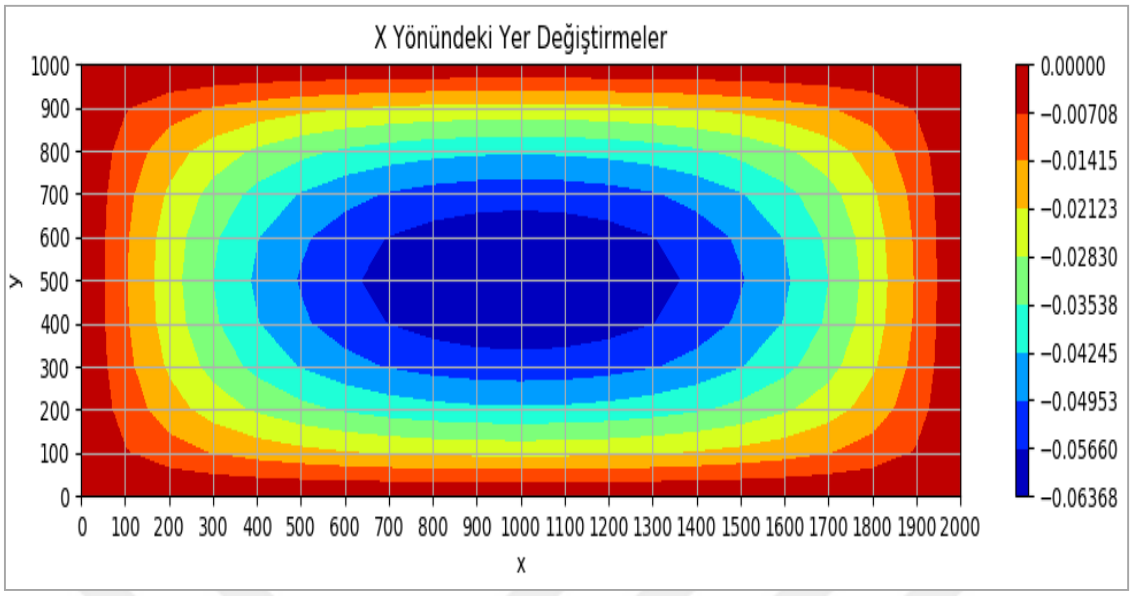


Şekil 6.30. Python-3 LSD için sıcaklık sonuçları

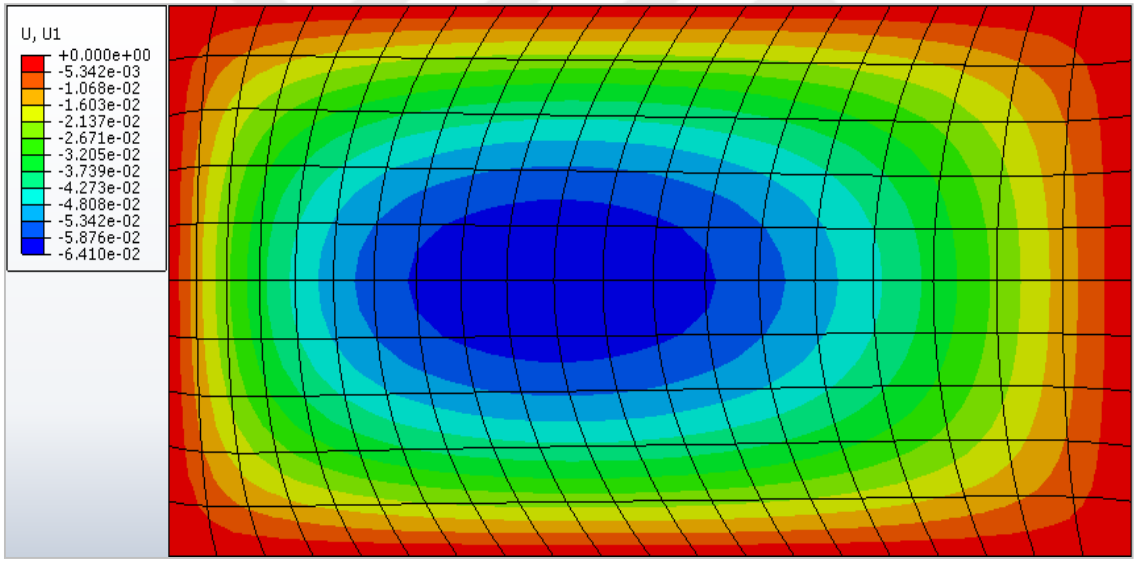


Şekil 6.31. Python-3 LSD için sıcaklık sonuçları (ABAQUS)

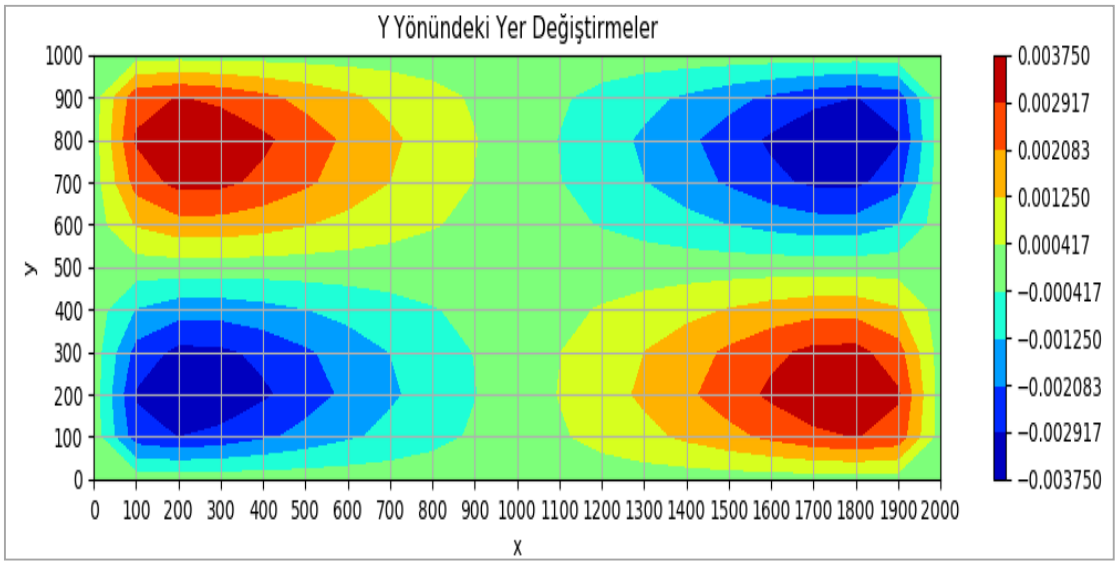
Şekil 6.30'da gösterilen lineer sıcaklık dağılımı (LSD) etkisindeki levha için belirtilen yer değiştirme sınır şartları ile birlikte Python ortamında oluşturulan yazılımdan elde edilen yer değiştirmeler x ve y yönlerinde sırasıyla Şekil 6.32 ve Şekil 6.34'te, ABAQUS sonlu elemanlar yazılımından elde edilen sonuçlar ise Şekil 6.33 ve Şekil 6.35'te gösterilmiştir.



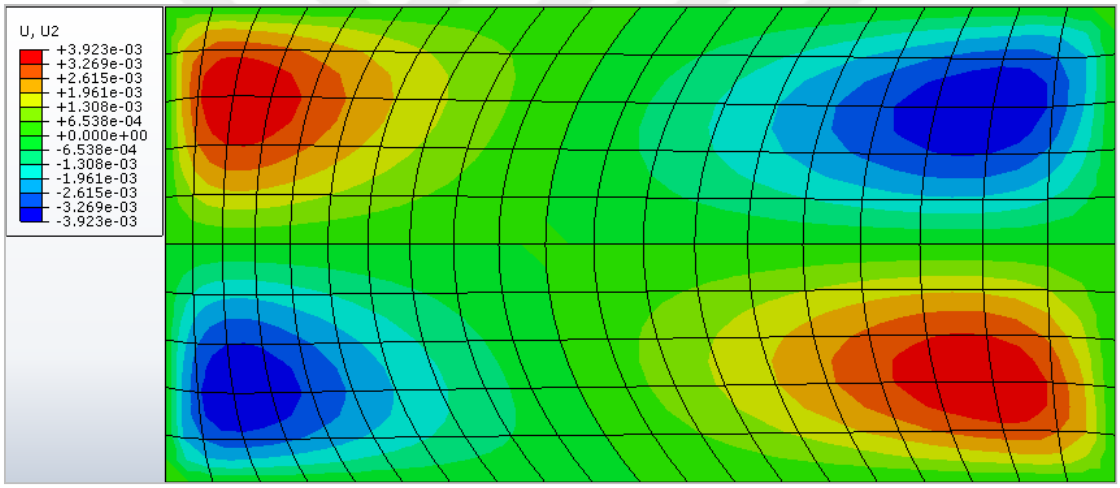
Şekil 6.32. Python-3 LSD için x yönündeki yer değiştirme sonuçları



Şekil 6.33. Python-3 LSD için x yönündeki yer değiştirme sonuçları (ABAQUS)

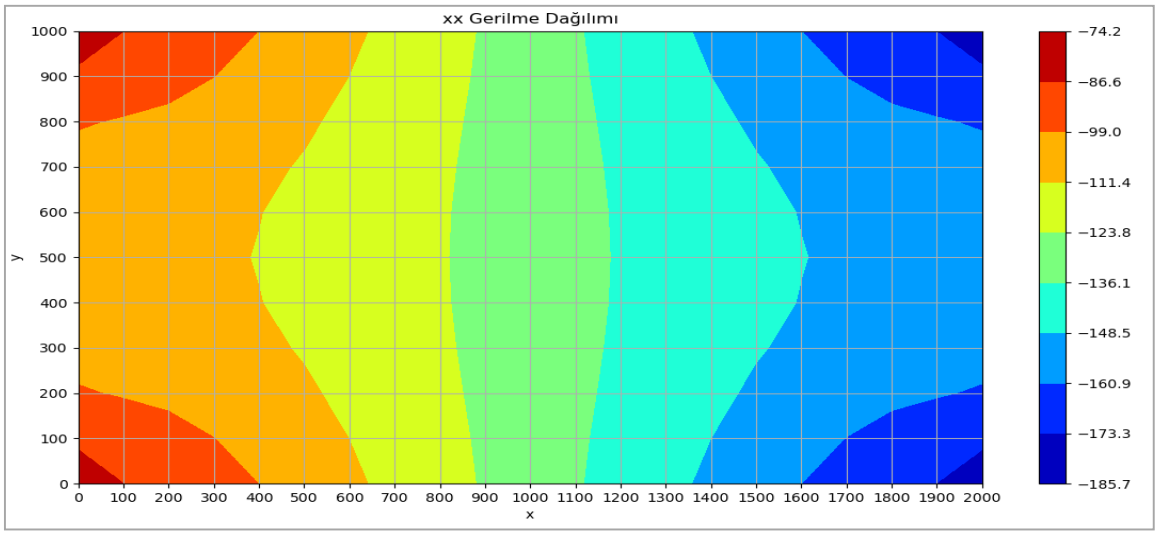


Şekil 6.34. Python-3 LSD için y yönündeki yer değiştirme sonuçları

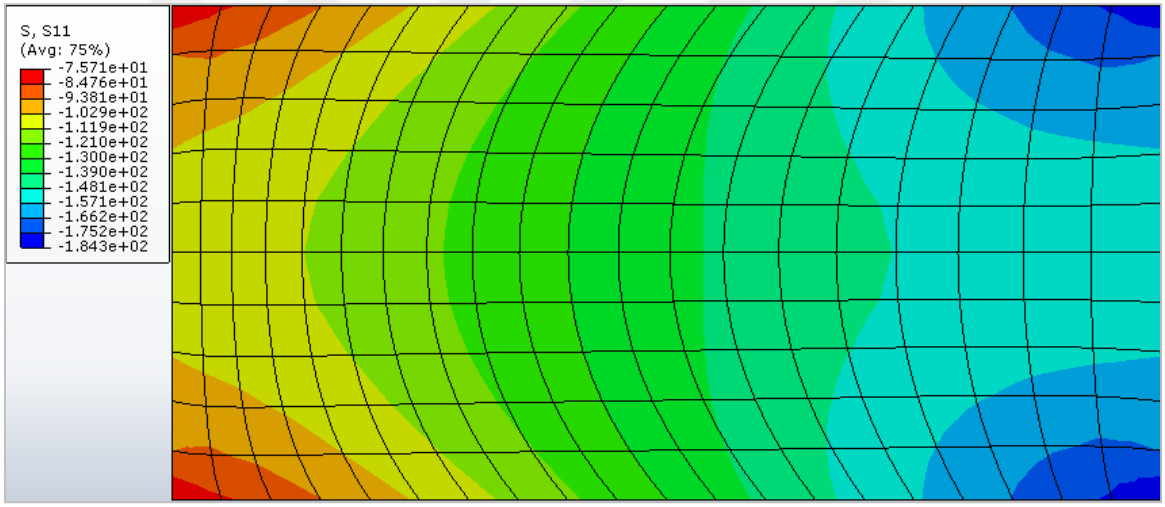


Şekil 6.35. Python-3 LSD için y yönündeki yer değiştirme sonuçları (ABAQUS)

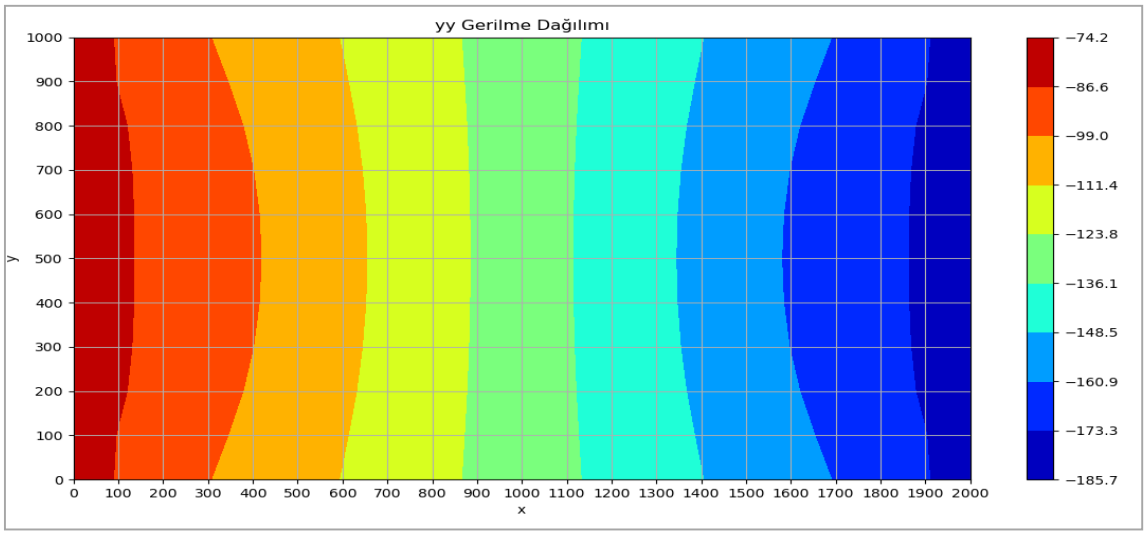
Python ortamında oluşturulan yazılım ile yukarıda verilen sıcaklık ve yer değiştirme dağılımları kullanılarak elde edilen ısıl gerilme değerleri *x* doğrultusundaki normal gerilmeler için Şekil 6.36'da ve *y* doğrultusundaki normal gerilmeler için Şekil 6.38'de verilmiştir. Benzer bir şekilde ABAQUS sonlu elemanlar yazılımından elde edilen sonuçlar ise Şekil 6.37 ve Şekil 6.39'da gösterilmiştir.



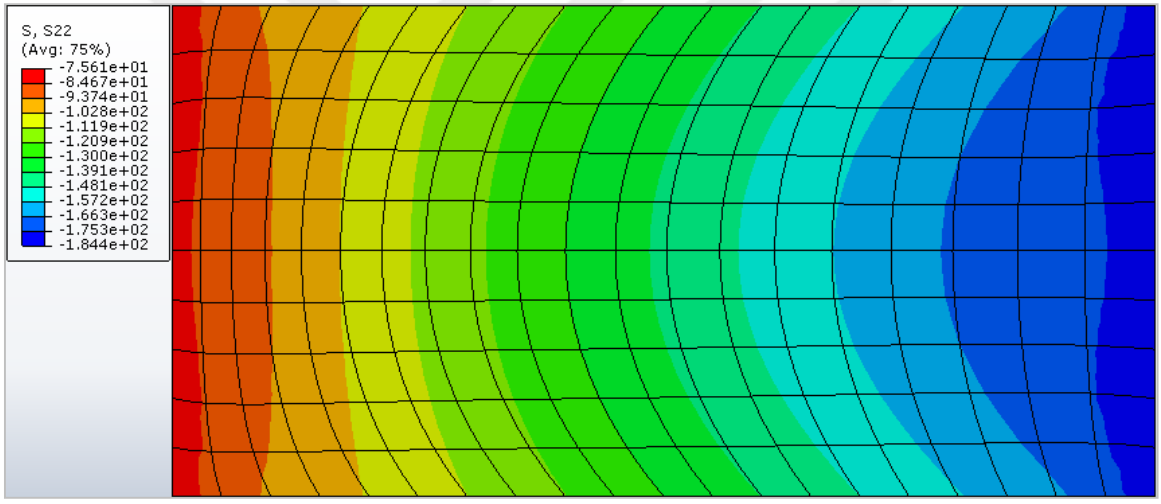
Şekil 6.36. Python-3 LSD için x doğrultusundaki normal gerilme sonuçları



Şekil 6.37. Python-3 LSD için x doğrultusundaki normal gerilme sonuçları (ABAQUS)

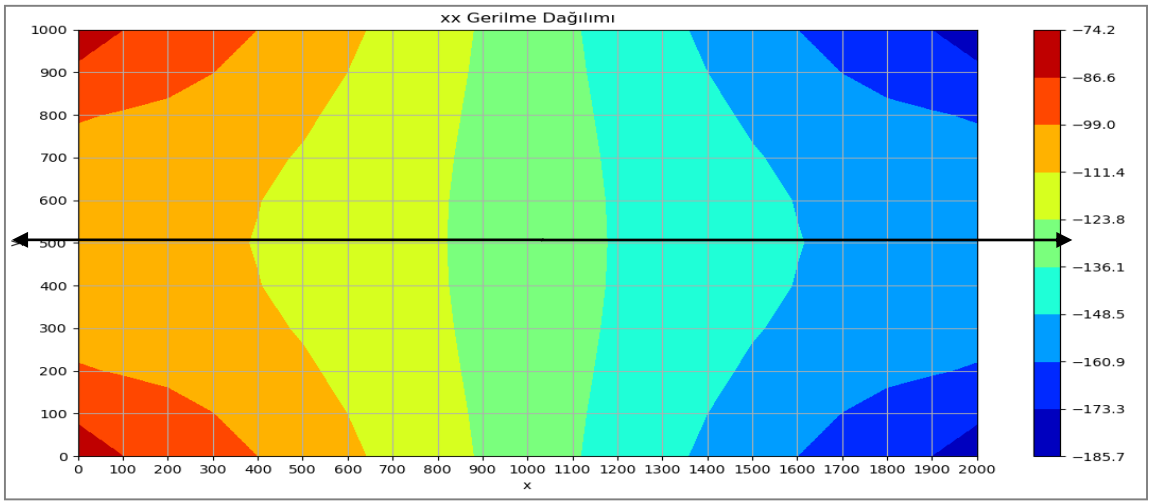


Şekil 6.38. Python-3 LSD için y doğrultusundaki normal gerilme sonuçları



Şekil 6.39. Python-3 LSD için y doğrultusundaki normal gerilme sonuçları (ABAQUS)

Şekil 6.40'da levhanın orta noktasından geçen yatay bir kesit çizgisi gösterilmektedir. Bu çizgi üzerindeki düğüm noktalarının yazılımdan ve ABAQUS sonlu elemanlar yazılımından elde edilen *x* doğrultusundaki normal gerilme değerleri Çizelge 6.10'da karşılaştırılarak gösterilmiştir. Her iki yazılımdan elde edilen sonuçların oldukça yakın oldukları görülmektedir.

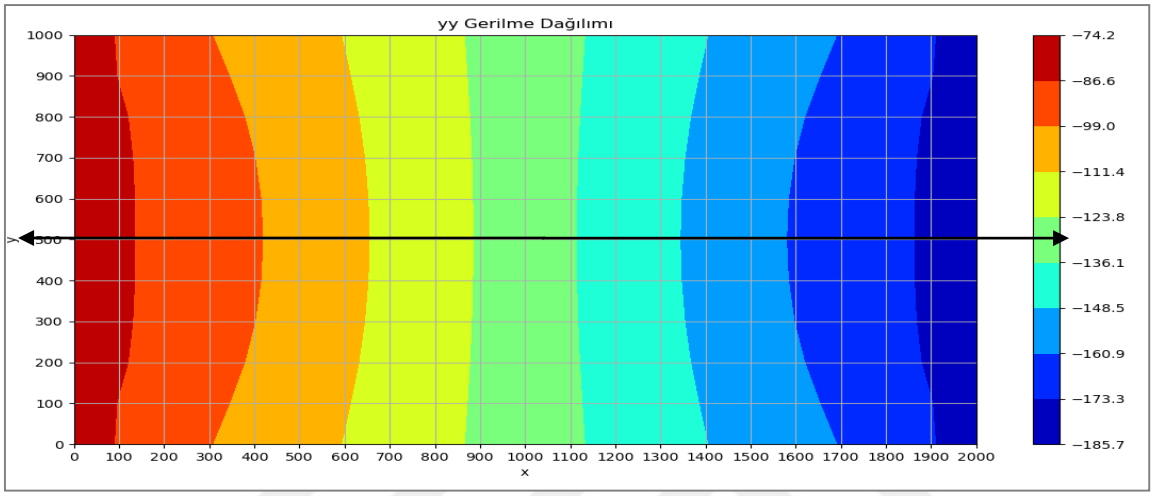


Şekil 6.40. Python-3 LSD için kesit alınan x doğrultusundaki normal gerilme grafiği

Çizelge 6.10. Python-3 LSD için x doğrultusundaki normal gerilme sonuçlarının karşılaştırılması

Düğüm Noktaları	Yazılım Sonuçları	ABAQUS Sonuçları	% Hata
$(\sigma_x)_{0,5}$	-105,6588694997986	-106,088	% 0,41
$(\sigma_x)_{1,5}$	-106,98784260761232	-107,395	% 0,38
$(\sigma_x)_{2,5}$	-108,31681571542605	-108,581	% 0,24
$(\sigma_x)_{3,5}$	-109,84689688484603	-109,982	% 0,12
$(\sigma_x)_{4,5}$	-111,74902299353501	-111,791	% 0,038
$(\sigma_x)_{5,5}$	-114,06989631831922	-114,059	% 0,01
$(\sigma_x)_{6,5}$	-116,77819069465569	-116,745	% 0,03
$(\sigma_x)_{7,5}$	-119,8059477865836	-119,77	% 0,03
$(\sigma_x)_{8,5}$	-123,0729123017221	-123,045	% 0,02
$(\sigma_x)_{9,5}$	-126,4974744760915	-126,483	% 0,01
$(\sigma_x)_{10,5}$	-129,99999787827625	-130	% $1{,}63\times10^{-6}$
$(\sigma_x)_{11,5}$	-133,50252129756072	-133,517	% 0,01
$(\sigma_x)_{12,5}$	-136,92708352156774	-136,955	% 0,02
$(\sigma_x)_{13,5}$	-140,19404811398533	-140,23	% 0,02
$(\sigma_x)_{14,5}$	-143,221805303032	-143,255	% 0,02
$(\sigma_x)_{15,5}$	-145,93009978640532	-145,941	% 0,01
$(\sigma_x)_{16,5}$	-148,25097321775178	-148,209	% 0,03
$(\sigma_x)_{17,5}$	-150,15309942470537	-150,018	% 0,09
$(\sigma_x)_{18,5}$	-151,68318068335995	-151,419	% 0,17
$(\sigma_x)_{19,5}$	-153,01215388165895	-152,605	% 0,27
$(\sigma_x)_{20,5}$	-154,34112707995794	-153,912	% 0,28

Benzer şekilde, Şekil 6.41'de levhanın orta noktasından geçirilen yatay kesit çizgisi gösterilmektedir. Kesit üzerindeki düğüm noktalarının yazılımdan ve ABAQUS sonlu elemanlar yazılımından elde edilen *y* doğrultusundaki normal gerilme değerleri Çizelge 6.11'de karşılaştırılarak gösterilmiştir. Burada da değerlerin oldukça yakın olduğu görülmektedir.



Şekil 6.41. Python-3 LSD için kesit alınan y doğrultusundaki normal gerilme grafiği

Çizelge 6.11. Python-3 LSD için y doğrultusundaki normal gerilme sonuçlarının karşılaştırılması

Düğüm Noktaları	Yazılım Sonuçları	ABAQUS Sonuçları	% Hata
$(\sigma_y)_{0,5}$	-82,00793946097187	-83,6538	% 2,01
$(\sigma_y)_{1,5}$	-85,39214458368204	-85,1985	% 0,23
$(\sigma_y)_{2,5}$	-88,77634970639221	-88,4555	% 0,36
$(\sigma_y)_{3,5}$	-93,13745648805906	-92,7988	% 0,36
$(\sigma_y)_{4,5}$	-98,02123682586831	-97,7289	% 0,30
$(\sigma_y)_{5,5}$	-103,17080581951413	-102,943	% 0,22
$(\sigma_y)_{6,5}$	-108,45335312121426	-108,287	% 0,15
$(\sigma_y)_{7,5}$	-113,80314567847043	-113,688	% 0,10
$(\sigma_y)_{8,5}$	-119,18769372691794	-119,115	% 0,061
$(\sigma_y)_{9,5}$	-124,59002976893666	-124,555	% 0,03
$(\sigma_y)_{10,5}$	-129,99999708764167	-130	$\% 2{,}24\times10^{-6}$
$(\sigma_y)_{11,5}$	-135,40996447519854	-135,445	% 0,03
$(\sigma_y)_{12,5}$	-140,81230072183385	-140,885	% 0,05
$(\sigma_y)_{13,5}$	-146,19684910477454	-146,312	% 0,08
$(\sigma_y)_{14,5}$	-151,54664211628253	-151,713	% 0,11
$(\sigma_y)_{15,5}$	-156,82918997693557	-157,057	% 0,14
$(\sigma_y)_{16,5}$	-161,97875961267755	-162,271	% 0,18
$(\sigma_y)_{17,5}$	-166,86254064464623	-167,201	% 0,20
$(\sigma_y)_{18,5}$	-171,22364812709526	-171,544	% 0,19
$(\sigma_y)_{19,5}$	-174,6078538924741	-174,802	% 0,11
$(\sigma_y)_{20,5}$	-177,99205965785296	-176,346	% 0,92

Farklı çözüm ağları için lineer sıcaklık dağılımına (LSD) sahip Python-1 örneğinde iteratif yöntem ile elde edilen ısıl gerilmeler için analiz süreleri Çizelge 6.12'de gösterilmiştir.

Çizelge 6.12. LSD altındaki farklı çözüm ağlarına sahip Python-1 örneği için hesaplama süreleri

Çözüm ağı	İterasyon sayısı	Sıcaklık hesaplama süreleri(sn)	İterasyon sayısı	Yer değiştirme hesaplama süreleri(sn)
4x4	192	0,06	153	0,10
5x5	324	0,10	270	0,18
6x6	501	0,19	420	0,37
7x7	704	0,32	586	0,70
8x8	963	0,54	776	1,23
9x9	1204	0,85	930	1,91
10x10	1545	1,35	1170	3,01
11x11	1855	1,98	1328	4,20
12x12	2234	3,07	1501	5,68
13x13	2594	3,78	1738	7,74
14x14	3106	5,36	1849	9,60
15x15	3569	6,86	2100	12,66
16x16	3944	8,71	2170	15,23
17x17	4524	11,36	2401	18,68
18x18	5159	14,17	2436	21,54
19x19	5819	18,23	2701	26,89
20x20	6507	25,50	2725	30,07
21x21	6894	26,94	2860	35,49
21x11	2885	6,32	2180	13,62

7. SONUÇLAR VE TARTIŞMA

Bu tez kapsamında sıcaklık etkisi altındaki iki boyutlu dikdörtgen levhada ısıl gerilme analizi yapılması için Python programlama dilinde bir yazılım oluşturulmuştur. Yazılımın başlangıçta sonlu farklar yöntemi gibi temel bir hesaplama yöntemi içermesi istenmiştir. Basit sınır şartları için analitik çözümler yer almakla birlikte, daha karmaşık şartları içeren problemlerin çözümleri için sayısal yöntemleri kullanan çeşitli araçların geliştirilmesi bir gerekliliktir. Ayrıca oluşturulan yazılımlar ticari amaçlı kullanılan yazılımlara göre daha ekonomik çözüm üretebilmektedir.

Öncelikle dikdörtgen bir levhanın çeşitli yer değiştirme sınır şartları ve sıcaklık sınır şartları altında sıcaklık dağılımları hesaplanmıştır. Üniform sıcaklık dağılımı ve doğrusal değişim gösteren sıcaklık dağılımı oluşturacak örnekler ele alınmıştır. Sıcaklık dağılımın belirlenmesinin ardından yer değiştirme dağılımı ve gerilme dağılımları elde edilmiştir. Python programlama dili ile oluşturulan yazılım kullanıcı dostu bir ara yüz penceresine sahip olacak şekildedir. Oluşturulan yazılımda termoelastisite denklemlerinin çözümü için sonlu farklar yöntemi kullanılmıştır. Bu yöntem sonucunda oluşan denklem sisteminin doğrudan çözümü yerine iteratif yöntem tercih edilmiştir.

Üniform sıcaklık dağılımına sahip örnekler için analitik gerilme fonksiyonları ile karşılaştırma yapılmış ve aynı sonuçlar elde edilmiştir. Doğrusal değişim gösteren sıcaklık dağılımına sahip örnekler için ABAQUS yazılımıyla hesaplanan değerler ile karşılaştırmalar yapılmıştır. Python ortamında oluşturulan yazılım ile ABAQUS sonlu elemanlar yazılımından elde edilen sonuçlar birbirlerine oldukça yakındır.

Çizelge 6.10 ve 6.11'de yer alan karşılaştırma sonuçlarına bakıldığında hata oranının levha sınırlarında daha fazla olduğu görülmektedir. Burada en büyük hata oranı yüzde iki civarında oluşmuştur.

Sonlu farklar yöntemi ile oluşturulan lineer denklem sistemi çözümünü için Jacobi iteratif yöntemi kullanılmıştır. Çözüm ağı sıklaştıkça iterasyon sayısı ve harcanan süre artmaktadır. Bilgisayar işlemcisinin harcadığı süre yaklaşık olarak iterasyon süresinin karesiyle doğru orantılı olacak şekilde artmaktadır.

KAYNAKLAR

1. Duhamel, J.M.C. (1838). Second memoire sur les phenomenes thermomecaniques. *Journal de I'Ecole Polytecnique*, 15(25), 1-57.

2. Thomson, W. (1857). On the thermo-elastic and thermo-magnetic properties of matter. *Quarterly Journal of Mathematics*, 1, 57-77.

3. Carter, J.P. and Booker, J.R. (1989). Finite element analysis of coupled thermoelasticity. *Computers & Structures*, 31(1), 73-80.

4. Balla, M. (1989). Formülation of coupled problems of thermoelasticity by finite elements. *Periodica Polytechnica Mechanical Engineerig*, 33(1-2), 59-70.

5. Chapra, S.C. and Canale, R.P. (2015). *Numerical methods for engineers* (7). New York/United States of America: McGRAW-HILL, 856-859.

6. Patil, S. and Prasad, J.S.V.R.K. (2013). Some steady-state thermoelastic problems of rectangular plates. *Applied Mathematical Sciences*,7(117), 5807-5819.

7. Verma, S. and Kulkarni, V.S. (2014). Thermal stress analysis of rectangular plate due to convection using finite element method. *International Journal of Engineering and Innovative Technology*, 3(7).

8. Pandit, B.B. and Kulkarni, V.S. (2015). Finite difference approach for non-homogeneous problem of thermal stresses in cartesian domain. *International Journal of Advances in Applied Mathematics and Mechanics*, 3(2), 100-112.

9. Silveira Junior, M.H., Piaz, C.E., Cislinski, J., Marciniak, A., Jurgensen, D.S., Melo, M.A. and Hacke, O. (2016). The finite difference method applied to the 1d thermoelastic problem. *XXXVII Iberian Latin American Congress on Computational Methods in Engineering*, 1, 1-5.

10. Gandhe, G.R., Ghugal, Y.M. and Kulkarni, V.S. (2016, December). *Thermoelastic stress analysis of rectangular plate by using ıntegral transform technique*. Paper presented at the First Structrual Engineering Convention, Chennai, INDIA.

11. Arnab, B., Islam, S.M.R., Khalak, A.A. and Afsar, A.M. (2014). Finite difference solution to thermoelastic field in a thin circular fgm disk with a concentric hole. *Procedia Engineering*,90, 193-198.

12. Hernandez, M.J., White, S.E., Chessa, J.F. and Ramana, C.V. (2015). Analytical and finie element analysis of thermal stresses in tin coatings. *Mechanics of Advanced Materials and Structures*, 22, 1024-1030.

13. Ersan, Ç. (2008). *Fonksiyonel derecelendirilmiş disklerde termal gerilme analizi*. Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Denizli.

14. Akarsu, M. (2013). *Üniform sıcaklık etkisindeki kompozit plakta ansys programı ile ısıl gerilme analizi*. Yüksek Lisans Tezi, Namık Kemal Üniversitesi Fen Bilimleri Enstitüsü, Tekirdağ.

15. Malvern, L.E. (1969). *Introduction to the mechanics of a continuous medium* (1). Englewood Cliffs: Prentice-Hall, 1-61.

16. Sadd, M.H. (2005). *Elasticity, theory, applications, and numerics* (Academic press). United States of America: Elsevier, 319-326.

17. Eslami, M.R., Hetnarski, R.B., Ignaczak, J., Noda, N., Sumi, N., Tanigawa, Y. (2013). *Theory of Elasticity and Thermal Stresses* (1). Dordrecht: Springer, 391-424.

18. Dassault Systemes, ABAQUS, Student Version, 2019.

19. İnternet: URL: http://web.archive.org/web/20191222180039/http://dsk.ippt.pan.pl/docs/abaqus/v6.13/books/gsa/default.htm, Son Erişim Tarihi: 30/12/2019.

20. Severance, C. (2015). Guido van Rossum: the early years of Python. *Computer*, 48(2), 7–9.

21. JetBrains, PyCharm, Community Edition, 2019.

22. Hetnarski, R.B. and Eslami, M.R. (2009). *Thermal Stresses, Advanced Theory and Applications (1).* New York: Springer, 27-29.

EKLER

EK-1. Sıcaklık modülü

```
import numpy as np
import matplotlib.pyplot as plt
import datetime
class SicaklikAnalizi:
    x = 0
    y = 0
    lx = 0.0
    ly = 0.0
    deltax = 0.0
    deltay = 0.0
    # kenarlardan verilen isi degerleri (arayuzden bu degiskenler atanirlar)
    t_sol = 0.0
    t_sag = 0.0
    t_ust = 0.0
    t_alt = 0.0
    t_ortam = 0.0
    # kenarlarin izole durumlari (0 izole, 1 isi akisi var)
    k_sol = 0
    k_sag = 0
    k_ust = 0
    k_alt = 0
    sicaklik_mat_t_ilk = None
    sicaklik_mat_t_son = None
    sicaklik_mat = None
    sicaklik_mat_aktif = None
```

EK-1. (devam) Sıcaklık modülü

```
    def __init__(self, x, y, lx, ly, t_sol, t_sag, t_ust, t_alt, t_ortam, k_sol, k_sag, k_ust, k_alt):
        # sınıfın calısması icin gereken veriler init özel fonksiyonu ile alınır.
        # Tanimlanan degiskenlere disaridan alinacak verilerin ilk atamasi burada yapilir.
        self.x = x
        self.y = y
        self.lx = lx
        self.ly = ly
        self.t_ust = t_ust
        self.t_alt = t_alt
        self.t_sol = t_sol
        self.t_sag = t_sag
        self.t_ortam = t_ortam
        self.k_sol = k_sol
        self.k_sag = k_sag
        self.k_ust = k_ust
        self.k_alt = k_alt
        self.deltax = round(float(self.lx / (self.x - 1)), 8)
        self.deltay = round(float(self.ly / (self.y - 1)), 8)
    # sınıfın fonksiyonu oldugu icin self paremetresi alır.
    def grafik_ciz(self):
        fig, fx = plt.subplots(1, 1) # 2x1 (2 satir 1 sutun 1. grafik)
        fig = plt.gcf()  # grafik penceresine isim verebilmek icin
        x_ax, y_ax = np.meshgrid(np.linspace(0, self.lx, self.x), np.linspace(0, self.ly, self.y))
        fx.set_title("Sıcaklık Dağılımı")
        fx.set_xticks(np.linspace(0, self.lx, self.x))
```

EK-1. (devam) Sıcaklık modülü

```
        fx.set_yticks(np.linspace(0, self.ly, self.y))

        fx.set_xlabel("x")

        fx.set_ylabel("y")

        fx.grid(True)

        color_bar_x_data = np.linspace(round(np.nanmin(self.sicaklik_mat), 2), round(np.nanmax(self.sicaklik_mat), 2) + 0.001, 10, endpoint=True)

        color_bar_x = fx.contourf(x_ax, y_ax, np.round(self.sicaklik_mat.transpose(), decimals=2), color_bar_x_data, cmap=plt.cm.jet)

        fig.colorbar(color_bar_x, ticks=color_bar_x_data, ax=fx)

        fig.canvas.set_window_title("Sıcaklık")

        plt.show()


    def sicaklik_hesapla(self):

        start = datetime.datetime.now() #sıcaklik hesaplamanın baslangic zamanini tutar

        # sıcaklık hesabinin yapildigi matris, baslangicta ortam sicaklıgi atanir

        self.sicaklik_mat = np.full((self.x, self.y), self.t_ortam, dtype='float')

        self.sicaklik_mat_t_ilk = self.sicaklik_mat.copy() #sicaklik_mat_t_ilk sıcaklık matrisinin ilk hali


        # sinir degerleri atamasi

        self.sicaklik_mat[:, self.y - 1] = self.t_ust

        self.sicaklik_mat[:, :1] = self.t_alt

        self.sicaklik_mat[self.x - 1:, :] = self.t_sag

        self.sicaklik_mat[:1, :] = self.t_sol

        #sınır deger atamasi yapılaman yerler

        # sol ust kose

        if self.k_ust == 1 and self.k_sol == 1:
```

EK-1. (devam) Sıcaklık modülü

```
            self.sicaklik_mat[0, self.y - 1] = None
        # sol alt kose
        if self.k_sol == 1 and self.k_alt == 1:
            self.sicaklik_mat[0, 0] = None
        # sag ust kose
        if self.k_alt == 1 and self.k_sag == 1:
            self.sicaklik_mat[self.x - 1, 0] = None
        # sag alt kose
        if self.k_sag == 1 and self.k_ust == 1:
            self.sicaklik_mat[self.x - 1, self.y - 1] = None
        # sıcaklık matrisinin kose elemanları None değilse hangi degerin atanacagını belirler
        if not np.isnan(self.sicaklik_mat[0,0]):
            if self.k_sol ==1:
                self.sicaklik_mat[0,0]=self.t_sol
            if self.k_alt==1:
                self.sicaklik_mat[0,0]=self.t_alt
        if not np.isnan(self.sicaklik_mat[0, self.y - 1]):
            if self.k_sol == 1:
                self.sicaklik_mat[0, self.y - 1] = self.t_sol
            if self.k_ust == 1:
                self.sicaklik_mat[0, self.y - 1] = self.t_ust
        if not np.isnan(self.sicaklik_mat[self.x - 1, 0]):
            if self.k_alt == 1:
                self.sicaklik_mat[self.x - 1, 0] = self.t_alt
            if self.k_sag == 1:
                self.sicaklik_mat[self.x - 1, 0] = self.t_sag
```

EK-1. (devam) Sıcaklık modülü

```
if not np.isnan(self.sicaklik_mat[self.x - 1, self.y - 1]):
    if self.k_sag == 1:
        self.sicaklik_mat[self.x - 1, self.y - 1] = self.t_sag
    if self.k_ust == 1:
        self.sicaklik_mat[self.x - 1, self.y - 1 ] = self.t_ust
 # ardişik iki kenardan ısı uygulandığında none olan elemanlara uygulanan ısının ortalamasını ata
if np.isnan(self.sicaklik_mat[0,0]):
        self.sicaklik_mat[0,0]= self.t_sol
if np.isnan(self.sicaklik_mat[0, self.y - 1]):
        self.sicaklik_mat[0, self.y - 1] = self.t_ust
if np.isnan(self.sicaklik_mat[self.x - 1, 0]):
        self.sicaklik_mat[self.x - 1, 0] = self.t_sag
if np.isnan(self.sicaklik_mat[self.x - 1, self.y - 1]):
        self.sicaklik_mat[self.x - 1, self.y - 1] = self.t_ust


self.sicaklik_mat_aktif = self.sicaklik_mat.copy()


itr = 0
# sicaklik_matrisi sicaklik_mat_aktif matrisi eşit mi.
#while true ise calismaya devam eder,false ise çikar.
# np.array_equal fonksiyonu, parametre olarak aldigi iki matrix birbirine tamamen esitse
# True doner, farkli ise False doner. Biz bunun aski durumunda while yapmak istedigimiz
# icin np.array_equal'den donen deger, "not" ile terse cevrilerek, matrislerin esit olmamasi
# durumunda donguye devam eder.
```

EK-1. (devam) Sıcaklık modülü

```
    while (not np.array_equal(np.round(self.sicaklik_mat[1:self.x - 1, 1:self.y - 1], decimals=8),
           np.round(self.sicaklik_mat_aktif[1:self.x - 1, 1:self.y - 1], decimals=8))) or itr == 0:
        itr += 1
        if itr >= 10000: # 10.000 adet donuste de bitmezde uyari ile looptan cikacak.
            break
        self.sicaklik_mat = self.sicaklik_mat_aktif.copy()
        for i in range(self.k_sol, self.x - self.k_sag):
            try:
                for j in range(self.k_alt, self.y - self.k_ust):
                    # try icindeki kod calistirilir. Eger basarili olursa islem tamamlanir.
                    # try blogunda hata alinirsa kod o satirdan except satirina atlar ve
                    # except blogundaki islemleri yapar.
                    try:
                        if i - 1 < 0:
                            n_sol = self.sicaklik_mat[i + 1, j]
                        else:
                            n_sol = self.sicaklik_mat[i - 1, j]
                    except Exception as e:
                        print("Except n_sol {}".format(e))

                    try:
                        n_sag = self.sicaklik_mat[i + 1, j]
                    except:
                        n_sag = self.sicaklik_mat[i - 1, j]
```

EK-1. (devam) Sıcaklık modülü

```
                try:
                    n_ust = self.sicaklik_mat[i, j + 1]
                except:
                    n_ust = self.sicaklik_mat[i, j - 1]


                try:
                    if j - 1 < 0:
                        n_alt = self.sicaklik_mat[i, j + 1]
                    else:
                        n_alt = self.sicaklik_mat[i, j - 1]
                except Exception as e:
                    print("Except n_alt {}".format(e))


                self.sicaklik_mat_aktif[i, j] = ((self.deltay ** 2 / (2 * self.deltay ** 2 + 2 *
self.deltax ** 2)) * n_sag) + \
                        ((self.deltay ** 2 / (2 * self.deltay ** 2 + 2 * self.deltax ** 2)) *
n_sol) + \
                        ((self.deltax ** 2 / (2 * self.deltay ** 2 + 2 * self.deltax ** 2)) *
n_ust) + \
                        ((self.deltax ** 2 / (2 * self.deltay ** 2 + 2 * self.deltax ** 2)) *
n_alt)


            except Exception as e:
                print("hata i,j: {},{} {}".format(i, j, e))
        self.sicaklik_mat = self.sicaklik_mat_aktif.copy()
        self.sicaklik_mat_t_son = self.sicaklik_mat.copy() # son sicaklik degerleri
        #
```

EK-1. (devam) Sıcaklık modülü

```
        istatistik_bilgi = “Sicaklik”.ljust(18) + “ ==>> X: {}, Y: {}”.format(self.x, self.y).ljust(13) + \
                        “ | {}”.format(itr).ljust(10) + “iterasyon” + “ | “ + \
                        “sure: {}”.format(datetime.datetime.now() - start)
        print(istatistik_bilgi)
        # bilgileri deplasmana gondermek icin arayuze dondur (arayuz.pyw)
        return  (self.sicaklik_mat_t_son - self.sicaklik_mat_t_ilk), self.sicaklik_mat_t_son
```

EK-2. Yer değiştirme modülü

```
import numpy as np
import matplotlib.pyplot as plt
import datetime

class YerDegistirmeAnalizi:
    prec = 18 # matrisin kac hanesi kontrol edilecek
    divider = 300 # 300 iterasyonda bir precision azaltacak
    x = 0
    y = 0
    lx = 0
    ly = 0
    u = None
    u_cr = None
    v = None
    v_cr = None
    deltax = 0.0
    deltay = 0.0
    teta = None
    em = 0.0
    nu = 0.0
    alfa = 0.0
    lamda = 0.0
    mu = 0.0
    # katsayilar ve sabitler U
    ks_u_i_a_1_j = 0.0
    ks_u_i_j_a_1 = 0.0
```

EK-2. (devam) Yer değiştirme modülü

```
    ks_u_i_a_1_j_a_1 = 0.0
    sabit_u = 0.0
    payda_u = 0.0
    # katsayilar ve sabitler V
    ks_v_i_j_a_1 = 0.0
    ks_v_i_a_1_j = 0.0
    ks_v_i_a_1_j_a_1 = 0.0
    sabit_v = 0.0
    payda_v = 0.0
    # u ve v matrisinde sabit noktalar listesi, ayni dictionary'dedir
    sabit_noktalar = None
    # Kenarlarin sabit yada serbest olup olmama durumu: 1 sabit, 0 serbest
    k_u_sol = 0
    k_u_sag = 0
    k_u_ust = 0
    k_u_alt = 0
    k_v_sol = 0
    k_v_sag = 0
    k_v_ust = 0
    k_v_alt = 0

    def __init__(self, x, y, lx, ly, teta, em, nu, alfa, sabit_noktalar,
                 k_u_sol, k_u_sag, k_u_ust, k_u_alt, k_v_sol, k_v_sag, k_v_ust, k_v_alt):
        self.x = x
        self.y = y
        self.lx = lx
```

EK-2. (devam) Yer değiştirme modülü

```
        self.ly = ly
        self.teta = teta
        self.em = em
        self.nu = nu
        self.alfa = alfa
        self.sabit_noktalar = sabit_noktalar
        self.deltax = float(self.lx / (self.x - 1))
        self.deltay = float(self.ly / (self.y - 1))
        self.lamda = (self.nu * self.em) / ((1 - 2 * self.nu) * (1 + self.nu))
        self.mu = self.em / (2 * (1 + self.nu))
        self.k_u_sol = k_u_sol
        self.k_u_sag = k_u_sag
        self.k_u_ust = k_u_ust
        self.k_u_alt = k_u_alt
        self.k_v_sol = k_v_sol
        self.k_v_sag = k_v_sag
        self.k_v_ust = k_v_ust
        self.k_v_alt = k_v_alt


        self.ks_u_i_a_1_j = (-4 * self.deltay ** 2) * (4 * self.mu * self.lamda + 4 * self.mu **
2)
        self.ks_u_i_j_a_1 = (-4 * self.deltax ** 2) * (self.mu * self.lamda + 2 * self.mu ** 2)
        self.ks_v_i_a_1_j_a_1 = (-1 * self.deltax * self.deltay) * (3 * self.mu * self.lamda + 2
* self.mu ** 2)
        self.payda_u = ((-8 * self.deltay ** 2) * (4 * self.mu * self.lamda + 4 * self.mu ** 2))
+ \
                       ((-8 * self.deltax ** 2) * (self.mu * self.lamda + 2 * self.mu ** 2))
```

EK-2. (devam) Yer değiştirme modülü

```
        self.ks_v_i_j_a_1 = (-4 * self.deltax ** 2) * (4 * self.mu * self.lamda + 4 * self.mu **
2)

        self.ks_v_i_a_1_j = (-4 * self.deltay ** 2) * (self.mu * self.lamda + 2 * self.mu ** 2)

        self.ks_u_i_a_1_j_a_1 = (-1 * self.deltax * self.deltay) * (3 * self.mu * self.lamda + 2
* self.mu ** 2)

        self.payda_v = ((-8 * self.deltax ** 2) * (4 * self.mu * self.lamda + 4 * self.mu ** 2))
+ \
                    ((-8 * self.deltay ** 2) * (self.mu * self.lamda + 2 * self.mu ** 2))


    def grafik_ciz(self):

        fig = plt.figure()

        fx = fig.add_subplot(2, 1, 1) # 2x1 (2 satir 1 sutun 1. grafik)

        fy = fig.add_subplot(2, 1, 2) # 2x1 (2 satir 1 sutun 2. grafik)

        fig = plt.gcf()  # grafik penceresine isim verebilmek icin

        x_ax, y_ax = np.meshgrid(np.linspace(0, self.lx, self.x), np.linspace(0, self.ly, self.y))

        # X grafigi

        fx.set_title('X Yönündeki Yer Değiştirmeler')

        fx.set_xticks(np.linspace(0, self.lx, self.x))

        fx.set_yticks(np.linspace(0, self.ly, self.y))

        fx.set_xlabel("x")

        fx.set_ylabel("y")

        fx.grid(True)

        # Y grafigi

        fy.set_title('Y Yönündeki Yer Değiştirmeler')

        fy.set_xticks(np.linspace(0, self.lx, self.x))

        fy.set_yticks(np.linspace(0, self.ly, self.y))

        fy.set_xlabel("x")
```

EK-2. (devam) Yer değiştirme modülü

```
        fy.set_ylabel("y")
        fy.grid(True)


        cb_x_min = round(np.nanmin(self.u), 5)
        cb_x_max = round(np.nanmax(self.u), 5)
        cb_y_min = round(np.nanmin(self.v), 5)
        cb_y_max = round(np.nanmax(self.v), 5)


        if cb_x_min == cb_x_max:
            cb_x_max += 0.001
        if cb_y_min == cb_y_max:
            cb_y_max += 0.001


        color_bar_x_data = np.linspace(cb_x_min, cb_x_max, 10, endpoint=True)
        color_bar_x = fx.contourf(x_ax, y_ax, np.round(self.u.transpose(), decimals=7), color_bar_x_data, cmap=plt.cm.jet)


        color_bar_y_data = np.linspace(cb_y_min, cb_y_max, 10, endpoint=True)
        color_bar_y = fy.contourf(x_ax, y_ax, np.round(self.v.transpose(), decimals=7), color_bar_y_data, cmap=plt.cm.jet)


        plt.subplots_adjust(bottom=0.1, left=0.2, right=0.9, top=0.9, hspace=0.4, wspace=0.1)
        fig.colorbar(color_bar_x, ticks=color_bar_x_data, ax=fx)
        fig.colorbar(color_bar_y, ticks=color_bar_y_data, ax=fy)
        fig.canvas.set_window_title("Yer Değiştirme")
        plt.show()
```

EK-2. (devam) Yer değiştirme modülü

```
    def yer_degistirme_hesapla(self):
        start = datetime.datetime.now()
        self.u = np.zeros((self.x, self.y), dtype='float')
        self.u_cr = np.zeros((self.x, self.y), dtype='float')
        self.v = np.zeros((self.x, self.y), dtype='float')
        self.v_cr = np.zeros((self.x, self.y), dtype='float')
        itr = 0
        # Durum (0): Tum kenarlar sabit
        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 1 and self.k_u_alt == 1 \
                and self.k_v_sol == 1 and self.k_v_sag == 1 and self.k_v_ust == 1 and self.k_v_alt == 1:
            while not \
                    (np.array_equal(
                        np.round(self.u[1:self.x - 1, 1:self.y - 1], decimals=(self.prec - (int(itr / self.divider)))),
                        np.round(self.u_cr[1:self.x - 1, 1:self.y - 1],
                                 decimals=(self.prec - (int(itr / self.divider))))) \
                            and np.array_equal(np.round(self.v[1:self.x - 1, 1:self.y - 1],
                                                        decimals=(self.prec - (int(itr / self.divider)))),
                                               np.round(self.v_cr[1:self.x - 1, 1:self.y - 1],
                                                        decimals=(self.prec - (int(itr / self.divider)))))) \
                    or itr == 0:
                itr += 1
                if itr >= 10000:
                    break
                self.u = self.u_cr.copy()
```

EK-2. (devam) Yer değiştirme modülü

```
            self.v = self.v_cr.copy()
            for i in range(1, self.x - 1):
                for j in range(1, self.y - 1):
                    self.sabit_u = (2 * self.deltax * self.deltay ** 2) * (2 * self.mu) * \
                                   (3 * self.lamda + 2 * self.mu) * self.alfa * (
                                   self.teta[i + 1, j] - self.teta[i - 1, j])

                    self.sabit_v = (2 * self.deltay * self.deltax ** 2) * (2 * self.mu) * \
                                   (3 * self.lamda + 2 * self.mu) * self.alfa * (
                                   self.teta[i, j + 1] - self.teta[i, j - 1])

                    self.u_cr[i, j] = (self.ks_u_i_a_1_j / self.payda_u) * (self.u[i + 1, j] + self.u[i
- 1, j]) + \
                                      (self.ks_u_i_j_a_1 / self.payda_u) * \
                                      (self.u[i, j + 1] + self.u[i, j - 1]) + (
                                                                              self.ks_v_i_a_1_j_a_1 / self.payda_u) *
\
                                                                              (self.v[i + 1, j + 1] - self.v[
                                                                                 i - 1, j + 1] -
                                                                               self.v[i + 1, j - 1] + self.v[
                                                                                 i - 1, j - 1]) + (
                                      self.sabit_u / self.payda_u)

                    self.v_cr[i, j] = (self.ks_v_i_j_a_1 / self.payda_v) * (self.v[i, j + 1] +
self.v[i, j - 1]) + \
                                      (self.ks_v_i_a_1_j / self.payda_v) * \
                                      (self.v[i + 1, j] + self.v[i - 1, j]) + (
```

EK-2. (devam) Yer değiştirme modülü

```
                                                    self.ks_u_i_a_1_j_a_1 / self.payda_v) *
\
                                                    (self.u[i + 1, j + 1] - self.u[
                                                        i - 1, j + 1] - self.u[
                                                         i + 1, j - 1] + self.u[
                                                         i - 1, j - 1]) + \
                                (self.sabit_v / self.payda_v)


        itr = 0
        # Durum (1)
        # U: sol sağ sabit, ust alt hareketli
        # V: alt üst sabit, ust alt hareketli
        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt ==
0 \
            and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 1 and self.k_v_alt
== 1:
            # esitlenince kendisi looptan cikacak, submatrix farki kontrol edilir.
            while not \
                (np.array_equal(np.round(self.u[1:self.x - 1, 1:self.y - 1], decimals=(self.prec-
(int(itr/self.divider)))), np.round(self.u_cr[1:self.x - 1, 1:self.y - 1], decimals=(self.prec-
(int(itr/self.divider))))) \
                and np.array_equal(np.round(self.v[1:self.x - 1, 1:self.y - 1], decimals=(self.prec-
(int(itr/self.divider)))), np.round(self.v_cr[1:self.x - 1, 1:self.y - 1], decimals=(self.prec-
(int(itr/self.divider)))))) \
                or itr == 0:
                itr += 1
                if itr >= 10000:
                    break
                self.u = self.u_cr.copy()
```

EK-2. (devam) Yer değiştirme modülü

```
            self.v = self.v_cr.copy()


            # u degerleri icin dongu (1)
            for i in range(self.k_u_sol, self.x - self.k_u_sag):
                for j in range(self.k_u_alt, self.y - self.k_u_ust):
                    self.yer_degistirme_donusum_d1(i, j, 'u')


            # v degerleri icin dongu (1)
            for i in range(self.k_v_sol, self.x - self.k_v_sag):
                for j in range(self.k_v_alt, self.y - self.k_v_ust):
                    self.yer_degistirme_donusum_d1(i, j, 'v')


        print_string = "Deplasman ==>>  X: {}, Y: {}".format(self.x, self.y).ljust(13) + \
                       " | " + "sure: {}".format(datetime.datetime.now() - start) + \
                       " | " + "{}".format(itr).ljust(12) + " iterasyon"
        print(print_string)


    # Durum (2) baslar.
    start = datetime.datetime.now()
    print_string = ""
    itr = 0
    # Durum (2)
    # U sağ-sol ve V alt sabit
    if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt ==
0 \
        and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 0 and self.k_v_alt
== 1: # 1 di geri al
```

EK-2. (devam) Yer değiştirme modülü

```
        while not \
            (np.array_equal(np.round(self.u[1:self.x - 1, 0:self.y ], decimals=(self.prec-(int(itr/self.divider)))), np.round(self.u_cr[1:self.x - 1, 0:self.y], decimals=(self.prec-(int(itr/self.divider))))) \
            and np.array_equal(np.round(self.v[0:self.x, 1:self.y], decimals=(self.prec-(int(itr/self.divider)))), np.round(self.v_cr[0:self.x, 1:self.y], decimals=(self.prec-(int(itr/self.divider))))) \
            or itr == 0:
            itr += 1
            if itr >= 10000:
                break
            self.u = self.u_cr.copy()
            self.v = self.v_cr.copy()

            # u degerleri icin dongu
            for i in range(self.k_u_sol, self.x - self.k_u_sag):
                for j in range(self.k_u_alt, self.y - self.k_u_ust):
                    self.yer_degistirme_donusum_d2(i, j, 'u')

            # v degerleri icin dongu
            for i in range(self.k_v_sol, self.x - self.k_v_sag):
                for j in range(self.k_v_alt, self.y - self.k_v_ust):
                    self.yer_degistirme_donusum_d2(i, j, 'v')

        print_string = "Deplasman ==>>  X: {}, Y: {}".format(self.x, self.y).ljust(13) + \
                       " | " + "sure: {}".format(datetime.datetime.now() - start) + \
                       " | " + "{}".format(itr).ljust(12) + " iterasyon"
        print(print_string)
```

EK-2. (devam) Yer değiştirme modülü

```
        # Durum (3) baslar
        start = datetime.datetime.now()
        print_string = ""
        itr = 0
        # Durum (3)
        # U sag-sol sabit. V serbest yalnizca belirlenmis noktalar sabit - 1 tane
        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt == 0 \
            and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 0 and self.k_v_alt == 0:
            while not (np.array_equal(
                    np.round(self.u[1:self.x - 1, 0:self.y], decimals=(self.prec - (int(itr / self.divider)))),
                    np.round(self.u_cr[1:self.x - 1, 0:self.y], decimals=(self.prec - (int(itr / self.divider))))) \
                        and np.array_equal(
                    np.round(self.v[0:self.x, 1:self.y], decimals=(self.prec - (int(itr / self.divider)))),
                    np.round(self.v_cr[0:self.x, 1:self.y], decimals=(self.prec - (int(itr / self.divider))))) \
                    or itr == 0:
                itr += 1
                if itr >= 10000:
                    break
                self.u = self.u_cr.copy()
                self.v = self.v_cr.copy()

                # u degerleri icin dongu
                for i in range(self.k_u_sol, self.x - self.k_u_sag):
```

EK-2. (devam) Yer değiştirme modülü

```
                for j in range(self.k_u_alt, self.y - self.k_u_ust):
                    if not “u:{},{}”.format(i, j) in self.sabit_noktalar:
                        self.yer_degistirme_donusum_d3(i, j, ‘u’)


            # v degerleri icin dongu
            for i in range(self.k_v_sol, self.x - self.k_v_sag):
                for j in range(self.k_v_alt, self.y - self.k_v_ust):
                    if not “v:{},{}”.format(i, j) in self.sabit_noktalar:
                        self.yer_degistirme_donusum_d3(i, j, ‘v’)


        print_string = “Deplasman ==>>  X: {}, Y: {}”.format(self.x, self.y).ljust(13) + \
                       “ | “ + “sure: {}”.format(datetime.datetime.now() - start) + \
                       “ | “ + “{}”.format(itr).ljust(12) + “ iterasyon”
        print(print_string)


    return self.u.copy(), self.v.copy(), self.mu, self.lamda


def yer_degistirme_donusum_d1(self, i, j, hesap):
    try:
        try:  # eger sol index disina cikarsa karsi degeri al
            if i - 1 < 0:
                teta_sol = self.teta[i + 1, j]
            else:
                teta_sol = self.teta[i - 1, j]
        except Exception as e:
            print(“Except teta_sol {}”.format(e))
```

EK-2. (devam) Yer değiştirme modülü

```
        try:  # eger sag index disina cikarsa karsi degeri al
            teta_sag = self.teta[i + 1, j]
        except Exception as e:
            teta_sag = self.teta[i - 1, j]
            # print("Except teta_sag {}".format(e))

        try:  # eger ust index disina cikarsa karsi degeri al
            teta_ust = self.teta[i, j + 1]
        except Exception as e:
            teta_ust = self.teta[i, j - 1]
            # print("Except teta_ust {}".format(e))

        try:  # eger alt index disina cikarsa karsi degeri al
            if j - 1 < 0:
                teta_alt = self.teta[i, j + 1]
            else:
                teta_alt = self.teta[i, j - 1]
        except Exception as e:
            print("Except teta_alt {}".format(e))

        # ********************** V donusumu **********************
        if hesap == "v":
            try:  # eger sol index disina cikarsa karsi degeri al
                if i - 1 < 0:
                    v_sol = self.v[i + 1, j]
                else:
```

EK-2. (devam) Yer değiştirme modülü

```
            v_sol = self.v[i - 1, j]
        except Exception as e:
            print("Except v_sol {}".format(e))

        try:  # eger sag index disina cikarsa karsi degeri al
            v_sag = self.v[i + 1, j]
        except Exception as e:
            v_sag = self.v[i - 1, j]

        try:  # eger ust index disina cikarsa karsi degeri al (kenara gore sinir icindedir)
            v_ust = self.v[i, j + 1]
        except Exception as e:
            # pass
            print("Except v_ust {}".format(e))

        try:  # eger alt index disina cikarsa karsi degeri al (kenara gore sinir icindedir)
            if j - 1 < 0:
                raise Exception("sinir disi v_alt")
            else:
                v_alt = self.v[i, j - 1]
        except Exception as e:
            # pass
            print("Except v_alt {}".format(e))

    # ********** U donusumu **********
    if hesap == "u":
```

EK-2. (devam) Yer değiştirme modülü

```
try:  # eger sol index disina cikarsa karsi degeri al
    if i - 1 < 0:
        raise Exception("sinir disi u_sol")
    else:
        u_sol = self.u[i - 1, j]
except Exception as e:
    print("Except u_sol {}".format(e))

try:  # eger sag index disina cikarsa karsi degeri al
    u_sag = self.u[i + 1, j]
except Exception as e:
    print("Except u_sag {}".format(e))

try:  # eger ust index disina cikarsa karsi degeri al
    u_ust = self.u[i, j + 1]
except Exception as e:
    u_ust = self.u[i, j - 1]

try:  # eger alt index disina cikarsa karsi degeri al
    if j - 1 < 0:
        u_alt = self.u[i, j + 1]
    else:
        u_alt = self.u[i, j - 1]
except Exception as e:
    print("Except u_alt {}".format(e))
```

EK-2. (devam) Yer değiştirme modülü

```
        self.sabit_u = (2 * self.deltax * self.deltay ** 2) * (2 * self.mu) * \
                       (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_sag - teta_sol)


        self.sabit_v = (2 * self.deltay * self.deltax ** 2) * (2 * self.mu) * \
                       (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_ust - teta_alt)


        if hesap == "u":
            if i == 0 or i == self.x - 1 or j == 0 or j == self.y - 1:  # kenarlar
                sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \
                         (u_ust + u_alt) + (self.sabit_u / self.payda_u)
            else:  # ic kisimlar
                sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \
                         (u_ust + u_alt) + (self.ks_v_i_a_1_j_a_1 / self.payda_u) * \
                         (self.v[i + 1, j + 1] - self.v[i - 1, j + 1] - self.v[i + 1, j - 1] + self.v[i - 1, j - 1]) +\
                         (self.sabit_u / self.payda_u)


            self.u_cr[i, j] = sayi_u


        if hesap == "v":
            if i == 0 or i == self.x - 1 or j == 0 or j == self.y - 1:  # kenarlar
                sayi_v = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \
                         (v_sag + v_sol) + (self.sabit_v / self.payda_v)
            else:
```

EK-2. (devam) Yer değiştirme modülü

```
                sayi_v = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \
                        (v_sag + v_sol) + (self.ks_u_i_a_1_j_a_1 / self.payda_v) * \
                        (self.u[i + 1, j + 1] - self.u[i - 1, j + 1] - self.u[i + 1, j - 1] + self.u[i - 1, j - 1]) + \
                        (self.sabit_v / self.payda_v)


            self.v_cr[i, j] = sayi_v


        except Exception as e:
            print("yer_degistirme_donusum_d1 i,j: {},{}, hata: {}".format(i, j, e))
            print(e)


    def yer_degistirme_donusum_d2(self, i ,j, hesap):
        try:
            try:  # eger sol index disina cikarsa karsi degeri al
                if i - 1 < 0:
                    teta_sol = self.teta[i + 1, j]
                else:
                    teta_sol = self.teta[i - 1, j]
            except Exception as e:
                print("Except teta_sol {}".format(e))


            try:  # eger sag index disina cikarsa karsi degeri al
                teta_sag = self.teta[i + 1, j]
            except Exception as e:
                teta_sag = self.teta[i - 1, j]
```

EK-2. (devam) Yer değiştirme modülü

```
        try:  # eger ust index disina cikarsa karsi degeri al
            teta_ust = self.teta[i, j + 1]
        except Exception as e:
            teta_ust = self.teta[i, j - 1]


        try:  # eger alt index disina cikarsa karsi degeri al
            if j - 1 < 0:
                teta_alt = self.teta[i, j + 1]
            else:
                teta_alt = self.teta[i, j - 1]
        except Exception as e:
            print("Except teta_alt {}".format(e))


        # bu problemde U'nun tum kenarlari sabit oldugundan donusum sadece V icin yapilir
        if hesap == "v":
            try:  # eger sol index disina cikarsa karsi degeri al
                if i - 1 < 0:
                    v_sol = self.v[i + 1, j]
                else:
                    v_sol = self.v[i - 1, j]
            except Exception as e:
                print("Except v_sol {}".format(e))


            try:  # eger sag index disina cikarsa karsi degeri al
                v_sag = self.v[i + 1, j]
```

EK-2. (devam) Yer değiştirme modülü

```
            except Exception as e:
                v_sag = self.v[i - 1, j]


            try:  # eger ust index disina cikarsa karsi degeri al
                v_ust = self.v[i, j + 1]
            except Exception as e:
                v_ust = self.v[i, j - 1] + (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                            (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)


            try:  # eger alt index disina cikarsa karsi degeri al
                if j - 1 < 0:
                    v_alt = self.v[i, j + 1] - (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                                (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
                else:
                    v_alt = self.v[i, j - 1]
            except Exception as e:
                print("Except v_alt {}".format(e))
        if hesap == "u":
            try:  # eger sol index disina cikarsa karsi degeri al
                if i - 1 < 0:
                    u_sol = self.u[i + 1, j]
                else:
                    u_sol = self.u[i - 1, j]
            except Exception as e:
                print("Except u_sol {}".format(e))
```

EK-2. (devam) Yer değiştirme modülü

```
        try:  # eger sag index disina cikarsa karsi degeri al
            u_sag = self.u[i + 1, j]
        except Exception as e:
                u_sag = self.u[i - 1, j]


        try:  # eger ust index disina cikarsa karsi degeri al
            u_ust = self.u[i, j + 1]
        except Exception as e:
            u_ust = self.u[i, j - 1]


        try:  # eger alt index disina cikarsa karsi degeri al
            if j - 1 < 0:
                u_alt = self.u[i, j + 1]
            else:
                u_alt = self.u[i, j - 1]
        except Exception as e:
            print("Except u_alt {}".format(e))


    self.sabit_u = (2 * self.deltax * self.deltay ** 2) * (2 * self.mu) * \
                   (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_sag - teta_sol)


    self.sabit_v = (2 * self.deltay * self.deltax ** 2) * (2 * self.mu) * \
                   (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_ust - teta_alt)


    if hesap == 'u':
        if j == 0 or j == self.y - 1:  # kenarlar
```

EK-2. (devam) Yer değiştirme modülü

```
            sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \

                (u_ust + u_alt) + (self.sabit_u / self.payda_u)

        else:  # ic kisimlar

            sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \

                (u_ust + u_alt) + (self.ks_v_i_a_1_j_a_1 / self.payda_u) * \

                (self.v[i + 1, j + 1] -self.v[i - 1, j + 1] - self.v[i + 1, j - 1] + self.v[i - 1, j - 1]) + \

                (self.sabit_u / self.payda_u)


        self.u_cr[i, j] = sayi_u


    # sol, sag, ust kenarlar hesaplanirken

    if hesap == "v":

        if i == 0 or i == self.x - 1  or j == self.y - 1:

            sayi_v = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \

                (v_sag + v_sol) + (self.sabit_v / self.payda_v)

        else:

            sayi_v = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \

                (v_sag + v_sol) + (self.ks_u_i_a_1_j_a_1 / self.payda_v) * \

                (self.u[i + 1, j + 1] - self.u[i - 1, j + 1] - self.u[i + 1, j - 1] + self.u[i - 1, j - 1]) + \

                (self.sabit_v / self.payda_v)


        self.v_cr[i, j] = sayi_v
```

EK-2. (devam) Yer değiştirme modülü

```
        except Exception as e:
            print("yer_degistirme_donusum_d2 i,j: {},{}, hata: {}".format(i, j, e))
            print(e)


    def yer_degistirme_donusum_d3(self, i, j, hesap):
        try:
            try:  # eger sol index disina cikarsa karsi degeri al
                if i - 1 < 0:
                    teta_sol = self.teta[i + 1, j]
                else:
                    teta_sol = self.teta[i - 1, j]
            except Exception as e:
                print("Except teta_sol {}".format(e))


            try:  # eger sag index disina cikarsa karsi degeri al
                teta_sag = self.teta[i + 1, j]
            except Exception as e:
                teta_sag = self.teta[i - 1, j]


            try:  # eger ust index disina cikarsa karsi degeri al
                teta_ust = self.teta[i, j + 1]
            except Exception as e:
                teta_ust = self.teta[i, j - 1]


            try:  # eger alt index disina cikarsa karsi degeri al
                if j - 1 < 0:
```

EK-2. (devam) Yer değiştirme modülü

```
            teta_alt = self.teta[i, j + 1]
        else:
            teta_alt = self.teta[i, j - 1]
    except Exception as e:
        print("Except teta_alt {}".format(e))


    # bu problemde U'nun tum kenarlari sabit oldugundan donusum sadece V icin yapilir
    if hesap == "v":
        try:  # eger sol index disina cikarsa karsi degeri al
            if i - 1 < 0:
                v_sol = self.v[i + 1, j]
            else:
                v_sol = self.v[i - 1, j]
        except Exception as e:
            print("Except v_sol {}".format(e))


        try:  # eger sag index disina cikarsa karsi degeri al
            v_sag = self.v[i + 1, j]
        except Exception as e:
            v_sag = self.v[i - 1, j]


        try:  # eger ust index disina cikarsa karsi degeri al
            v_ust = self.v[i, j + 1]
        except Exception as e:
            v_ust = self.v[i, j - 1] + (self.alfa * (3 * self.lamda + 2 * self.mu) *
```

EK-2. (devam) Yer değiştirme modülü

```
                                    (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)

        try:  # eger alt index disina cikarsa karsi degeri al
            if j - 1 < 0:
                v_alt = self.v[i, j + 1] - (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                            (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
            else:
                v_alt = self.v[i, j - 1]
        except Exception as e:
            print("Except v_alt {}".format(e))
    if hesap == "u":
        try:  # eger sol index disina cikarsa karsi degeri al
            if i - 1 < 0:
                u_sol = self.u[i + 1, j]
            else:
                u_sol = self.u[i - 1, j]
        except Exception as e:
            print("Except u_sol {}".format(e))

        try:  # eger sag index disina cikarsa karsi degeri al
            u_sag = self.u[i + 1, j]
        except Exception as e:
            u_sag = self.u[i - 1, j]

        try:  # eger ust index disina cikarsa karsi degeri al
            u_ust = self.u[i, j + 1]
```

EK-2. (devam) Yer değiştirme modülü

```
            except Exception as e:
                u_ust = self.u[i, j - 1]

            try:  # eger alt index disina cikarsa karsi degeri al
                if j - 1 < 0:
                    u_alt = self.u[i, j + 1]
                else:
                    u_alt = self.u[i, j - 1]
            except Exception as e:
                print("Except u_alt {}".format(e))

        self.sabit_u = (2 * self.deltax * self.deltay ** 2) * (2 * self.mu) * \
                       (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_sag - teta_sol)

        self.sabit_v = (2 * self.deltay * self.deltax ** 2) * (2 * self.mu) * \
                       (3 * self.lamda + 2 * self.mu) * self.alfa * (teta_ust - teta_alt)

        if hesap == 'u':
            if j == 0 or j == self.y - 1:  # kenarlar
                sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \
                                                                (u_ust + u_alt) + (
                         self.sabit_u / self.payda_u)
            else:  # ic kisimlar
                sayi_u = (self.ks_u_i_a_1_j / self.payda_u) * (u_sag + u_sol) + (self.ks_u_i_j_a_1 / self.payda_u) * \
                         (u_ust + u_alt) + (self.ks_v_i_a_1_j_a_1 / self.payda_u) * \
```

EK-2. (devam) Yer değiştirme modülü

```
                (self.v[i + 1, j + 1] - self.v[i - 1, j + 1] - self.v[i + 1, j - 1] + self.v[i - 1, j - 1]) + \
                (self.sabit_u / self.payda_u)

            self.u_cr[i, j] = sayi_u

        # sol, sag, ust kenarlar hesaplanirken
        if hesap == "v":
            if i == 0 or i == self.x - 1 or j == self.y - 1:
                sayi_u = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \
                (v_sag + v_sol) + (self.sabit_v / self.payda_v)
            else:
                sayi_u = (self.ks_v_i_j_a_1 / self.payda_v) * (v_ust + v_alt) + (self.ks_v_i_a_1_j / self.payda_v) * \
                (v_sag + v_sol) + (self.ks_u_i_a_1_j_a_1 / self.payda_v) * \
                (self.u[i + 1, j + 1] - self.u[i - 1, j + 1] - self.u[i + 1, j - 1] + self.u[i - 1, j - 1]) +\
                (self.sabit_v / self.payda_v)

            self.v_cr[i, j] = sayi_u

    except Exception as e:
        print("yer_degistirme_donusum_d3 i,j: {},{}, hata: {}".format(i, j, e))
        print(e)
```

EK-3. Gerilme modülü

```
import numpy as np
import matplotlib.pyplot as plt


class GerilmeAnalizi:
    x = 0
    y = 0
    lx = 0
    ly = 0
    teta = None
    u = None
    v = None
    lamda = 0.0
    mu = 0.0
    alfa = 0.0
    deltax = 0.0
    deltay = 0.0
    # gerilme matrisi
    xx = None
    yy = None
    xy = None
    # Kenarlarin sabit olup olmama durumu: sabit 1, serbest 0
    k_u_sol = 0
    k_u_sag = 0
    k_u_ust = 0
    k_u_alt = 0
    k_v_sol = 0
```

EK-3. (devam) Gerilme modülü

```
    k_v_sag = 0
    k_v_ust = 0
    k_v_alt = 0


    def __init__(self, x, y, lx, ly, teta, u, v, lamda, mu, alfa,
                 k_u_sol, k_u_sag, k_u_ust, k_u_alt, k_v_sol, k_v_sag, k_v_ust, k_v_alt):
        self.x = x
        self.y = y
        self.lx = lx
        self.ly = ly
        self.teta = teta
        self.u = u
        self.v = v
        self.lamda = lamda
        self.mu = mu
        self.alfa = alfa
        self.deltax = float(self.lx / (self.x - 1))
        self.deltay = float(self.ly / (self.y - 1))
        self.k_u_sol = k_u_sol
        self.k_u_sag = k_u_sag
        self.k_u_ust = k_u_ust
        self.k_u_alt = k_u_alt
        self.k_v_sol = k_v_sol
        self.k_v_sag = k_v_sag
        self.k_v_ust = k_v_ust
        self.k_v_alt = k_v_alt
```

EK-3. (devam) Gerilme modülü

```
    def grafik_ciz(self, grafik_isim):
        fig, fx = plt.subplots(1, 1)
        fig = plt.gcf() # grafik penceresine isim verebilmek icin
        x_ax, y_ax = np.meshgrid(np.linspace(0, self.lx, self.x), np.linspace(0, self.ly, self.y))
        fx.set_title(grafik_isim + " Gerilme Dağılımı")
        fx.set_xticks(np.linspace(0, self.lx, self.x))
        fx.set_yticks(np.linspace(0, self.ly, self.y))
        fx.set_xlabel("x")
        fx.set_ylabel("y")
        fx.grid(True)
        if grafik_isim == "xx":
            color_bar_data = np.linspace(round(np.nanmin(self.xx), 2),round(np.nanmax(self.xx), 2) + 0.01, 10, endpoint=True)
            color_bar_x = fx.contourf(x_ax, y_ax, np.round(self.xx.transpose(), decimals=2), color_bar_data, cmap=plt.cm.jet)
            fig.colorbar(color_bar_x, ticks=color_bar_data, ax=fx)
        elif grafik_isim == "yy":
            color_bar_data = np.linspace(round(np.nanmin(self.yy), 2), round(np.nanmax(self.yy), 2) + 0.01, 10, endpoint=True)
            color_bar_x = fx.contourf(x_ax, y_ax, np.round(self.yy.transpose(), decimals=2), color_bar_data, cmap=plt.cm.jet)
            fig.colorbar(color_bar_x, ticks=color_bar_data, ax=fx)
        else:
            color_bar_data = np.linspace(round(np.nanmin(self.xy), 2), round(np.nanmax(self.xy), 2) + 0.01, 10, endpoint=True)
            color_bar_x = fx.contourf(x_ax, y_ax, np.round(self.xy.transpose(), decimals=2), color_bar_data, cmap=plt.cm.jet)
            fig.colorbar(color_bar_x, ticks=color_bar_data, ax=fx)
        fig.canvas.set_window_title("Gerilme")
```

EK-3. (devam) Gerilme modülü

```
        plt.show()


    def xx_gerilme_hesapla(self):
        self.xx = np.zeros((self.x, self.y), dtype='float')
        # Durum (0): Tum kenarlar sabit
        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 1 and self.k_u_alt == 1 \
                and self.k_v_sol == 1 and self.k_v_sag == 1 and self.k_v_ust == 1 and self.k_v_alt == 1:
            for i in range(1, self.x - 1):
                for j in range(1, self.y - 1):
                    self.xx[i, j] = ((2 * self.mu * self.lamda + 2 * self.mu ** 2) / ((self.lamda + 2 * self.mu) * self.deltax)) * (
                            self.u[i + 1, j] - self.u[i - 1, j]) + \
                            ((2 * self.mu * self.lamda) / ((self.lamda + 2 * self.mu) * 2 * self.deltay)) * (
                            self.v[i, j + 1] - self.v[i, j - 1]) - \
                            ((2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) * \
                            (self.teta[i, j])
            # gerilme hesabinda kullanilan yer degistirme sinir sarti belli olmadigi icin
            for j in range(1, self.y -1 ):
                self.xx[0, j] = self.xx[1, j] + (self.xx[1, j] - self.xx[2, j])
            for j in range(1, self.y -1 ):
                self.xx[self.x - 1, j] = self.xx[self.x - 2, j] + (self.xx[self.x - 2, j] - self.xx[self.x - 3, j])
            for i in range(1, self.x - 1):
                self.xx[i, 0] = self.xx[i, 1] + (self.xx[i, 1] - self.xx[i, 2])
            for i in range(1, self.x - 1):
```

EK-3. (devam) Gerilme modülü

```
            self.xx[i, self.y - 1] = self.xx[i, self.y - 2] + (self.xx[i, self.y - 2] - self.xx[i, self.y
- 3])



        self.xx[0, 0] =  ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 *
self.lamda + 2 * self.mu) * (self.teta[0, 0])

        self.xx[0, self.y - 1] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 *
self.lamda + 2 * self.mu) * (self.teta[0, self.y - 1])

        self.xx[self.x - 1, 0] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 *
self.lamda + 2 * self.mu) * (self.teta[self.x - 1, 0])

        self.xx[self.x - 1, self.y - 1] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa
* (3 * self.lamda + 2 * self.mu) * (self.teta[self.x - 1, self.y - 1])

    # Durum (1)

    # U: sol sağ sabit, ust alt hareketli

    # V: alt üst sabit, ust alt hareketli

    if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt ==
0 \

        and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 1 and self.k_v_alt
== 1:

        for i in range(1, self.x - 1):

            for j in range(1, self.y - 1):

                u_sag = self.u[i + 1, j]

                u_sol = self.u[i - 1, j]

                v_alt = self.v[i, j - 1]

                v_ust = self.v[i, j + 1]

                teta = self.teta[i, j]

                if not (i == 0 or i == self.x - 1 or j==0 or j==self.y - 1):

                    self.xx[i, j] = ((2 * self.mu * self.lamda + 2 * self.mu**2) / ((self.lamda + 2
* self.mu) * self.deltax)) * \

                                    (u_sag - u_sol) + ((2 * self.mu * self.lamda) / ((self.lamda + 2 *
self.mu) * 2 * self.deltay)) *\
```

EK-3. (devam) Gerilme modülü

```
                    (v_ust - v_alt) - (((2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa *
                    (3 * self.lamda + 2 * self.mu) * teta)
        # gerilme hesabinda kullanilan yer degistirme sinir sarti belli olmadigi icin
        for j in range(1, self.y - 1):
            self.xx[0, j] = self.xx[1, j] + (self.xx[1, j] - self.xx[2, j])
        for j in range(1, self.y - 1):
            self.xx[self.x - 1, j] = self.xx[self.x - 2, j] +\
            (self.xx[self.x - 2, j] - self.xx[self.x - 3, j])
        for i in range(1, self.x - 1):
            self.xx[i, 0] = self.xx[i, 1] + (self.xx[i, 1] - self.xx[i, 2])
        for i in range(1, self.x - 1):
            self.xx[i, self.y - 1] = self.xx[i, self.y - 2] + \
            (self.xx[i, self.y - 2] - self.xx[i, self.y - 3])
        self.xx[0, 0] = self.xx[0, 1]
        self.xx[0, self.y - 1] = self.xx[0, self.y -2]
        self.xx[self.x - 1, 0] = self.xx[self.x - 1, 1]
        self.xx[self.x - 1, self.y - 1] = self.xx[self.x - 1, self.y - 2]
    # Durum (2): U sag-sol ve V alt sabit
    if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt == 0 \
        and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 0 and self.k_v_alt == 1:
        for i in range(0, self.x):
            for j in range(0, self.y):
                try:
                    try:  # eger ust index disina cikarsa karsi degeri al
```

EK-3. (devam) Gerilme modülü

```
            v_ust = self.v[i, j + 1]
        except Exception as e:
            v_ust = self.v[i, j - 1] + (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                        (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
        try:  # eger alt index disina cikarsa karsi degeri al
            if j - 1 < 0:
                v_alt = self.v[i, j + 1] - (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                            (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
            else:
                v_alt = self.v[i, j - 1]
        except Exception as e:
            print("Except v_alt: i,j {},{} {}".format(i, j, e))

        try:  # eger sol index disina cikarsa karsi degeri al
            if i - 1 < 0:
                u_sol = self.u[i + 1, j]
            else:
                u_sol = self.u[i - 1, j]
        except Exception as e:
            print("Except u_sol: i,j {},{} {}".format(i, j, e))
        try:  # eger sag index disina cikarsa karsi degeri al
            u_sag = self.u[i + 1, j]
        except Exception as e:
            u_sag = self.u[i - 1, j]
```

EK-3. (devam) Gerilme modülü

```
                    self.xx[i, j] = ((2 * self.mu * self.lamda + 2 * self.mu**2) / ((self.lamda + 2
* self.mu) * self.deltax)) * \

                            (u_sag - u_sol) + ((2 * self.mu * self.lamda ) / ((self.lamda + 2 *
self.mu) * 2 * self.deltay)) * \

                             (v_ust - v_alt) - ((2 * self.mu) / (self.lamda + 2 * self.mu)) *\

                             self.alfa * (3 * self.lamda + 2 * self.mu) * (self.teta[i, j])

                except Exception as e:

                    print("xx_gerilme_hesapla i,j: {},{}: {}".format(i, j, e))


        return self.xx # arayuzde yazdirmak icin self.xx i aliriz


    def yy_gerilme_hesapla(self):

        self.yy = np.zeros((self.x, self.y), dtype='float')

        # Durum (0): Tum kenarlar sabit

        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 1 and self.k_u_alt ==
1 \

                and self.k_v_sol == 1 and self.k_v_sag == 1 and self.k_v_ust == 1 and
self.k_v_alt == 1:

            for i in range(1, self.x - 1):

                for j in range(1, self.y - 1):

                    self.yy[i, j] = (((2 * self.mu * self.lamda + 2 * self.mu ** 2) / (

                    (self.lamda + 2 * self.mu) * self.deltay)) * ( self.v[i, j + 1]- self.v[i, j - 1] )) + \

                            (((2 * self.mu * self.lamda) / ((self.lamda + 2 * self.mu) * 2 * self.deltax))
* (

                                self.u[i + 1, j]- self.u[i - 1, j] )) - \

                            (((2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (

                            3 * self.lamda + 2 * self.mu) * self.teta[i, j])

            # gerilme hesabinda kullanilan yer degistirme sinir sarti belli olmadigi icin
```

EK-3. (devam) Gerilme modülü

```
        for j in range(1, self.y -1):
            self.yy[0, j] = self.yy[1, j] + (self.yy[1, j] - self.yy[2, j])
        for j in range(1, self.y - 1):
            self.yy[self.x - 1, j] = self.yy[self.x - 2, j] + \
            (self.yy[self.x - 2, j] - self.yy[self.x - 3, j])
        for i in range(1, self.x - 1):
            self.yy[i, 0] = self.yy[i, 1] + (self.yy[i, 1] - self.yy[i, 2])
        for i in range(1, self.x - 1):
            self.yy[i, self.y - 1] = self.yy[i, self.y - 2] + \
            (self.yy[i, self.y - 2] - self.yy[i, self.y - 3])
        self.yy[0, 0] =  ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) * (self.teta[0, 0])
        self.yy[0, self.y - 1] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) * (self.teta[0, self.y - 1])
        self.yy[self.x - 1, 0] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) * (self.teta[self.x - 1, 0])
        self.yy[self.x - 1, self.y - 1] = ((-2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) *\
                                    (self.teta[self.x - 1, self.y - 1])
    # Durum (1):
    # U: sol sağ sabit, ust alt hareketli
    # V: alt üst sabit, ust alt hareketli
    if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt == 0 \
        and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 1 and self.k_v_alt == 1:
        for i in range(1, self.x - 1 ):
            for j in range(1, self.y - 1):
                try:
```

EK-3. (devam) Gerilme modülü

```
                v_ust = self.v[i, j + 1]
                v_alt = self.v[i, j - 1]
                u_sag = self.u[i + 1, j]
                u_sol = self.u[i - 1, j]
                if not (i == 0 or i == self.x - 1 or j==0 or j==self.y - 1):
                    sayi = (((2 * self.mu * self.lamda + 2 * self.mu**2) / ((self.lamda + 2 * self.mu) * self.deltay)) * (v_ust - v_alt)) + \
                           (((2 * self.mu * self.lamda )/ ((self.lamda + 2 * self.mu) * 2 * self.deltax)) * (u_sag - u_sol)) - \
                           (((2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa * (3 * self.lamda + 2 * self.mu) * self.teta[i, j])
                    self.yy[i, j] = sayi
            except Exception as e:
                print("yy_gerilme_hesapla i,j: {},{}: {}".format(i, j, e))
        # gerilme hesabinda kullanilan yer degistirme sinir sarti belli olmadigi icin
        for j in range(1, self.y - 1):
            self.yy[0, j] = self.yy[1, j] + (self.yy[1, j] - self.yy[2, j])
        for j in range(1, self.y - 1):
            self.yy[self.x - 1, j] = self.yy[self.x - 2, j] + (self.yy[self.x - 2, j] - self.yy[self.x - 3, j])
        for i in range(1, self.x - 1):
            self.yy[i, 0] = self.yy[i, 1] + (self.yy[i, 1] - self.yy[i, 2])
        for i in range(1, self.x - 1):
            self.yy[i, self.y - 1] = self.yy[i, self.y - 2] + (self.yy[i, self.y - 2] - self.yy[i, self.y - 3])
        self.yy[0, 0] = self.yy[0, 1]
        self.yy[0, self.y - 1] = self.yy[0, self.y - 2]
        self.yy[self.x - 1, 0] = self.yy[self.x - 1, 1]
```

EK-3. (devam) Gerilme modülü

```
            self.yy[self.x - 1, self.y - 1] = self.yy[self.x - 1, self.y - 2]


        # Durum (2) : U sag-sol ve V alt sabit
        if self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt ==
0 \
            and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 0 and self.k_v_alt
== 1:
            for i in range(0, self.x ):
                for j in range(0, self.y):
                    try:
                        try:  # eger ust index disina cikarsa karsi degeri al
                            v_ust = self.v[i, j + 1]
                        except Exception as e:
                            v_ust = self.v[i, j - 1] + (self.alfa * (3 * self.lamda + 2 * self.mu) * \
                                    (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
                        try:  # eger alt index disina cikarsa karsi degeri al
                            if j - 1 < 0:
                                v_alt = self.v[i, j + 1] - (self.alfa * (3 * self.lamda + 2 * self.mu) *
                                        (self.teta[i, j]) * self.deltay) / (self.lamda + self.mu)
                            else:
                                v_alt = self.v[i, j - 1]
                        except Exception as e:
                            print("Except v_alt: i,j {},{} {}".format(i, j, e))


                        try:  # eger sol index disina cikarsa karsi degeri al
                            if i - 1 < 0:
                                u_sol = self.u[i + 1, j]
```

EK-3. (devam) Gerilme modülü

```
                else:
                    u_sol = self.u[i - 1, j]
            except Exception as e:
                print("Except u_sol: i,j {},{} {}".format(i, j, e))
            try:  # eger sag index disina cikarsa karsi degeri al
                u_sag = self.u[i + 1, j]
            except Exception as e:
                u_sag = self.u[i - 1, j]

            self.yy[i, j] = (((2 * self.mu * self.lamda + 2 * self.mu ** 2) / (
                (self.lamda + 2 * self.mu) * self.deltay)) * (v_ust - v_alt)) + \
                (((2 * self.mu * self.lamda) / ((self.lamda + 2 * self.mu) * 2 * self.deltax)) *
                (u_sag - u_sol)) - (((2 * self.mu) / (self.lamda + 2 * self.mu)) * self.alfa *
                 (3 * self.lamda + 2 * self.mu) * self.teta[i, j])
        except Exception as e:
            print("yy_gerilme_hesapla i,j: {},{}: {}".format(i, j, e))
    return self.yy


def xy_gerilme_hesapla(self):
    self.xy = np.zeros((self.x, self.y), dtype='float')
    # Durum (0, 1, 2):
    if (self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt == 0 \
            and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 1 and self.k_v_alt == 1) or \
```

EK-3. (devam) Gerilme modülü

```
        (self.k_u_sol == 1 and self.k_u_sag == 1 and self.k_u_ust == 0 and self.k_u_alt == 0 \
          and self.k_v_sol == 0 and self.k_v_sag == 0 and self.k_v_ust == 0 and self.k_v_alt == 1):
        for i in range(1, self.x - 1):
            for j in range(1, self.y - 1):
                try:
                    self.xy[i, j] = self.mu * (( self.u[i, j + 1] - self.u[i, j - 1]) / ( 2 * self.deltay)) + \
                                ((self.v[i + 1, j] - self.v[i - 1, j]) / (2 * self.deltax))
                except Exception as e:
                    print("xy_gerilme_hesapla i,j: {},{}: {}".format(i, j, e))
    return self.xy
```

EK-4. Ara yüz modülü

```
from tkinter import *
from tkinter import messagebox
import tkinter as tk

import sicaklik
import yer_degistirme
import gerilme

class Arayuz:
    # image label, levha goruntusunu tutar
    lblImg = None
    lbl_img_data = “” # mavi levha goruntusu
    bgcolor = “#BAC7CD” # ana form icin arka plan rengi
    lblbgcolor = “#00A2E8” # label’ler icin arka plan rengi (levha rengi ile ayni olmasi icin)
    x = 0
    y = 0
    lx = 0
    ly = 0
    # sicaklik verileri
    t_sol = 0.0
    t_sag = 0.0
    t_ust = 0.0
    t_alt = 0.0
    t_ortam = 0.0
    # veri degiskenleri
    teta = None
```

EK-4. (devam) Ara yüz modülü

```
    sicaklik_mat_t_son = None
    u = None
    v = None
    em = 0.0
    nu = 0.0
    alfa = 0.0
    lamda = 0.0
    mu = 0.0
    sinir_sarti_noktalari = {}

    # iki farkli yerde kullanilir
    dep = None

    # gerilme degerleri matrisleri
    xx = None
    yy = None
    xy = None

    # UI elements
    mw = None # main window
    lblGeo = None
    lblMlz = None
    lblButonCizgi = None
    lbl_x = None
    txt_x = None
    lbl_y = None
```

EK-4. (devam) Ara yüz modülü

```
    txt_y = None
    lblLx = None
    txt_lx = None
    lblLy = None
    txt_ly = None
    lblDeltaX = None
    lblDeltaY = None
    lblEm = None
    txtEm = None
    lblNu = None
    txtNu = None
    lblAlfa = None
    txtAlfa = None
    lbl_sinir_sarti = None
    txt_sinir_sarti = None
    lbl_sinir_sarti_goster = None
    # sicaklik icin arayuz textbox ve label nesneleri
    lblSiSol = None
    txtSiSol = None
    lblSiSag = None
    txtSiSag = None
    lblSiUst = None
    txtSiUst = None
    lblSiAlt = None
    txtSiAlt = None
    txlSiOrt = None
```

EK-4. (devam) Ara yüz modülü

```
    # sabit noktalar icin checkbox nesneleri
    cb_u_sinir = None
    cb_v_sinir = None
    # sabit noktalar icin checkbox nesneleri icin deger tutacak degiskenler
    u_sinir = None
    v_sinir = None
    #
    # checkbox degerleri
    k_varsol = None
    k_varsag = None
    k_varust = None
    k_varalt = None
    # checkbox nesneler
    cb_sol = None
    cb_sag = None
    cb_ust = None
    cb_alt = None
    # checkbox atamalar
    k_sol = 1
    k_sag = 1
    k_ust = 1
    k_alt = 1
    # U ve V icin checkbox, deger ve atama degiskenleri
    # -- chk degerleri
    k_u_varsol = None
    k_u_varsag = None
```

EK-4. (devam) Ara yüz modülü

```
    k_u_varust = None
    k_u_varalt = None
    k_v_varsol = None
    k_v_varsag = None
    k_v_varust = None
    k_v_varalt = None
    # chkbox nesneleri
    cb_u_sol = None
    cb_u_sag = None
    cb_u_ust = None
    cb_u_alt = None
    cb_v_sol = None
    cb_v_sag = None
    cb_v_ust = None
    cb_v_alt = None
    # ---- chk degelerinin atanacagi local degiskenler
    k_u_sol = 1
    k_u_sag = 1
    k_u_ust = 1
    k_u_alt = 1
    k_v_sol = 1
    k_v_sag = 1
    k_v_ust = 1
    k_v_alt = 1
    # form menusu
    menu = None
```

EK-4. (devam) Ara yüz modülü

```
    # buttonlar
    btn_sicaklik_hesapla = None
    btn_yer_degistirme_hesapla = None
    btn_gerilme_hesapla_xx = None
    btn_gerilme_hesapla_yy = None
    btn_gerilme_hesapla_xy = None
    btn_sabit_noktalar_ekle = None
    btn_sabit_noktalar_cikar = None

    def deger_al(self):
        # Arayuzdeki degerler bu fonksiyon ile degiskenlere atanir
        try:
            self.x = int(self.txt_x.get())
            self.y = int(self.txt_y.get())
            self.lx = float(self.txt_lx.get())
            self.ly = float(self.txt_ly.get())
            self.lblDeltaX["text"] = "   delta x: " + str(round(int(self.lx) / (int(self.x) - 1), 4))
            self.lblDeltaY["text"] = "   delta y: " + str(round(int(self.ly) / (int(self.y) - 1), 4))
            # sicakliklar icin checkbox atamalari
            self.k_sol = self.k_varsol.get()
            self.k_sag = self.k_varsag.get()
            self.k_ust = self.k_varust.get()
            self.k_alt = self.k_varalt.get()

            # sicaklik degerleri atamalari
            # sol kenar izole secilmisse
```

EK-4. (devam) Ara yüz modülü

```
        if self.k_sol == 0:
            self.t_sol = float(self.txtSiOrt.get())
        else:
            self.t_sol = float(self.txtSiSol.get())

        # sag kenar izole secilmisse
        if self.k_sag == 0:
            self.t_sag = float(self.txtSiOrt.get())
        else:
            self.t_sag = float(self.txtSiSag.get())

        # ust kenar izole secilmisse
        if self.k_ust == 0:
            self.t_ust = float(self.txtSiOrt.get())
        else:
            self.t_ust = float(self.txtSiUst.get())

        # alt kenar izole secilmisse
        if self.k_alt == 0:
            self.t_alt = float(self.txtSiOrt.get())
        else:
            self.t_alt = float(self.txtSiAlt.get())

        self.t_ortam = float(self.txtSiOrt.get())
        # U ve V checkbox atamalari
        self.k_u_sol = self.k_u_varsol.get()
```

EK-4. (devam) Ara yüz modülü

```
            self.k_u_sag = self.k_u_varsag.get()
            self.k_u_ust = self.k_u_varust.get()
            self.k_u_alt = self.k_u_varalt.get()
            self.k_v_sol = self.k_v_varsol.get()
            self.k_v_sag = self.k_v_varsag.get()
            self.k_v_ust = self.k_v_varust.get()
            self.k_v_alt = self.k_v_varalt.get()
            self.em = float(eval(str(self.txtEm.get()).replace(“,”, “.”)))
            self.nu = float(eval(self.txtNu.get()))
            self.alfa = float(eval(self.txtAlfa.get()))
            # form load sirasinda deger_al fonksiyonu cagrilir, biz de burada yararlanalim.
            # duruma gore gizlenmesi gereken txtbox varsa gizlensin.
            self.cbSol()
            self.cbSag()
            self.cbUst()
            self.cbAlt()
            return True
        except Exception as e:
            messagebox.showerror(“Hata”, “Verileri kontrol ediniz.\n\n{}”.format(e))
            return False

    def excel_print(self, matrix, matris_adi):
        # excel ciktisi formatinda yazar
        line = ““
        for i in range(0, self.x):
            for j in range(0, self.y):
```

EK-4. (devam) Ara yüz modülü

```
            line += matris_adi + "({},{})\t{}\n".format(i, j, str(matrix[i][j]).replace(".", ","))
        return line


    def sonuc_goster(self, sonuc):
        # Hangi buton cagirirsa ona gore sonuclari konsole yazar
        if sonuc == "sicaklik":
            print("Sicaklik")
            print(self.excel_print(self.sicaklik_mat_t_son, "T"))
        elif sonuc == "deplasman":
            print("u")
            print(self.excel_print(self.u, "u"))
            print("v")
            print(self.excel_print(self.v, "v"))
        elif sonuc == "gerilme_xx":
            print("gerilme xx")
            print(self.excel_print(self.xx, "gxx"))
        elif sonuc == "gerilme_yy":
            print("gerilme yy")
            print(self.excel_print(self.yy, "gyy"))
        elif sonuc == "gerilme_xy":
            print("gerilme xy")
            print(self.excel_print(self.xy, "gxy"))


    def arayuz_olustur(self):
        self.mw = Tk()
        self.mw.title('Isıl Gerilme Analizi - Melike Korkmaz')
```

EK-4. (devam) Ara yüz modülü

```
self.mw.geometry("941x335")
self.mw.resizable(0, 0)
self.mw.configure(background=self.bgcolor)
self.mw.after(10, self.deger_al)
#
windowWidth = self.mw.winfo_reqwidth()
windowHeight = self.mw.winfo_reqheight()
#
positionRight = int(self.mw.winfo_screenwidth() / 5 - windowWidth / 2)
positionDown = int(self.mw.winfo_screenheight() / 3 - windowHeight / 2)
#
self.mw.geometry("+{}+{}".format(positionRight, positionDown))
#
self.menu = tk.Menu()
self.menu.add_command(label="Çıkış", command=self.mw.destroy)
self.menu.add_command(label="Hakkında", command=self.hakkinda)
self.mw.config(menu=self.menu)
#
self.lblGeo = Label(master=self.mw, text="Geometri Bilgisi")
self.lblMlz = Label(master=self.mw, text="Malzeme Bilgisi")
self.lblGeo.configure(bg=self.bgcolor)
self.lblMlz.configure(bg=self.bgcolor)
#
# Geometri ozellikleri
self.lbl_x = Label(master=self.mw, text="x", padx=11)
self.lbl_x.configure(bg=self.bgcolor)
```

EK-4. (devam) Ara yüz modülü

```
self.txt_x = Entry(master=self.mw, width=7)
self.txt_x.insert(tk.END, “21”)
self.lbl_y = Label(master=self.mw, text=“y”, padx=11)
self.lbl_y.configure(bg=self.bgcolor)
self.txt_y = Entry(master=self.mw, width=7)
self.txt_y.insert(tk.END, “11”)
self.lblLx = Label(master=self.mw, text=“lx”, padx=11)
self.lblLx.configure(bg=self.bgcolor)
self.txt_lx = Entry(master=self.mw, width=7)
self.txt_lx.insert(tk.END, “2000”)
self.lblLy = Label(master=self.mw, text=“ly”, padx=11)
self.lblLy.configure(bg=self.bgcolor)
self.txt_ly = Entry(master=self.mw, width=7)
self.txt_ly.insert(tk.END, “1000”)
self.lblDeltaX = Label(master=self.mw, text=“   delta x: -”)
self.lblDeltaY = Label(master=self.mw, text=“   delta y: -”)
self.lblDeltaX.configure(bg=self.bgcolor)
self.lblDeltaY.configure(bg=self.bgcolor)
#
# Malzeme ozellikleri
self.lblEm = Label(master=self.mw, text=“Elas. Mod.”, padx=5)
self.lblEm.configure(bg=self.bgcolor)
self.txtEm = Entry(master=self.mw, width=12)
self.txtEm.insert(tk.END, “2 * 10**5”)
self.lblAlfa = Label(master=self.mw, text=“Alfa”, padx=5)
self.lblAlfa.configure(bg=self.bgcolor)
```

EK-4. (devam) Ara yüz modülü

```
self.txtAlfa = Entry(master=self.mw, width=12)
self.txtAlfa.insert(tk.END, “1.3 * 10**-5”)

self.lblNu = Label(master=self.mw, text=“Nu”, padx=5)
self.lblNu.configure(bg=self.bgcolor)
self.txtNu = Entry(master=self.mw, width=12)
self.txtNu.insert(tk.END, “0.3”)

self.lblGeo.config(font=(“, 12, ‘bold’))
self.lblGeo.grid(row=0, column=0, columnspan=2, sticky=‘we’)
self.lbl_x.grid(row=1, column=0)
self.txt_x.grid(row=1, column=1, sticky=‘w’)
self.lbl_y.grid(row=2, column=0)
self.txt_y.grid(row=2, column=1, sticky=‘w’)
self.lblLx.grid(row=3, column=0)
self.txt_lx.grid(row=3, column=1, sticky=‘w’)
self.lblLy.grid(row=4, column=0)
self.txt_ly.grid(row=4, column=1, sticky=‘w’)
self.lblDeltaX.grid(row=5, column=0, columnspan=2, sticky=‘w’)
self.lblDeltaY.grid(row=6, column=0, columnspan=2, sticky=‘w’)

self.lblMlz.config(font=(“, 12, ‘bold’))
self.lblMlz.grid(row=0, column=9, columnspan=2, sticky=‘we’)
self.lblEm.grid(row=1, column=9, sticky=‘e’)
self.txtEm.grid(row=1, column=10, sticky=‘w’)
self.lblAlfa.grid(row=2, column=9, sticky=‘e’)
```

EK-4. (devam) Ara yüz modülü

```
        self.txtAlfa.grid(row=2, column=10, sticky='w')

        self.lblNu.grid(row=3, column=9, sticky='e')

        self.txtNu.grid(row=3, column=10, sticky='w')

        # *

        self.lbl_sinir_sarti = Label(master=self.mw, text="Özel Sabit Nokta")

        self.lbl_sinir_sarti.configure(bg=self.bgcolor)

        self.lbl_sinir_sarti.config(font=('', 12, 'bold'))

        self.lbl_sinir_sarti.grid(row=4, column=9, columnspan=2, sticky='we')

        # *

        self.u_sinir = IntVar(value=1)

        self.cb_u_sinir = Checkbutton(master=self.mw, bg=self.bgcolor, onvalue=1,
offvalue=0, text="u", variable=self.u_sinir)

        self.cb_u_sinir.grid(row=5, column=9, sticky='e')

        # *

        self.v_sinir = IntVar(value=0)

        self.cb_v_sinir = Checkbutton(master=self.mw, bg=self.bgcolor, onvalue=1,
offvalue=0, text="v", variable=self.v_sinir)

        self.cb_v_sinir.grid(row=5, column=10, sticky='w', padx=8)

        # *

        self.btn_sabit_noktalar_ekle = Button(master=self.mw, text='Ekle',
command=lambda: self.sabit_nokta_tanimlari("ekle"))

        self.btn_sabit_noktalar_ekle.grid(row=6, column=9, sticky='e', padx=2)

        # *

        self.btn_sabit_noktalar_cikar = Button(master=self.mw, text='Çıkar',
command=lambda: self.sabit_nokta_tanimlari("cikar"))

        self.btn_sabit_noktalar_cikar.grid(row=6, column=11, sticky='w', padx=2)

        # *
```

EK-4. (devam) Ara yüz modülü

```
        self.txt_sinir_sarti = Entry(master=self.mw, width=12)

        self.txt_sinir_sarti.insert(tk.END, “x,y”)

        self.txt_sinir_sarti.grid(row=6, column=10, sticky=‘w’)

        # *

        self.lbl_sinir_sarti_goster = Label(master=self.mw, text=““)

        self.lbl_sinir_sarti_goster.grid(row=7, column=9, columnspan=2, rowspan=2, sticky=‘w’)

        self.lbl_sinir_sarti_goster.configure(bg=self.bgcolor)


        img_data = PhotoImage(data=self.lbl_img_data).subsample(1, 2)

        self.lblImg = Label(master=self.mw, image=img_data, border=0, bg=self.bgcolor).grid(row=2, column=3, columnspan=6, rowspan=7)

        self.txtSiSol = Entry(master=self.mw, width=6)

        self.txtSiSol.insert(tk.END, “20”)

        self.txtSiSag = Entry(master=self.mw, width=6)

        self.txtSiSag.insert(tk.END, “50”)

        self.txtSiUst = Entry(master=self.mw, width=6)

        self.txtSiUst.insert(tk.END, “30”)

        self.txtSiAlt = Entry(master=self.mw, width=6)

        self.txtSiAlt.insert(tk.END, “30”)


        # checkboxlar

        self.k_varsol = IntVar(value=1)

        self.k_varsag = IntVar(value=1)

        self.k_varust = IntVar(value=0)

        self.k_varalt = IntVar(value=0)
```

EK-4. (devam) Ara yüz modülü

```
        self.cb_sol = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=0, offvalue=1, text="izole ", variable=self.k_varsol, command=self.cbSol)

        self.cb_sag = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=0, offvalue=1, text="izole  ", variable=self.k_varsag, command=self.cbSag)

        self.cb_ust = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=0, offvalue=1, text="izole ", variable=self.k_varust, command=self.cbUst)

        self.cb_alt = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=0, offvalue=1, text="izole ", variable=self.k_varalt, command=self.cbAlt)

        #

        # chk lar icin bas deger atamasi

        self.k_u_varsol = IntVar(value=1)

        self.k_u_varsag = IntVar(value=1)

        self.k_u_varust = IntVar(value=1)

        self.k_u_varalt = IntVar(value=1)

        self.k_v_varsol = IntVar(value=1)

        self.k_v_varsag = IntVar(value=1)

        self.k_v_varust = IntVar(value=1)

        self.k_v_varalt = IntVar(value=1)

        # nesneler

        # u

        self.cb_u_sol = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="u sabit", variable=self.k_u_varsol)

        self.cb_u_sag = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="u sabit", variable=self.k_u_varsag)

        self.cb_u_ust = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="u sabit", variable=self.k_u_varust)

        self.cb_u_alt = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="u sabit", variable=self.k_u_varalt)

        # v
```

EK-4. (devam) Ara yüz modülü

```
        self.cb_v_sol = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="v sabit", variable=self.k_v_varsol)

        self.cb_v_sag = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="v sabit", variable=self.k_v_varsag)

        self.cb_v_ust = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="v sabit", variable=self.k_v_varust)

        self.cb_v_alt = Checkbutton(master=self.mw, bg=self.lblbgcolor, onvalue=1, offvalue=0, text="v sabit", variable=self.k_v_varalt)

        #

        # U ve V chkbox yerlestir.

        # u

        self.cb_u_sol.grid(row=2, column=3, sticky='w', padx=3, pady=2)

        self.cb_u_sag.grid(row=2, column=8, sticky='e', padx=3, pady=2)

        self.cb_u_ust.grid(row=2, column=4, sticky='w')

        self.cb_u_alt.grid(row=8,column=4, sticky='w', padx=1, pady=1)

        # v

        self.cb_v_sol.grid(row=3, column=3, sticky='w', padx=4)

        self.cb_v_sag.grid(row=3, column=8, sticky='e', padx=3)

        self.cb_v_ust.grid(row=2, column=5, sticky='w')

        self.cb_v_alt.grid(row=8, column=5, sticky='w')

        #

        # izole checkbox lari yerlestir.

        self.cb_sol.grid(row=4, column=3, sticky="w", padx=3)

        self.cb_sag.grid(row=4, column=8, sticky='e', padx=3)

        self.cb_ust.grid(row=2, column=6, sticky='w')

        self.cb_alt.grid(row=8,column=6, sticky='w')

        #
```

EK-4. (devam) Ara yüz modülü

```
        # Textbox lari yerlestir
        self.txtSiSol.grid(row=5, column=3, sticky='w', padx=3, pady=3)
        self.txtSiSag.grid(row=5, column=8, sticky='e', padx=3, pady=3)
        self.txtSiUst.grid(row=2, column=7, sticky='w')
        self.txtSiAlt.grid(row=8, column=7, sticky='w')
        #
        self.txtSiOrt = Entry(master=self.mw, width=6)
        self.txtSiOrt.insert(tk.END, "0")
        self.txtSiOrt.grid(row=4, column=5, sticky='e')
        #
        self.lblButonCizgi = Label(master=self.mw, text="", width=10, bg=self.bgcolor)
        self.lblButonCizgi.grid(row=11, column=0, columnspan=10, sticky='we')
        #
        # sicaklik butonu
        self.btn_sicaklik_hesapla = Button(master=self.mw, text='Sıcaklık Analizi',
command=self.sicaklik_hesapla)
        self.btn_sicaklik_hesapla.grid(row=12, column=5, padx=2, pady=1)
        #
        # deplasman butonu
        self.btn_yer_degistirme_hesapla = Button(master=self.mw, text='Yer Değiştirme
Analizi',state="disabled", command=self.yer_degistirme_hesapla)
        self.btn_yer_degistirme_hesapla.grid(row=12, column=6, padx=2, pady=1)
        #
        # gerilme butonlari
        self.btn_gerilme_hesapla_xx = Button(master=self.mw, text='Gerilme Analizi xx',
state="disabled", command=self.gerilme_hesapla_xx)
        self.btn_gerilme_hesapla_xx.grid(row=12, column=7, sticky='e', padx=2, pady=1)
```

EK-4. (devam) Ara yüz modülü

```
        #
        self.btn_gerilme_hesapla_yy = Button(master=self.mw, text='Gerilme Analizi yy', state="disabled", command=self.gerilme_hesapla_yy)
        self.btn_gerilme_hesapla_yy.grid(row=13, column=7, sticky='e', padx=2, pady=1)
        #
        self.btn_gerilme_hesapla_xy = Button(master=self.mw, text='Gerilme Analizi xy', state="disabled", command=self.gerilme_hesapla_xy)
        self.btn_gerilme_hesapla_xy.grid(row=14, column=7, sticky='e', padx=2, pady=1)
        #
        # uygulamayi baslat
        self.mw.mainloop()


    def sicaklik_hesapla(self):
        if self.deger_al():
            sic = sicaklik.SicaklikAnalizi(x=self.x, y=self.y, lx=self.lx, ly=self.ly,
                                           t_sol=self.t_sol, t_sag=self.t_sag, t_ust=self.t_ust, t_alt=self.t_alt, t_ortam=self.t_ortam,
                                           k_sol=self.k_sol, k_sag=self.k_sag, k_ust=self.k_ust, k_alt=self.k_alt)
            self.teta, self.sicaklik_mat_t_son = sic.sicaklik_hesapla()
            self.btn_yer_degistirme_hesapla.config(state="active")
            self.sonuc_goster("sicaklik")
            sic.grafik_ciz()


    def yer_degistirme_hesapla(self):
        if self.deger_al():
            self.dep = yer_degistirme.YerDegistirmeAnalizi(x=self.x, y=self.y, lx=self.lx, ly=self.ly,teta=self.teta,
```

EK-4. (devam) Ara yüz modülü

```
                em=self.em, nu=self.nu,
alfa=self.alfa,sabit_noktalar=self.sinir_sarti_noktalari,
                k_u_sol=self.k_u_sol, k_u_sag=self.k_u_sag,
k_u_ust=self.k_u_ust,k_u_alt=self.k_u_alt,
                k_v_sol=self.k_v_sol, k_v_sag=self.k_v_sag,
k_v_ust=self.k_v_ust,k_v_alt=self.k_v_alt
                )
        # mu ve lamda verileri burada atanir.
        self.u, self.v, self.mu, self.lamda = self.dep.yer_degistirme_hesapla()
        self.btn_gerilme_hesapla_xx.config(state="active")
        self.btn_gerilme_hesapla_yy.config(state="active")
        self.btn_gerilme_hesapla_xy.config(state="active")
        self.sonuc_goster("deplasman")
        self.dep.grafik_ciz()

    def gerilme_hesapla_xx(self):
        if self.deger_al():
            ger = gerilme.GerilmeAnalizi(x=self.x, y=self.y, lx=self.lx, ly=self.ly,
teta=self.teta,
                u=self.u, v=self.v, lamda=self.lamda, mu=self.mu, alfa=self.alfa,
                k_u_sol=self.k_u_sol, k_u_sag=self.k_u_sag,
k_u_ust=self.k_u_ust, k_u_alt=self.k_u_alt,
                k_v_sol=self.k_v_sol, k_v_sag=self.k_v_sag,
k_v_ust=self.k_v_ust, k_v_alt=self.k_v_alt
                )
        self.xx = ger.xx_gerilme_hesapla()
        self.sonuc_goster("gerilme_xx")
        ger.grafik_ciz("xx")
```

EK-4. (devam) Ara yüz modülü

```
    def gerilme_hesapla_yy(self):
        if self.deger_al():
            ger = gerilme.GerilmeAnalizi(x=self.x, y=self.y, lx=self.lx, ly=self.ly, teta=self.teta,
                                         u=self.u, v=self.v, lamda=self.lamda, mu=self.mu, alfa=self.alfa,
                                         k_u_sol=self.k_u_sol, k_u_sag=self.k_u_sag, k_u_ust=self.k_u_ust, k_u_alt=self.k_u_alt,
                                         k_v_sol=self.k_v_sol, k_v_sag=self.k_v_sag, k_v_ust=self.k_v_ust, k_v_alt=self.k_v_alt)
            self.yy = ger.yy_gerilme_hesapla()
            self.sonuc_goster("gerilme_yy")
            ger.grafik_ciz("yy")

    def gerilme_hesapla_xy(self):
        if self.deger_al():
            ger = gerilme.GerilmeAnalizi(x=self.x, y=self.y, lx=self.lx, ly=self.ly, teta=self.teta,
                                         u=self.u, v=self.v, lamda=self.lamda, mu=self.mu, alfa=self.alfa,
                                         k_u_sol=self.k_u_sol, k_u_sag=self.k_u_sag, k_u_ust=self.k_u_ust, k_u_alt=self.k_u_alt,
                                         k_v_sol=self.k_v_sol, k_v_sag=self.k_v_sag, k_v_ust=self.k_v_ust, k_v_alt=self.k_v_alt)
            self.xy = ger.xy_gerilme_hesapla()
            self.sonuc_goster("gerilme_xy")
            ger.grafik_ciz("xy")

    def sabit_nokta_tanimlari(self, cmd):
        # icinde virgul yoksa ve "x,y" text ini eklemeye calisiyorsa yapma
```

EK-4. (devam) Ara yüz modülü

```
        if not (self.txt_sinir_sarti.get() == “x,y” or not (str(self.txt_sinir_sarti.get()).__contains__(“,”))):
            x = int(str(self.txt_sinir_sarti.get()).split(“,”)[0])
            y = int(str(self.txt_sinir_sarti.get()).split(“,”)[1])
            # girilen degerler levhayi geciyorsa ekleme
            if x >= int(self.txt_x.get()) or y >= int(self.txt_y.get()):
                return
            self.lbl_sinir_sarti_goster[“text”] = ““
            # ekleme ve cikarma butonlari ayni fonksiyonu kullanir, ekleme “e” ile
            # cikarma butonu “c” ile bu fonksiyonu cagirir
            if cmd == “ekle”:
                try:
                    i = 1
                    print(“sabit noktalar:”)
                    if self.u_sinir.get() == 1:
                        self.sinir_sarti_noktalari[“u:” + self.txt_sinir_sarti.get()] =“u:” + self.txt_sinir_sarti.get()
                    if self.v_sinir.get() == 1:
                        self.sinir_sarti_noktalari[“v:” + self.txt_sinir_sarti.get()] = “v:” + self.txt_sinir_sarti.get()
                    for k in self.sinir_sarti_noktalari.keys():
                        if i % 4 == 0:
                            self.lbl_sinir_sarti_goster[“text”] += “(“ + self.sinir_sarti_noktalari[k] + “) “ + “\n”
                        else:
                            self.lbl_sinir_sarti_goster[“text”] += “(“ + self.sinir_sarti_noktalari[k] + “) “
                        print(self.sinir_sarti_noktalari[k])
```

EK-4. (devam) Ara yüz modülü

```
                i += 1
            except Exception as e:
                print(e)
                pass
        if cmd == “cikar”:
            try:
                i = 1
                print(“sabit noktalar:”)
                if self.u_sinir.get() == 1:
                    self.sinir_sarti_noktalari.pop(“u:” + self.txt_sinir_sarti.get())
                if self.v_sinir.get() == 1:
                    self.sinir_sarti_noktalari.pop(“v:” + self.txt_sinir_sarti.get())
                for k in self.sinir_sarti_noktalari.keys():
                    if i % 4 == 0:
                        self.lbl_sinir_sarti_goster[“text”] += “(“ + self.sinir_sarti_noktalari[k] +
“) “ + “\n”
                    else:
                        self.lbl_sinir_sarti_goster[“text”] += “(“ + self.sinir_sarti_noktalari[k] +
“) “
                    print(self.sinir_sarti_noktalari[k])
                    i += 1
            except:
                pass


    def cbSol(self):
        if self.k_varsol.get() == 0:
            self.txtSiSol.lower(self.lblImg)
```

EK-4. (devam) Ara yüz modülü

```
        else:
            self.txtSiSol.lift(self.lblImg)
    def cbSag(self):
        if self.k_varsag.get() == 0:
            self.txtSiSag.lower(self.lblImg)
        else:
            self.txtSiSag.lift(self.lblImg)

    def cbUst(self):
        if self.k_varust.get() == 0:
            self.txtSiUst.lower(self.lblImg)
        else:
            self.txtSiUst.lift(self.lblImg)
    def cbAlt(self):
        if self.k_varalt.get() == 0:
            self.txtSiAlt.lower(self.lblImg)
        else:
            self.txtSiAlt.lift(self.lblImg)

    def hakkinda(self):
        hakkinda = Tk()
        bilgi = Label(master=hakkinda,
                      text="\nIsıl Gerilme Analizi\n\n" +
                           "Ekrandaki mavi alan bir dikdörtgen levhayı temsil eder.\n" +
                           "Girilen boyutlar levha gürüntüsünü değiştirmez.\n" +
                           "Analiz yapmaya sicaklik analizinden başlanmalıdır. Bu yüzden\n" +
```

EK-4. (devam) Ara yüz modülü

```
                "diğer butonlar pasif durumdadır.\n" +
                "Seçim kutularından ve metin kutularından veri girişi yapıldıktan\n" +
                "sonra sıcaklık analizi ile analiz yapılmaya başlanır.\n" +
                "Uzun süren analizlerde kullanıcı arabirimi bir süreliğine cevap vermez\n" +
                "görünebilir. Bu sırada hiçbir şey yapmadan bekleyiniz. Analiz bitiminde\n" +
                "arabirim yeniden cevap verir duruma dönecektir.\n\n" +
                "Soru & Önerileriniz için:\n" +
                "melikekorkmaz89@gmail.com\n\n" +
                "Melike Korkmaz - İnşaat Yüksek Mühendisi\n"
            )
        bilgi.configure(bg="white")
        bilgi.pack()
        hakkinda.title('Isıl Gerilme Analizi - Hakkında')
        hakkinda.resizable(0, 0)
        hakkinda.configure(background="white")
        windowWidth = hakkinda.winfo_reqwidth()
        windowHeight = hakkinda.winfo_reqheight()
        positionRight = int(hakkinda.winfo_screenwidth() / 5 - windowWidth / 2)
        positionDown = int(hakkinda.winfo_screenheight() / 3 - windowHeight / 2)
        hakkinda.geometry("+{}+{}".format(positionRight, positionDown))
        hakkinda.mainloop()


if __name__ == "__main__":
    m = Arayuz()
    # ekrandaki mavi levha goruntusu base64 string olarak ifade edilmistir.
```

EK-4. (devam) Ara yüz modülü

m.lbl_img_data = \

“iVBORw0KGgoAAAANSUhEUgAAAmQAAAFYCAYAAAD9QeD3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJ” +\

“cEhZcwAADsQAAA7EAZUrDhsAAAAndEVYdERlc2NyaXB0aW9uAEJsdWUgaGFuZC1wYWludGVkIGJhY2tkcm9wc+G3m” +\

“lsAAAtGSURBVHhe7dy/jxxnHcfxCf+EiZUCRbigoQFFtKbACopIT5RIbtJAFcVlKEiZiAoaN5EcmT4oIjjqKpI0skJ” +\

“AbCiNEETm+vyJ8n5l51nNzu7f3C33Op9dLWs/t3sw8s3snzVvPzPml4cGz7wYAAGK+Ny8BAAgRZAAAYYIMACBMkAE” +\

“AhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABh” +\

“ggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggA” +\

“wAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAM” +\

“IEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEE” +\

“GABAmyAAAwgQZAECYIAMACBNkAABhggyujP8Ow4ffH4Z37s3Pr7KTjnXf+7jItfQJ+143xw/xS1Ns87SWdcp+/+ZwO+ChsBSONwkd8bArRunJeXRu5+zmPjswRN/RrTKiUoCkGWL2m9WmmlrGNfSWZtRncJi29S/WvtUYEd1h61JWGFD4OlU6Ttyy6UuZbDrW8KzUSXTZpWdU5RfSSWO3c/h0KNvVwDjp4NV9qCrlHzs2MsHQDUk7+KqrOVMcwUsdn3gCR9AJEzhc8dh+dMN7HXvk43EvunjDXp1ycsYfjUzqjsGh4ad+1vmU1YwG7QsDcmyaHUGtWQ/0SOUnOVDfrb2X2KMjyXR9BZH3DIrM0/yRgMMUwXq6hVnjkGFuM2lzXHmbxN3yg7Uh5TXUCuUh5qhbJcnNsl8E3TiaAVuVobbqu0LR1rbLh8HL1WZodadWOs5O0YgHtYTWg+sGs6yfz+r9vtdeyxxBKygzl4d2O7IwPqA6iKnLnnrgLWkrHHDbr7TBDvT8orHCfnkm3HPG32zRHkfjyNKDR6gwWAeg3uJrxQ9WyMqzxLvVJk7WMHrPqKpoCwJJdK3/wxZwO7QCoqjKEUUzA5N8HCZsTAdoUvsDrrf4SEEafRIHPxfFnuLXVTKXtv6DsSaiN/SmVdTiKgxJKE5LW3L5sZDybJAYLgqddfEdeA4CTqnSCEtIQDbLy6GcdxzTs9SM4oE5BegmAE0w48jkChmhPXCmMgaJjQBo6VTGGNVgKXGsaR1OhL3CSW3xaJ6XoyiiGKtwJKxMK+hgYAkgDHyE15GFH5nyWc1Zkom+g1K8fMrNvWDUbAeWzD3hoeI9vmvmnM3ob3UAw2ZSSU/ta2Ua+K9fU8NuWjSyXUMK8Bc+wDPDIo5HMWSlpZ4YsMQVrHeyf2N5/HTKrmUA5mpiMzzhaxqXEXUXpLPD6Jwy7LtGMgfBwSFkIvsOLAMc1YDTp6JGmhKFGhLfgKIpC9xFOm32TvlEEFYFyXHCT0XWXvzT48X/4kc0mahlEmCwdKqM0vZV5QLGJcO+9hcqQD7QRHpjEtAtJkvCrgkaGGUTD5XA0PBiEz7Pxm4EQCDhHUtQ2jm/0wFdNspCzI8TGkXVNGDqpGcu8ENITWp0bhWbvmnG7yeXLVlfMOs1IqI23iUwbc6FPd/YnYmsBtG2PrrpuKuIhe+lrpzJ2pdv3X1PSqfSAAAAAElFTkSuQmCC” +\

“ggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggA” +\

“wAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAM” +\

“IEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEE” +\

“GABAmyAAAwgQZAECYIAMACBNkAABhggyujP8Ow4ffH4Z37s3Pr7KTjnXf+7jItlfQJ+143xiGw/n5NqdZZ+ms6wMv” +\

“PEEG53bZ8fCfYXhSi1u3pqdX2knHuu99XGTbq6Z+B75pyx8Nw43xhS1Os87SWdcHrgNBBud2yfFw+O9p+coPp+VVd” +\

“tKx7nsfF9n2yvnBMHzwbBgefDQ/3+Y06yyddX3g0hBkcF4nxcPBG8PwPw/rw/9Pf/ky2mdZnlp6rDVXblZ++” +\

EK-4. (devam) Ara yüz modülü

“3bfljbLI3rz4+DLfvr+lj90dfpx/V4ejpaHuuxY5sf67H6sf7k5/VPnymsR9vvke+V9Zhn2bY56TjGfdfj4N7z19f” +\

“j9XUeLz6T5ee6Xn9t7/iLbS/9eBafz7Z1gWtDkMF5/fMv03IZD007gT+s5UfP6vH7+uLvw3C/TqI31uFWJ+uvanHr” +\

“V9OlqX98Or46PKrHg9r2di2f1Bj9PqK237b++2325Ota73fT6zdfnZYbtd979b1bNXbbT3vcnY/xaR3L8NPnl8LGY” +\

“63Xxn3WY/jDNN6+scZjbftpwfCzaaawrfvj5fdq0azHPMu2+45j3Hc9nr7+/H2ux+vrfFaLzeda++mBtF5/6cj4f6” +\

“7Pah6//8x3jXVpx9Nny+b1mjffnb8ArhNBBue1PtE2bWaqncDf+uP0ep/xWc6ifTPPrD3+Ylq+9ov6p+JkvG+o9vf” +\

“ulhNu3+/tioIWLu1E/Vqt27xcXy/1mbsn89gbfYz53qS+zxZu4z7L3c/rn31jLfbz11VQrcc46fm+bfe+577+28+D” +\

“c+d4Oz7XY+svHBu/xhijqPZ14nv7PxzP43urYwGuG0EG57LjxNlnzR5WbLTLSx9/Op1ExxP0qxU/tRhDqbb/rM0G1” +\

“cn7TouLfj/aPFu23n/f72Y2rL7/aEsQNjfqRD+GQ+3/yCWu1T1vfZ9jEC7sHavv519TJLT3sImE9X11u56fYtvTHs” +\

“ft19uT2a7x1p/r+r309Re2jX/kZ75jrMs+nhaG7ffoSOgB140gg/M4/NvxE2fTZ83a5cp2malfthpVeL3SlhUjB3+” +\

“atn/rN+2F2t88q7WJoy0n5qbPhm3G7yf2lbtt7K/r+/V1v+y565639Qxbt2uszX7eex5+BxUWzXqMnc9PsW236zj6” +\

“DOMmmMq+8ddBtGvMpWPjzz+TIzOcZdvxrNfZdzzr9Vuw3W+XSdvvlJv84ToTZHAe2y5FNjfrxNnuD+r3fa317z+sE” +\

“GmXCsfZsdJnYzYn/x2h8G2Ll36SLsdC4svh+H/DsZplW9/zNu6ztJvJ+31Mza6xlvv5ZbtHrjyqWGnWY5z0fN+23a” +\

EK-4. (devam) Ara yüz modülü

“7j+LbCtlkG5b7x15/rrjGXxvHn+/KaXeNvO571a/uOZ73+wW+nYOuXwLt+43+PWeCF
J8jgPPqJ86tfTyfG9mh/UXe” +\

“nTpxtAuXj+bX+evdyxdGowmx5D9H6frQefH225c5703K8FFon6TfnmFnOxozqRP9+Lc
ax2z1abw+b/z5hPUY/1n55” +\

“9WmN0S4f7htruZ9+ebTflL4e46Tn+7Y9y3F0+8Zff67b9tEdGf+L+lzrs2x6TO0bq7nQ8d
TvTfuDi6b/jHpsj+uVZ” +\

“fwBL7SXhgfPvpu/BmCXNis1/vXs50ejC+ASmCED2ObIJdx5tmrXPXsAF2SGDGCbcU
ZsvjTYtHv+Ptj2X1UAXJwgAw” +\

“AIc8kSACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAA
gTJABAIQJMgCAMEEGABAmyAAAwgQZAEC” +\

“YIAMACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAg
TZAAAYYIMACBMkAEAhAkyAIAwQQYAECbI” +\

“AADCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQ
ZAECYIAMACBNkAABhggwAIEyQAQCECTIAg” +\

“DBBBgAQJsgAAMIEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYA
ECbIAADCBBkAQJggAwAIE2QAAGGCDAAgTJ” +\

“ABAIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABhggwAIEyQAQCEC
TIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZAA” +\

“AYYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDA
AgTJABAIQJMgCAMEEGABAmyAAAwgQZAECY” +\

“IAMACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZ
AAAYYIMACBMkAEAhAkyAIAwQQYAECbIA” +\

“ADCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZ
AECYIAMACBNkAABhggwAIEyQAQCECTIAgD” +\

“BBBgAQJsgAAMIEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAE
CbIAADCBBkAQJggAwAIE2QAAGGCDAAgTJA” +\

EK-4. (devam) Ara yüz modülü

“BAIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABhggwAIEyQAQCECTI
AgDBBBgAQJsgAAMIEGQBAmCADAAgTZAAA” +\

“YYIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAAg
TJABAIQJMgCAMEEGABAmyAAAwgQZAECYI” +\

“AMACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZ
AAAYYIMACBMkAEAhAkyAIAwQQYAECbIAA” +\

“DCBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZA
ECYIAMACBNkAABhggwAIEyQAQCECTIAgDB” +\

“BBgAQJsgAAMIEGQBAmCADAAgTZAAAYYIMACBMkAEAhAkyAIAwQQYAEC
bIAADCBBkAQJggAwAIE2QAAGGCDAAgTJAB” +\

“AIQJMgCAMEEGABAmyAAAwgQZAECYIAMACBNkAABhggwAIEyQAQCECTIA
gDBBBgAQJsgAAMIEGQBAmCADAAgTZAAAY” +\

“YIMACBMkAEAhAkyAIAwQQYAECbIAADCBBkAQJggAwAIE2QAAGGCDAAgT
JABAIQJMgCAMEEGABAmyAAAwgQZAECYIA” +\

“MACBNkAABhggwAIEyQAQCECTIAgDBBBgAQJsgAAMIEGQBAmCADAAgTZA
AAYYIMACBMkAEAhAkyAIAwQQYAECbIAAD” +\

“CBBkAQJggAwAIE2QAAGGCDAAgTJABAIQJMgCAMEEGABAmyAAAwgQZAE
CYIAMACBNkAABhggwAIEyQAQCECTIAgDBBB” +\

“gAQJsgAAMIEGQBAmCADAIgahv8BZDaxjlZKUQwAAAAASUVORK5CYII=“

m.arayuz_olustur()

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : KORKMAZ, Melike

Uyruğu : T.C.

Doğum tarihi ve yeri : 15.10.1989, Kütahya

Medeni hali : Evli

Telefon : 0 (505) 497 37 80

e-mail : melikekorkmaz89@outlook.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek Lisans	Gazi Üniversitesi / İnşaat Mühendisliği	Devam Ediyor
Lisans	Eskişehir Osmangazi Üniversitesi / İnşaat Mühendisliği	2012
Lise	Ali Güral Anadolu Lisesi	2007

İş Deneyimi

Yıl	Yer	Görev
2018-devam ediyor	Karayolları 3. Bölge Müdürlüğü	Bakım Mühendisi
2013-2016	Kütahya Belediyesi	Proje Kontrol Müh.
2012-2013	Han Yapı Denetim	Kontrol Müh.

Yabancı Dil

İngilizce

Yayınlar

1. Korkmaz, M. ve Alyavuz, B. (2019, 7-8 Aralık). *Python Ortamında Termoelastik Gerilme Analizi.* Selçuk 1. Uluslararası Uygulamalı Bilimler Kongresi, Konya.

Hobiler

Tenis, Doğa yürüyüşleri

GAZİ GELECEKTİR..