T.C.
AKDENİZ ÜNİVERSİTESİ

HAVACILIK SEKTÖRÜNDE KULLANILAN KATMANLI KOMPOZİT YAPILARIN TASARIMI VE OPTİMİZASYONU

Fatih ÖZTÜRK

FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

HAZİRAN 2025

ANTALYA

T.C.
AKDENİZ ÜNİVERSİTESİ

HAVACILIK SEKTÖRÜNDE KULLANILAN KATMANLI KOMPOZİT YAPILARIN TASARIMI VE OPTİMİZASYONU

Fatih ÖZTÜRK

FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

HAZİRAN 2025

ANTALYA

T.C.
AKDENİZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAVACILIK SEKTÖRÜNDE KULLANILAN KATMANLI KOMPOZİT YAPILARIN TASARIMI VE OPTİMİZASYONU

Fatih ÖZTÜRK

MAKİNE MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Bu tez 30/06/2025 tarihinde jüri tarafından oybirliği ile kabul edilmiştir.

Prof. Dr. Gürkan ALTAN (Danışman)
Prof. Dr. Hakan ERSOY
Dr. Öğr. Üyesi Kayra KURŞUN

ÖZET

HAVACILIK SEKTÖRÜNDE KULLANILAN KATMANLI KOMPOZİT YAPILARIN TASARIMI VE OPTİMİZASYONU

Fatih ÖZTÜRK

Yüksek Lisans Tezi, Makine Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Gürkan ALTAN

Haziran 2025; 68 sayfa

Havacılık sektöründe geniş bir kullanım alanına sahip olan kompozit malzemeler hafiflik, dayanıklılık ve yakıt verimliliği gibi konularda büyük avantajlar sağlamaktadır. Bu kapsamda, kompozit malzemelerin performansını artırmak amacıyla son yıllarda optimizasyon yöntemlerine sıklıkla başvurulmaktadır. Bu tez çalışmasında, Genetik Algoritma (GA) ile optimizasyon işlemleri gerçekleştirilerek sürekli elyaf takviyeli katmanlı kompozit bir yapının yapısal deformasyonunun en aza indirgenmesi amaçlanmıştır. Ayrıca bu çalışma ile, yapay zekâ temelli optimizasyon işlemlerinin havacılık sektöründe yoğun olarak kullanılan katmanlı kompozit yapıların tasarımında dikkate alınması, bu tarz yeni teknolojilerin işletmelere entegre edilmesinde yol gösterici olması hedeflenmiştir. Çalışmanın başlangıcında bu çalışmaya özel Python dilinde kodlar geliştirilmiş, bu kodlar aracılığıyla genetik algoritmanın sonlu elemanlar çözücüsü ile entegrasyonu sağlanmıştır. Kod içerisinde genetik algoritma ile optimizasyon işlemleri için gerekli olan temel fonksiyonlar geliştirilmiş, amaç fonksiyonu ve kısıt fonksiyonu başta olmak üzere temel optimizasyon parametreleri tanımlanmıştır. Çalışmanın ikinci kısmında, farklı katman dizilimlerine sahip kompozit kabuklar tasarlanmış ve optimizasyon sonuçlarıyla kıyaslayabilmek amacıyla her biri için FEM analizleri yapılmıştır. Çalışmanın son kısmında, tasarlanan kompozit kabuklar için optimizasyon işlemleri gerçekleştirilmiştir. Bu kısımda optimizasyon sürecini etkileyebilecek kritik parametreler belirlenerek her bir optimizasyon işleminde farklı parametre değerleri ile işlemler gerçekleştirilmiştir. Elde edilen sonuçlar incelendiğinde, kompozit kabukların FEM analizleri ile hesaplanan sehim miktarları, genetik algoritma ile yapılan optimizasyonlar sonucunda %60'lara kadar düşürülmüş ve optimum elyaf oryantasyon açıları elde edilmiştir.

ANAHTAR KELİMELER: Genetik algoritma, Katmanlı kompozit yapılar, Optimizasyon, Python, Sonlu elemanlar metodu

JÜRİ: Prof. Dr. Gürkan ALTAN

Prof. Dr. Hakan ERSOY

Dr. Öğr. Üyesi Kayra KURŞUN

ABSTRACT

DESIGN AND OPTIMIZATION OF THE LAMINATED COMPOSITE STRUCTURES USED IN AEROSPACE INDUSTRY

Fatih ÖZTÜRK

MSc Thesis in Mechanical Engineering Department

Supervisor: Prof. Dr. Gürkan ALTAN

June 2025; 68 pages

Composite materials, which have a wide range of applications in the aerospace industry, provide great advantages in terms of lightness, durability and fuel efficiency. In this context, optimization methods have been frequently used in recent years to improve the performance of composite materials. In this thesis, it is aimed to minimize the structural deformation of a continuous fiber-reinforced laminated composite structure by performing optimization processes with Genetic Algorithm (GA). In addition, with this study, it is aimed to consider artificial intelligence-based optimization processes in the design of laminated composite structures which are widely used in the aerospace industry and to guide the integration of such new technologies into enterprises. At the beginning of the study, codes specific to this study were developed in Python language, and the integration of the genetic algorithm with the finite element solver was achieved through these codes. In the code, the basic functions required for optimization operations with the genetic algorithm were developed and the basic optimization parameters, especially the objective function and constraint function, were defined. In the second part of the study, composite shells with different layer arrangements were designed and FEM analyses were performed for each of them in order to compare with the optimization results. In the last part of the study, optimization processes were performed for the designed composite shells. In this section, the critical parameters that may affect the optimization process were determined and operations were performed with different parameter values in each optimization process. When the obtained results are examined, the deflection amounts of the composite shells calculated by FEM analysis were reduced up to 60% as a result of the optimizations made with the genetic algorithm and optimum fiber orientation angles were obtained.

KEYWORDS: Finite element method, Genetic algorithm, Laminated composite structures, Optimization, Python

COMMITTEE: Prof. Dr. Gürkan ALTAN

Prof. Dr. Hakan ERSOY

Asst. Prof. Kayra KURŞUN

ÖNSÖZ

Uzun yıllar havacılık sektöründe çalışan bir mühendis olarak edindiğim birtakım bilgi ve tecrübeleri bu tez çalışmasıyla bir adım daha ileriye taşımak istedim. Kompozit malzemelerin kullanım alanlarının gelecekte daha da yaygınlaşacağı inancıyla bu alandaki tasarım faaliyetlerini yapay zekâ ile destekleyerek geliştirmek, yapay zekânın mühendislik uygulamalarına yapabileceği katkıya dikkat çekmek amacıyla bu tezi hazırladım. Çalışmamın mühendislik faaliyetlerinde faydalı olmasını ve bu alanda çalışan arkadaşlara yol gösterici olmasını umuyorum.

Bu süreç boyunca desteğini bir an olsun benden esirgemeyen, engin bilgi ve tecrübelerini benimle her daim paylaşan değerli hocam sayın Doç. Dr. Muhsin Gökhan GÜNAY'a, yapıcı ve motive edici yaklaşımıyla beni en güzel şekilde destekleyerek yön veren kıymetli hocam sayın Prof. Dr. Gürkan ALTAN'a teşekkürlerimi bir borç bilirim. Bu iki harika bilim insanıyla tanışmış olmak benim için bir lütuftur.

Son olarak beni bugünlere yetiştiren ve varlığına her daim şükrettiğim canım aileme sonsuz teşekkürlerimi sunuyorum.

Bu tezi, ilim irfan yolunda benden hiçbir şeyini esirgemeyen, bugünlere gelmemde sayısız emeği olan, maddi ve manevi desteğini asla unutmayacağım rahmetli dedem Cavit Öztürk'e atfediyorum.

İÇİNDEKİLER

AKADEMİK BEYAN

Yüksek Lisans Tezi olarak sunduğum “Havacılık Sektöründe Kullanılan Katmanlı Kompozit Yapıların Tasarımı ve Optimizasyonu” adlı bu çalışmanın, akademik kurallar ve etik değerlere uygun olarak yazıldığını belirtir, bu tez çalışmasında bana ait olmayan tüm bilgilerin kaynağını gösterdiğimi beyan ederim.

30/06/2025

Fatih ÖZTÜRK

İmzası

SİMGELER VE KISALTMALAR

Simgeler

% : Yüzde

° : Derece

mm : Milimetre

E_1 : Boyuna Elastisite Modülü

E_2 : Enine Elastisite Modülü

G : Düzlem İçi Kayma Modülü

ν : Poisson Oranı

Y_c : Elyafa Dik Yöndeki Basma Mukavemeti

Y_t : Elyafa Dik Yöndeki Çekme Mukavemeti

X_c : Elyaf Yönündeki Basma Mukavemeti

X_t : Elyaf Yönündeki Çekme Mukavemeti

S : Düzlem İçi Kayma Mukavemeti

W : İş

ρ : Yoğunluk

u : x Eksenindeki Yer Değiştirme

v : y Eksenindeki Yer Değiştirme

w : z Eksenindeki Yer Değiştirme

σ : Gerilme

ϵ : Şekil Değiştirme

Bu tez kapsamında ondalık ayracı olarak (.) kullanılmıştır.

Kısaltmalar

3B : Üç Boyutlu

ABC : Yapay Arı Kolonisi Algoritması

ACO : Karınca Kolonisi Optimizasyonu

AFP : Otomatik Fiber Yerleştirme

C : Karbon

CAD : Bilgisayar Destekli Tasarım

CFRP : Karbon Fiber Takviyeli Polimer

CLPT : Klasik Katmanlı Plaka Teorisi

CLT : Klasik Lamine Teorisi

DOF : Serbestlik Derecesi

EA : Evrimsel Algoritma

GA : Genetik Algoritma

FE : Sonlu Elemanlar

FEA : Sonlu Elemanlar Analizi

FEM : Sonlu Elemanlar Metodu

GFRP : Cam Elyaf Takviyeli Polimer

LEF : Hücum Kenarı Flabı

NSGA : Baskın Olmayan Sıralama Genetik Algoritması

PSO : Parçacık Sürü Optimizasyonu

SA : Benzetilmiş Tavlama

TS : Tabu Arama

UD : Tek Yönlü

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Kompozit malzemeler, birbirinden farklı fiziksel ve kimyasal özelliklere sahip birden fazla malzemenin bir araya gelip ortaya daha farklı özelliklere sahip yeni bir malzeme ortaya çıkması ile elde edilirler. Buradaki temel amaç elde edilen yeni malzemenin daha dayanıklı, daha hafif ve daha uzun ömürlü olması gibi güçlü özelliklere sahip olmasıdır. Etrafımıza göz attığımızda aslında birçok yapının kompozit malzemelerden oluştuğu görülebilmektedir.

Tarihsel gelişimine bakıldığında kompozit malzemelerin sadece günümüz modern mühendislik uygulamalarında değil aynı zamanda insanlık tarihinin erken dönemlerinden itibaren çeşitli işlevsel amaçlarla kullanıldığı görülmektedir. Arkeolojik bulgular, özellikle Mezopotamya ve Antik Mısır gibi erken uygarlıklarda saman gibi doğal elyafların kil ile karıştırılarak kerpiç tuğla üretiminde kullanıldığını ortaya koymuştur. Bu yöntem, malzemenin çekme dayanımını ve çatlamaya karşı direncini artırmak amacıyla geliştirilmiştir (Hull ve Clyne 1996). Eski Mısırlılar M.Ö. 3000 yıllarında çamur ve samanı karıştırarak kerpiç tuğlalardan yapılar inşa etmişlerdir. Saman, çamurla birlikte kullanıldığında çamurun dayanıklılığını artırırken aynı zamanda malzemenin çatlamasını önlediği için bu şekilde daha uzun ömürlü yapılar elde edilirdi (Ashby ve Jones 2012). Benzer şekilde, Antik Mısır'da ahşap levhaların doğal yapıştırıcılarla bir araya getirilmesiyle elde edilen kontrplak benzeri yapı elemanları erken dönem kompozit malzeme uygulamalarının önemli örneklerinden sayılmaktadır (Strong 2008). Bu ve benzeri eski yöntemler malzemelerin doğal özelliklerinden faydalanarak onları daha dayanıklı ve kullanışlı hale getirme düşüncesinin ilk örneklerini oluşturmakta, kompozit malzemelerin yüksek dayanım, hafiflik ve işlevsellik gibi özelliklerinden tarih boyunca yararlanıldığını ve yapı teknolojilerinin gelişiminde önemli bir yere sahip olduğunu ortaya koymaktadır. Kompozitlerin antik çağlardan bu yana doğal malzemelerle bu şekilde uyum içinde kullanılması dolayısıyla, bu malzemeler günümüzdeki modern kompozitlerin öncülleri olarak kabul edilmiştir. Günümüzde ise kompozit malzemeler sahip olduğu üstün özellikleri sayesinde inşaattan otomotive, tıp alanından havacılığa kadar geniş bir yelpazede kullanılmaktadır.

Havacılık sektörü, kompozit malzemelerin en sık kullanıldığı önemli alanlardan biridir. Hafiflik, yüksek dayanım, korozyon direnci ve uzun ömürlü olması gibi önemli avantajlara sahip olması, kompozitlerin uçaklarda ve uzay araçlarında sıklıkla tercih edilmesine yol açmaktadır. Bunlardan sürekli elyaf takviyeli katmanlı kompozit yapılar, uçak ve uzay araçlarının performans gereksinimlerini karşılamak amacıyla oldukça tercih edilmektedir. Bu tür kompozitler elyafların katmanlar halinde düzenlenip matris malzemesi ile birleştirilmesiyle üretilir. Karbon fiber, cam fiber ve aramid fiber havacılık sektöründe en yaygın kullanılan elyaf türleridir. Özellikle karbon fiber kompozitler kanatlarda, gövdede ve iç yapılarda kullanılarak uçakların toplam ağırlığını önemli ölçüde azaltmaktadır. Bu aynı zamanda hem yakıt tüketimini hem de bakım maliyetlerini düşürerek uçakların performans kabiliyetini artırmaktadır. Günümüz modern uçaklarının büyük bir kısmı, kompozit malzemelerin sağladığı bu avantajlar sayesinde daha uzun menziller kat edebilmekte ve daha çevre dostu bir alternatif sunmaktadır.

Kompozit malzemelerin havacılık sektöründeki kullanımı, 20. yüzyılın başlarına

kadar uzanmaktadır. İlk olarak 1930'larda, doğal kompozitler olan ahşap ve kanvas uçak gövdesi ve kanatlarında kullanılıyordu (Baker vd. 2004). Ancak bu malzemeler nemden etkilenme ve sınırlı dayanıklılık gibi problemlerle karşı karşıya kalınca sentetik kompozitler kullanılmaya başlandı. Bu nedenle cam elyaf gibi sentetik kompozitlerin kullanımı 1950'li yıllarda havacılıkta yaygınlaştı ve özellikle helikopter rotorlarında ve uçak kaplamalarında kullanılmaya başlandı (Mallick 2007).

1960'lara gelindiğinde karbon fiber kompozitler geliştirildi. Karbon fiber, hafiflik ve yüksek dayanıklılık gibi özellikleri nedeniyle uçak yapılarında geniş kullanım alanı bulmaya başladı (Campbell 2010). 1980'li yıllarda, Boeing ve Airbus gibi büyük uçak üreticileri karbon fiber takviyeli polimer (CFRP) gibi gelişmiş kompozitleri uçak gövdelerinde ve kanat yapılarında kullanarak havacılık sektöründe devrim yaratmaya başladılar (Baker vd. 2004). Bu dönemde kompozitlerin yakıt tasarrufu sağlaması ve bakım maliyetlerini ciddi ölçüde düşürmesi önemli bir avantaj olarak öne çıktı. Örneğin 1987'de Boeing 777 uçağı, kanatçıkları ve kuyruk kısmında karbon fiber kompozitler kullanılarak tasarlandı (Agarwal vd. 2006).

2000'li yıllardan itibaren kompozit malzemelerin kullanımı giderek daha da yaygınlaştı. Boeing 787 Dreamliner ve Airbus A350 gibi modern uçaklar, gövde ve kanat yapılarında karbon fiber kompozitlere daha fazla yer vererek, bu malzemelerin getirdiği hafiflik ve yüksek dayanıklılık avantajından faydalandılar. Bu uçakların yapısının %50'den fazlası kompozit malzemelerden oluşmaktadır (Campbell 2010).

Günümüzde artık kompozitler havacılık endüstrisinin önemli bir parçası haline gelmiş ve hava araçlarının verimlilik, dayanıklılık ve performansını artırmak için yaygın olarak tercih edilmektedir. Ayrıca kompozit malzemelerdeki teknolojik gelişmeler üretim süreçlerini hızlandırırken çevresel etkilerin de azalmasını sağlamaktadır. Özellikle Airbus ve Boeing gibi büyük üreticiler, kompozit malzemeleri daha büyük yapılarda kullanarak karbon ayak izini düşürmeye çalışmaktadır. Gelecekte daha fazla metal-kompozit hibrit yapı ve 3B baskı gibi yeniliklerin kullanımının yaygınlaşacağı tahmin edilmektedir.

Yukarıda bahsedildiği üzere karbon fiber, cam fiber ve aramid fiberler havacılık sektöründe ve aynı zamanda savunma sanayinde üstün özellikleri nedeniyle en sık tercih edilen kompozit malzemelerin başında gelmektedir. Bunlara ek olarak son yıllarda termoplastik malzemeler de eklenmiştir. Karbon fiber takviyeli polimer kompozitler, havacılık endüstrisinde yüksek özgül mukavemetleri, rijitlikleri ve korozyon dirençleri sayesinde giderek artan bir kullanım alanı bulmaktadır. Özellikle modern yolcu uçaklarında yapısal elemanlarda kullanılan karbon fiber kompozitler, ağırlık azaltımı yoluyla yakıt tüketimini düşürmekte ve böylece hem ekonomik hem de çevresel açıdan büyük avantaj sağlamaktadır. Geleneksel metal alaşımlarına kıyasla üstün yorulma ve darbe dayanımı sunan bu malzemeler uçak gövdeleri, kanatlar ve kuyruk yapılarında sıkça tercih edilerek havacılık tasarımında önemli bir dönüşüm yaratmaktadır (Gay vd. 2003). Cam elyaf takviyeli kompozitler (GFRP), havacılıkta hafiflik, yüksek darbe dayanımı ve üstün korozyon direnci gibi özelliklerinden dolayı uzun yıllardır tercih edilen malzemeler arasında bulunmaktadır. Özellikle gövde panelleri, iç kabin parçaları, anten radomları ve kaplama elemanlarında kullanılan GFRP malzemeler, üretim maliyetlerinin karbon fiber kompozitlere göre daha düşük olması ve kolay şekillendirilebilmeleri sayesinde havacılık uygulamalarında önemli bir alternatif

sunmaktadır. Ayrıca elektromanyetik dalgalara karşı geçirgenlik özellikleri dolayısıyla radar sistemlerinde ve hava aracı burun konileri gibi kritik bölgelerde de yaygın şekilde kullanılmaktadır (Daniel ve Ishai 2006). Kevlar takviyeli kompozitler, yüksek özgül mukavemetleri ve son derece üstün darbe dayanımları sebebiyle havacılık sektöründe önemli uygulama alanları bulmaktadır. Bu malzemeler, özellikle helikopter rotor paletleri, uçak kabin panelleri, balistik koruma elemanları ve radom yapılarında kullanılarak yapısal dayanıklılığı artırmakta ve ağırlık avantajı sağlamaktadır. Kevlar'ın yüksek enerji emme kapasitesi, titreşim sönümleme ve akustik yalıtım gibi özellikleri uçak performansını ve yolcu konforunu iyileştirmede kritik rol oynamaktadır (Hearle 2001). Termoplastik kompozit malzemeler, havacılık sektöründe özellikle yapısal hafiflik ve üretim verimliliği gibi avantajları sayesinde tercih edilmektedir. Bu malzemeler, termoplastik reçinelerin ergime ve yeniden şekillendirilebilir özellikleri sayesinde kompleks geometrilerin daha hızlı ve ekonomik olarak üretilmesine imkân tanımaktadır. Zeyrek vd. (2022), bu malzemelerin özellikle hızlı üretim süreçleri ve geri dönüştürülebilirlik özellikleri sayesinde uçak gövde panelleri, bağlantı elemanları ve iç kabin bileşenleri gibi yapılarda tercih edildiğini, aynı zamanda montaj süreçlerinde kaynaklanabilirlik gibi ilave kolaylıklar sağladıklarını bildirmektedir. Ayrıca son yıllarda yapılan araştırmalara göre, termoplastik kompozitlerin havacılık sektöründe termoset kompozitlerin yerini alabilecek önemli bir alternatif olarak ortaya çıktığını vurgulamaktadır.

Kompozit malzemelerin havacılık endüstrisinde bu kadar yaygınlaşması, bu ürünlerin mühendisliğine duyulan ilgiyi daha da çok artırmaktadır. Bu kapsamda, özellikle son yıllarda kompozit yapıların performansını artırmak, ağırlığını azaltmak ve belirli tasarım kriterlerine uygunluk sağlamak amacıyla optimizasyon yöntemlerine başvurulmaktadır.

Kompozit malzemelerin optimizasyon sürecinde, tasarım değişkenleri olarak elyaf oryantasyon açısı, katman dizilimi (stacking sequence), katman kalınlığı, toplam katman sayısı ve kullanılan malzeme türü gibi parametreler dikkate alınmaktadır. Bu parametrelerin uygun şekilde belirlenmesi, yapıların minimum ağırlık ve optimum performansla tasarlanmasına imkân sağlamaktadır. Optimizasyon problemleri genellikle belirli bir amaç fonksiyonunun (minimum ağırlık, maksimum rijitlik ya da minimum sehim gibi) önceden tanımlanmış kısıtlar altında en iyi değerine ulaşmasını amaçlamaktadır.

Optimizasyon yöntemleri hakkında ilerleyen bölümlerde daha detaylı bilgi aktarılacak ancak kompozit malzeme özelinde kısaca değinmek gerekirse, kompozitlerin optimizasyonu için geliştirilen yöntemler genel olarak deterministik ve sezgisel yöntemler olmak üzere iki grupta incelenebilir.

Deterministik yöntemler, klasik matematiksel optimizasyon tekniklerini içermektedir. Doğrusal programlama, doğrusal olmayan programlama ve türev temelli yöntemler bu grupta yer almaktadır. Bu yöntemler, küçük ve lineer yapıdaki problemlerde etkin sonuçlar sunmasına karşın, yüksek değişken sayısına ve parametrelerin belli alternatifler veya seçenekler arasından seçildiği ayrık tasarım parametrelerine sahip karmaşık kompozit yapıların optimizasyonunda yetersiz kalabilmektedir. Bu tür durumlarda sezgisel ve evrimsel yöntemler tercih edilmektedir. Karmaşık ve çok değişkenli optimizasyon problemlerinde sezgisel ve evrimsel

yöntemler daha etkin ve esnek çözümler sunmaktadır. Genetik Algoritmalar (GA), Parçacık Sürü Optimizasyonu (PSO), Yapay Arı Kolonisi Algoritması (ABC) ve Tabu Arama (TS) gibi yöntemler, özellikle ayrık tasarım parametrelerinin optimize edilmesinde sıklıkla kullanılmaktadır. Bu algoritmalar, global optimuma ulaşma olasılığı yüksek ve problem yapısına kolayca uyarlanabilir olmaları nedeniyle tercih edilmektedir.

Kompozit malzemelerin optimizasyonunda genetik algoritma kullanımı, mühendislik ve bilimsel araştırma alanlarında son yıllarda önemli bir ilgi odağı haline gelmiştir. Genetik algoritmalar, biyolojik evrim mekanizmalarını temel alan karmaşık ve çok değişkenli problemlerin çözümünde oldukça etkili bir yapay zekâ yöntemidir. Bu algoritmalar, özellikle çözüm uzayının geniş ve doğrusal olmayan yapıda olduğu durumlarda optimum veya optimuma yakın çözümlerin bulunmasında avantaj sağlamaktadır. Kompozit malzemelerin katman diziliminden malzeme özelliklerine ve üretim parametrelerine kadar çeşitli tasarım değişkenlerinin optimizasyonunda genetik algoritmaların kullanımı, daha hafif, daha dayanıklı ve yüksek performanslı yapıların geliştirilmesine oldukça katkı sağlamaktadır. Bu nedenle, kompozit malzeme tasarımı süreçlerinde genetik algoritma tabanlı optimizasyon yöntemleri mühendislik uygulamalarında yaygın ve etkin bir araç olarak tercih edilmektedir. (Tezin bundan sonraki kısımlarında kolaylık olması açısından, genetik algoritma ifadesi yerine zaman zaman GA kısaltması kullanılacaktır).

Kompozit yapıların optimizasyon sürecinde, yapıların mekanik davranışlarının önceden sayısal olarak hesaplanması gereklidir. Bu amaçla sonlu elemanlar metodu (FEM) kullanılarak gerçekleştirilen analiz işlemleri, belirlenen yükleme ve sınır koşulları altında yapının davranışını belirlemek için yaygın biçimde kullanılmaktadır. Elde edilen analiz sonuçları, optimizasyon algoritmalarına performans verisi sağlamakta ve en uygun tasarım parametrelerinin belirlenmesine olanak sağlamaktadır.

Kompozit malzemelerin optimizasyonu, hafif ve yüksek performanslı yapıların tasarlanması açısından kritik bir öneme sahiptir. Deterministik yöntemler küçük ve basit problemler için uygunken, karmaşık ve çok değişkenli problemlerde sezgisel ve evrimsel yöntemlerin daha etkin olduğu görülmektedir. Ayrıca sonlu elemanlar metodu, optimizasyon sürecinin önemli bir bileşeni olarak öne çıkmaktadır. Gelişen bilgisayar teknolojisi ve sayısal analiz yöntemleri sayesinde çok katmanlı kompozit yapıların ağırlık, rijitlik ve dayanım gibi parametrelerinin eşzamanlı olarak optimize edilmesi günümüzde başarıyla gerçekleştirilebilmektedir.

2. KAYNAK TARAMASI

2.1. Optimizasyon

Optimizasyon, belirli bir amaca yönelik en uygun çözümü bulmak için matematiksel modeller ve algoritmalar kullanarak karar alma sürecini iyileştirmeye odaklanan, çok yönlü ve güçlü bir yöntemdir. Kaynakların sınırlı olduğu durumlarda bu kaynakların en etkili şekilde kullanılmasını sağlayarak karmaşık problemlerin çözümünde kritik bir rol oynar. Optimizasyon süreçlerinde, belirlenen bir hedefin en iyi şekilde yerine getirilmesi amaçlanır; bu hedef, maliyeti en aza indirmek ya da performansı en üst düzeye çıkarmak gibi ölçülebilir kriterler olabilir.

Optimizasyon teknikleri mühendislikten ekonomiye, lojistikten finansa, tarımdan enerji yönetimine kadar geniş bir yelpazede kullanılır. Örneğin, mühendislikte optimum tasarım problemleriyle ağırlık azaltma veya mukavemeti artırma gibi hedefler belirlenebilirken, lojistikte rota optimizasyonu ve zaman planlaması yapılarak maliyetler azaltılabilir. Optimizasyon teknikleriyle gerçek hayatın karmaşıklıkları çözüm sürecine dahil edilerek, karar verme süreçleri daha stratejik, hesaplı ve etkili hale getirilir.

Optimizasyonda, amaç fonksiyonu adı verilen bir matematiksel model kullanılır ve bu model, çözülmesi gereken problemde optimize edilecek hedefi tanımlar. Çözüm sırasında bu fonksiyonun yanında, bazı kısıtlayıcı koşullar da göz önünde bulundurulmalıdır. Bu kısıtlamalar, genellikle sistemin fiziksel, ekonomik veya mantıksal sınırlarını ifade eder ve çözüm aralığını daraltarak problemin çözüm sürecini daha gerçekçi hale getirir. Amaç fonksiyonu, çözüm arayışı esnasında değeri değişebilen parametrelere sahiptir ve bunlar kontrol veya karar değişkenleri olarak tanımlanmaktadır. Optimizasyon, amaç fonksiyonunun en büyük veya en küçük değerinin bulunmasıyla gerçekleşir. Bir optimizasyon problemi matematiksel olarak genellikle şu şekilde ifade edilir:

$$
\begin{aligned}
&\text{min / maks} \quad && f_m(x) && m = 1, \dots, M \\
& && g_j(x) \leq 0 && j = 1, \dots, J \\
& && h_k(x) = 0 && k = 1, \dots, K \\
& && x_i^L \leq x_i \leq x_i^U && i = 1, \dots, N \\
& && x \in \Omega &&
\end{aligned}
$$

Burada x_i ifadesi optimize edilecek i'inci karar (tasarım) değişkenini, x_i^L ve x_i^U ifadeleri değişkenin alt ve üst limitlerini, f_m ifadesi m'inci amaç fonksiyonunu, g_j ifadesi j'inci eşitsizlik kısıtlamasını ve h_k ifadesi de k'ıncı eşitlilik kısıtlamasını ifade etmektedir. Ayrıca, M amaç fonksiyonu sayısını, J ve K kısıtlayıcı fonksiyon sayısını, N değişken sayısını ve Ω arama uzayını ifade etmektedir. Burada, f amaç fonksiyonunun tüm kısıtlamaları sağlayacak şekilde minimize ya da maksimize edilmesi amaçlanmaktadır. Spesifik olarak bir amaç fonksiyonu maksimize edilmek istenilirse (max f_i), problem, fonksiyonun negatifinin minimize edilmesi (min $-f_i$) şeklinde de

tanımlanabilir. Kısıtlı bir optimizasyon problemi örneği aşağıda gösterildiği gibi yazılabilir:

$$\begin{aligned} &\text{min} && f_1(x) = 50(x_1^2 + x_2^2) \\ &\text{maks} && f_2(x) = -(x_1 - 3)^2 - x_2^2 \\ & && g_1(x) = 3(x_1 - 0.2)(x_1 - 0.7) \leq 0 \\ & && g_2(x) = 30(x_1 - 0.5)(x_1 - 0.4) \geq 0 \\ & && -2 \leq x_1 \leq 2 \\ & && -2 \leq x_2 \leq 2 \\ & && x \in \mathbb{R} \end{aligned}$$

Bu optimizasyon problemi, f_1 fonksiyonunun minimize edildiği ve f_2 fonksiyonunun maksimize edildiği 2 adet amaç fonksiyonu ($M = 2$) içermektedir. Optimizasyon işleminde, g_1 küçük eşit ve g_2 büyük eşit olmak üzere 2 adet eşitsizlik kısıtlaması uygulanmaktadır ($J = 2$). Problemde, x_1 ve x_2 olmak üzere 2 adet tasarım değişkeni tanımlanmış olup ($N = 2$) herhangi bir eşitlilik kısıtlaması tanımlanmamıştır ($K = 0$).

2.2. Optimizasyonun Kısa Bir Tarihi

Optimizasyonun tarihsel kökenlerine bakıldığında çok eski matematiksel çalışmalara kadar dayandığı görülmektedir. Özellikle Antik Yunan ve Çin'deki matematikçiler, çeşitli geometrik ve aritmetik problemlerin en iyi çözümlerini aramaya yönelik araştırmalar ortaya koymuşlardır. Örneğin, Yunan matematikçi Heron, ışığın en kısa yoldan geçmesi ilkesini inceleyerek optimizasyon prensiplerine dayalı ilkeleri ortaya koymuştur (Struik 1948). Orta Çağ'da da İslam dünyasında matematikçiler, özellikle Harizmi ve Ömer Hayyam gibi bilim insanları, denklemler ve sayısal analiz üzerine çalışmış, belirli matematiksel sorunlara en uygun çözümler geliştirmeye çalışmışlardır (Rashed 2002).

Modern anlamda optimizasyonun ilk önemli adımları 17. yüzyılda İskoç matematikçi James Gregory ve Alman matematikçi Johannes Kepler'in maksimum ve minimum değerlerin bulunmasına yönelik çalışmalarıyla atılmıştır. 18. yüzyılda ise Joseph-Louis Lagrange ve Leonhard Euler, optimizasyon problemlerine yönelik çalışmalar yapmış ve matematiksel analiz hesaplarını geliştirmişlerdir. Bu dönemde geliştirilen Lagrange çarpanları yöntemi, belirli kısıtlar altında en uygun çözüme ulaşmada önemli bir yenilik olarak kabul edilmektedir (Luenberger ve Ye 2015).

II. Dünya Savaşı sırasında optimizasyon alanında büyük bir gelişme yaşanmıştır. Yöneylem araştırması (operations research) adı altında bir araya getirilen çalışmalar, askeri lojistik ve kaynak dağıtımına dair stratejik kararlar almak için kullanılmaya başlanmıştır. Özellikle İngiltere ve ABD'deki araştırmacılar radar yerleşimi ve hava

savunma gibi alanlarda matematiksel modellemeyi kullanarak kaynakların en etkin biçimde nasıl kullanılacağına dair analizler yapmıştır (Hillier ve Lieberman 2001).

Savaş sonrasında doğrusal programlama alanında George Dantzig'in 1947'de geliştirdiği Simpleks algoritması ile optimizasyon matematiğinde devrim niteliğinde bir ilerleme kaydedilmiştir. Simpleks algoritması, doğrusal programlama problemlerini çözmede etkili bir yöntem sergileyerek optimizasyon tekniklerinin mühendislik ve ekonomiden lojistiğe kadar geniş bir alanda kullanılmasının önünü açmıştır (Dantzig 1963). Bilgisayar teknolojilerinin özellikle 1950'ler ve 1960'larda gelişmeye başlamasıyla birlikte daha karmaşık problemlerin çözümüne yönelik olarak tam sayılı programlama, doğrusal olmayan programlama ve dinamik programlama gibi yeni optimizasyon teknikleri geliştirilmiştir (Nemhauser ve Wolsey 1988).

1970'lerden itibaren, optimizasyon teknikleri giderek daha geniş bir alanda uygulama bulmaya başlamıştır. Ekonomi, mühendislik, lojistik ve sağlık hizmetleri gibi pek çok alanda yaygın hale gelmiştir. Özellikle 1980'ler ve 1990'larda, büyük ve karmaşık veri setleriyle başa çıkabilmek için sezgisel (heuristic) ve metasezgisel (metaheuristic) yöntemlerin gelişimiyle birlikte optimizasyonda önemli adımların atılması sağlanmıştır. Genetik algoritmalar ve parçacık sürüsü optimizasyonu gibi teknikler, karmaşık sistemlerde etkin çözümler ortaya koyarak optimizasyonun gücünü artırmıştır (Goldberg 1989). Günümüzde optimizasyon, yapay zekâ ve makine öğrenimi gibi ileri teknolojilerle entegre edilmekte, veriye dayalı karar alma süreçlerinde önemli bir yer tutmaktadır.

2.3. Optimizasyon Türleri

Optimizasyon, çeşitli yöntem ve tekniklerin kullanılabildiği geniş bir yelpazeye sahip bir alandır. Burada optimizasyon türlerinin temel kavramları, uygulama alanları ve çeşitli çözüm yöntemleri ele alınacaktır. Matematiksel optimizasyondan sezgisel yöntemlere kadar uzanan bu türler, farklı problem türlerine özgü çözüm yolları sunmaktadır. Söz konusu yöntemlerin genel özellikleri ve kullanım alanları üzerinde yapılacak bir inceleme, optimizasyon sürecine dair daha derin bir anlayış kazanmamıza imkân sağlayacaktır.

Doğrusal optimizasyon, hem amaç fonksiyonunun hem de kısıtlarının doğrusal olduğu özel bir optimizasyon kategorisini ifade etmektedir. Bu tür problemler, matematiksel modelleme ve optimizasyon teknikleri açısından oldukça geniş bir uygulama alanına sahiptir ve genellikle Simplex algoritması gibi gelişmiş doğrusal programlama yöntemleri ile çözülür. Bu yöntem, çözüm uzayındaki her bir doğrusal denklem için optimizasyon sürecini adım adım ilerleterek en uygun çözümü bulmaya çalışır. Fakat, eğer amaç fonksiyonu veya kısıtlarından herhangi biri doğrusal olmayan bir yapıya sahipse, bu durumda problem doğrusal olmayan optimizasyon olarak sınıflandırılır. Doğrusal olmayan optimizasyon, fonksiyonların ve kısıtların doğrusal olmayan matematiksel modelleri içerdiğinden, çözüm yöntemleri genellikle daha karmaşıktır. Bu tür problemler türev tabanlı algoritmalar, çok amaçlı genetik algoritmalar veya sayısal optimizasyon yöntemleri gibi gelişmiş tekniklerle çözülür.

Özellikle değişkenlerin yalnızca tam sayı değerler alabildiği durumlar, tam sayılı optimizasyonun önem kazandığı alanları işaret eder. Bu tür problemler lojistik, üretim

planlaması, rota optimizasyonu ve tedarik zinciri yönetimi gibi uygulama alanlarında sıkça karşımıza çıkar. Tam sayılı optimizasyon, saf tam sayılı ve karışık tam sayılı olmak üzere iki ana grupta incelenebilir. Saf tam sayılı optimizasyon, tüm değişkenlerin tam sayı olmak zorunda olduğu durumları ifade ederken, karışık tam sayılı optimizasyon, bazı değişkenlerin sürekli değerler alırken bazılarının tam sayı olması gereken daha karmaşık yapıları ele alır. Bu tür optimizasyonlar, çözüm alanının sınırlı olması ve çözümün doğrusal olmayan yapıları da içerebilmesi nedeniyle hesaplama açısından daha zorlayıcıdır.

Optimizasyon problemleri çözüm aralığına göre de farklı kategorilere ayrılabilir. Yerel optimizasyon ile yalnızca belirli bir çözüm bölgesinde, genellikle başlangıç noktası etrafında en iyi çözümün elde edilmesi amaçlanır. Bu yöntem, özellikle doğrusal olmayan ve karmaşık problemlerde hızlı ve hesaplama açısından daha etkin çözümler ortaya koyabilir. Ancak bu yöntemde elde edilen yerel optimum çözümler, global optimumdan uzak kalabilir. Buna karşılık global optimizasyon, tüm çözüm uzayını kapsayacak şekilde en uygun çözümü arar ve bu sayede yerel optimallikten kaçınarak daha geniş bir çözüm kümesi üzerinde değerlendirme yapar. Global optimizasyon yöntemleri, özellikle çok sayıda değişkenin ve kısıtın yer aldığı karmaşık problemlerde daha yüksek doğruluklu ve güvenilir sonuçlar elde edilmesini sağlar.

Optimizasyon problemleri, kısıtların varlığına göre de farklı türlere ayrılır. Kısıtlı optimizasyon problemlerinde, çözümün belirli koşulları sağlaması gerekir. Kısıtlar, çözüm uzayını sınırlayan eşitsizlikler veya denklemler olarak modellenebilir. Bu tür problemlerde, çözüm kümesi yalnızca bu kısıtların sağlandığı noktalarla sınırlıdır. Örneğin, bir üretim planlaması probleminde, belirli kaynakların kısıtlı olması üretim miktarlarının sınırlandırılmasını gerektirir. Kısıtsız optimizasyon ise, herhangi bir sınırlama olmaksızın çözüm uzayındaki tüm noktaların potansiyel çözümler olarak değerlendirilebildiği problemlerdir. Kısıtsız problemler genellikle daha basit bir yapıya sahip olup, çözüm bulma süreci daha hızlı gerçekleşebilir. Ancak, gerçekte birçok uygulama problemi kısıtlı optimizasyon kategorisinde yer almaktadır.

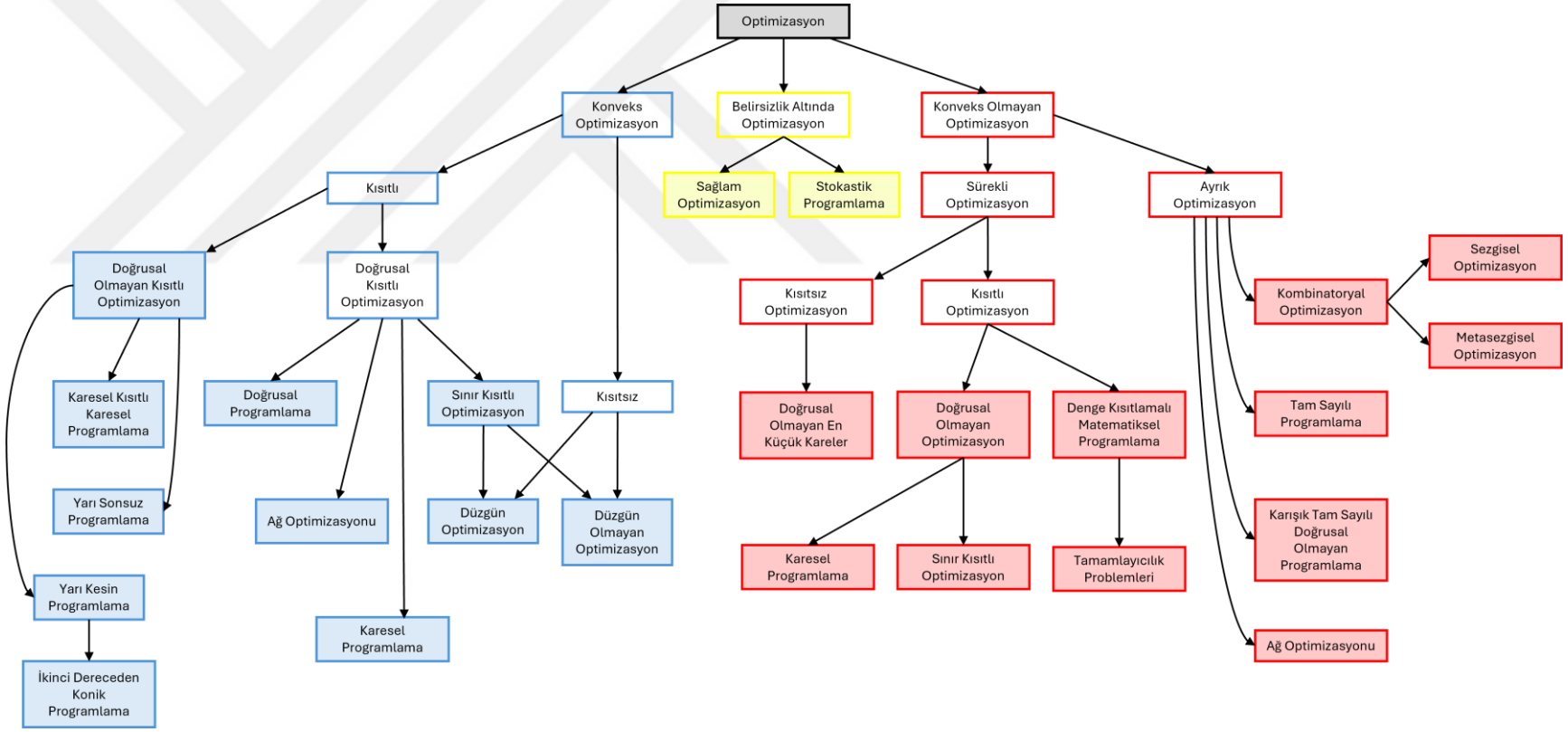
Zamanla değişen ve dinamik yapıya sahip sistemlerin optimize edilmesi ise dinamik optimizasyon alanına girer. Dinamik optimizasyon özellikle kontrol teorisi, ekonomi, mühendislik ve çevresel sistemlerde önemli bir rol oynar. Bu tür optimizasyonlar, zamanla değişen sistem parametrelerinin etkisini hesaba katarak sürekli olarak en uygun kararların alınmasını sağlar. Dinamik optimizasyon, genellikle bir kontrol problemini ifade eder ve bu tür problemlerde sistemin geçmiş ve gelecekteki davranışları göz önünde tutularak kararlar alınır. Örneğin, bir otomobilin hızını kontrol etmek, bir elektrik santralinin enerji üretim düzeyini optimize etmek veya bir şirketin envanter yönetimini iyileştirmek gibi durumlarda dinamik optimizasyon teknikleri tercih edilir.

Son olarak çoklu hedeflerin aynı anda optimize edilmeye çalışıldığı durumlar, çok amaçlı optimizasyon olarak adlandırılır. Bu tür optimizasyonlar, birden fazla çatışan hedefin optimizasyonunu gerektiren problemlerde ortaya çıkar. Bu tür problemlerin çözümünde, genellikle Pareto optimalliği gibi kavramlar kullanılır. Pareto optimalliği, hiçbir hedefin diğer bir hedefin değerini artırmadan iyileştirilemeyeceği bir çözüm kümesini ifade eder. Yani, bir çözüm bir hedefi iyileştirirken diğer hedefleri kötüleştirebilir, ancak Pareto optimalleri, bu tür iyileştirmelerin en verimli dengeyi

sağladığı çözümleri ifade eder. Çok amaçlı optimizasyon yöntemleri mühendislik tasarımı, tedarik zinciri yönetimi, karar destek sistemleri ve stratejik planlama gibi alanlarda geniş bir uygulama alanına sahiptir. Bu tür optimizasyon, karar vericilerin karşılaştıkları çoklu hedefler arasında en uygun çözümün bulunmasına yardımcı olur ve genellikle çeşitli endüstrilerde verimliliği artıran sonuçlar doğurur.

Genel anlamda temel optimizasyon türleri yukarıda ifade edildiği gibi olmakla birlikte literatürde yer alan optimizasyon ailesine ait tüm yöntemler hiyerarşik bir yapıda

Şekil **2.1**'de gösterilmektedir.



Şekil 2.1. Optimizasyon türlerinin hiyerarşik olarak gösterimi (NEOS Guide 2022)

2.4. Metasezgisel Optimizasyon Algoritmaları

Optimizasyon türleri, bir problemi çözmek için en iyi çözümü (örneğin, en düşük maliyet, en yüksek performans gibi) bulmak amacıyla kullanılan yöntemleri ifade eder. Çeşitli alanlarda kullanılabilen optimizasyon türlerinin çözüm yöntemlerine, model yapılarına ve kısıtlara göre farklılık gösterdikleri bir önceki bölümde ayrıntılı olarak ele alınmıştı. Bu yöntemler içerisinde en yaygın kullanılan optimizasyon türlerinden biri de metasezgisel (metaheuristics) optimizasyonlardır. Metasezgisel optimizasyon, özellikle büyük ve karmaşık problemlerin çözümünde optimum sonuçlar elde etmek için kullanılan, sezgisel yöntemleri kapsayan bir optimizasyon dalıdır.

Geleneksel optimizasyon yöntemleri, büyük çözüm uzayında global optimuma ulaşmakta güçlük çekebilir veya işlem maliyetleri oldukça yüksek olabilir. Metasezgisel optimizasyon algoritmaları, geniş bir çözüm uzayını etkili bir şekilde taramak için rastgeleleştirme (randomization), keşif (exploration) ve sömürü (exploitation) stratejilerini kullanır. Bu algoritmalar, çözümün iyileştirilmesi için belirli bir yol izlemek yerine esnek bir çerçeve sunarak, problemin global optimuma yakın çözümler sunmasını amaçlar (Talbi 2009). Bu sayede çok karmaşık problemlerde geleneksel yöntemlere kıyasla daha hızlı ve etkili çözümler elde edilir.

Metasezgisel algoritmaların karakteristiğini belirleyen en temel özellikler şu şekilde sıralanabilir:

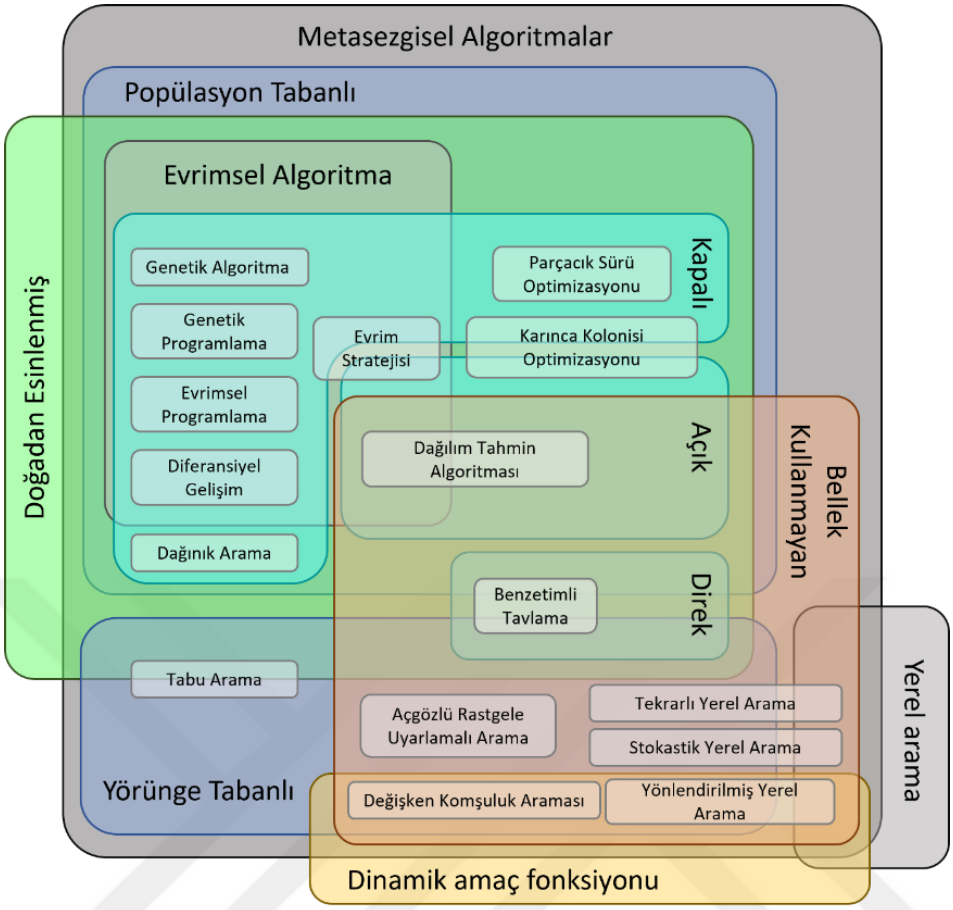
a) **Stokastik yapı:** Metasezgisel algoritmalar rastgele unsurlar içerir ve çözümleri ararken belirsizlikten yararlanır. Bu yapısı sayesinde aynı problem için farklı optimizasyonlarda farklı sonuçlar elde edilebilir. Böylece geniş bir çözüm uzayı taranmış olur.

b) **Problem bağımsızlık**: Belirli bir problem türüne özgü olarak geliştirilmemiştir ve genel yapısı itibariyle farklı türde optimizasyon problemlerine uyarlanabilir. Bu da metasezgisel algoritmaları daha esnek hale getirir.

c) **Keşif ve sömürü dengesi**: Metasezgisel algoritmalar, çözüm uzayında yeni alanların keşfedilmesi ve en iyi çözümü bulma yolunda mevcut çözümlere yoğunlaşarak bilginin sömürülmesi arasında bir denge kurar. Böylece yerel optimumdan çıkabilme ve global optimuma ulaşabilme şansı artar.

d) **Optimumu garanti etmeme**: Metasezgisel algoritmalar problemin global optimumunu garanti etmez. Çoğu durumda yalnızca iyi bir çözüme ulaşmayı hedeflerler ve bu yüzden genellikle "yaklaşık optimizasyon" olarak değerlendirilir.

Metasezgisel algoritmalar farklı yaklaşımlar ve stratejiler ile çeşitli türlere ayrılır. Şekil 2.2'de literatür araştırmalarına göre en çok kabul görmüş olan metasezgisel algoritmaların yapılarına göre sınıflandırılması gösterilmiştir.



Şekil 2.2. Metasezgisel algoritmaların sınıflandırılması (Dreo ve Candan 2011)

Şekil 2.2'de gösterilen metasezgisel algoritma yapılarından en sık kullanılan bazı metasezgisel algoritmalar aşağıda listelenmiştir.

a) **Genetik Algoritmalar (GA):** Evrimsel biyolojiden ilham alarak geliştirilen genetik algoritmalar, popülasyon tabanlı bir metasezgisel algoritmadır. Genetik çaprazlama, mutasyon gibi işlemlerle popülasyonda yeni çözümler oluşturulur. En başarılı çözümler hayatta kalırken, yeni çözümler oluşturulmaya devam edilir (Goldberg 1989).

b) **Parçacık Sürü Optimizasyonu (PSO):** Parçacık sürü optimizasyonu, kuş ve balık sürülerinin kolektif davranışlarından esinlenir. Çözüm adayları olan parçacıklar, sürünün en iyi bilgisi ile yönlendirilen bireylerin en iyi konumlarını güncellemesiyle hareket ederler (Kennedy ve Eberhart 1995).

c) **Benzetilmiş Tavlama (SA):** Benzetilmiş tavlama algoritması, metal soğutma sürecinden ilham alır. Algoritma, yüksek sıcaklıkta rastgele çözümler kabul ederken sıcaklık düştükçe yalnızca daha iyi çözümleri kabul etmeye başlar. Bu sayede yerel optimumlardan kaçınarak daha geniş çözüm alanlarını tarar (Kirkpatrick vd. 1983).

d) **Karınca Kolonisi Optimizasyonu (ACO):** Karınca kolonisi optimizasyonu, karıncaların feromon izlerini takip ederek en iyi çözüm yolunu bulmasına dayanır. Çözüm yolları üzerinde biriken feromon izleri sayesinde en iyi yol belirginleşir ve tercih edilir (Dorigo ve Stützle 2004).

e) **Tabu Arama (TS):** Tabu arama, daha önce ziyaret edilen çözümleri tabu listesi ile işaretleyerek aynı çözüme tekrar ulaşılmasını engeller. Bu yaklaşım, çözüm uzayında yeni alanları keşfetme olasılığını artırır (Glover 1989).

Metasezgisel algoritmalar çok karmaşık veya büyük ölçekli problemlere etkili çözümler sunabilmekte ve birçok farklı türde problemlere uyarlanabilmektedir. Ayrıca geleneksel algoritmalara göre daha hızlı sonuç verebilmektedir. Bu algoritmalar çözüm alanını geniş kapsamlı olarak tarayarak yerel optimuma takılı kalmadan, global optimuma ulaşma olasılığını artırmaktadır. Diğer yandan çoğu metasezgisel algoritma optimum çözümü bulma garantisi vermez, sadece iyi bir çözüm bulmayı amaçlar. Ayrıca amaç fonksiyonunda kullanılan belirli parametrelerin doğru ayarlanması zor olabilir. Metasezgisel algoritmaların başarısı, bu parametrelerin doğru şekilde ayarlanmasına bağlıdır.

Bazı dezavantajlarına rağmen metasezgisel algoritmalar, geleneksel yöntemlerin yetersiz kaldığı problemlerde son derece önemli bir yaklaşımdır ve mühendislikte oldukça geniş bir uygulama alanına sahiptir.

2.5. Genetik Algoritmalar

En yaygın kullanılan metasezgisel algoritmalardan biri olan genetik algoritmalar (GA), biyolojik evrim süreçlerinden ilham alan yapay zekâ temelli bir optimizasyon yöntemidir. İlk olarak John Holland tarafından 1960'larda ortaya atılmıştır ve karmaşık problemlerin çözümünde etkili bir araç olarak kullanılmaktadır. Genetik algoritmalar, evrimsel algoritmaların (EA) en bilinen ve yaygın kullanılan türüdür. Evrimsel algoritmalar, biyolojik evrim süreçlerini taklit eden ve çözüm arayışlarını bu süreçlere dayandıran bir optimizasyon yöntemleri ailesidir. Bu algoritmalar doğal seleksiyon, genetik varyasyon ve adaptasyon gibi biyolojik mekanizmaları kullanarak çözüm arayışı geliştirir. Evrimsel algoritmalar, geleneksel algoritmaların zorlandığı özellikle karmaşık ve doğrusal olmayan problemlerde güçlü bir performans ortaya koyar. Evrimsel algoritmaların temel prensipleri, bir popülasyonun sürekli olarak daha iyi çözümler üretmek amacıyla evrimleşmesini kapsar. Bu popülasyondaki bireyler, çaprazlama, mutasyon ve seçim gibi genetik operatörlerle yeni çözümler üretir. Popülasyondaki her bir birey, genetik bir temsil (kromozom) ile tanımlanır ve üretilen çözüm, genellikle uygunluk fonksiyonu (fitness function) ile değerlendirilir. Evrimsel algoritmalar, genellikle çok büyük çözüm uzaylarında esnek ve etkili arama yapabilme kabiliyetleri nedeniyle tercih edilir.

Genetik algoritmaların kökeni, 1950'lerde bilim insanlarının biyolojik evrimi matematiksel olarak modelleme çabalarına dayanmaktadır. Alan Turing, 1950 yılında kaleme aldığı "*Computing Machinery and Intelligence*" adlı makalesinde ilk kez evrimsel mekanizmalarla öğrenen makineler fikrine değinmişti. 1960'larda Michigan Üniversitesi'nden John Holland, biyolojik evrim süreçlerini bilgisayar bilimlerine uygulamanın yollarını araştırmış ve genetik algoritmaların temel ilkelerini belirlemiştir.

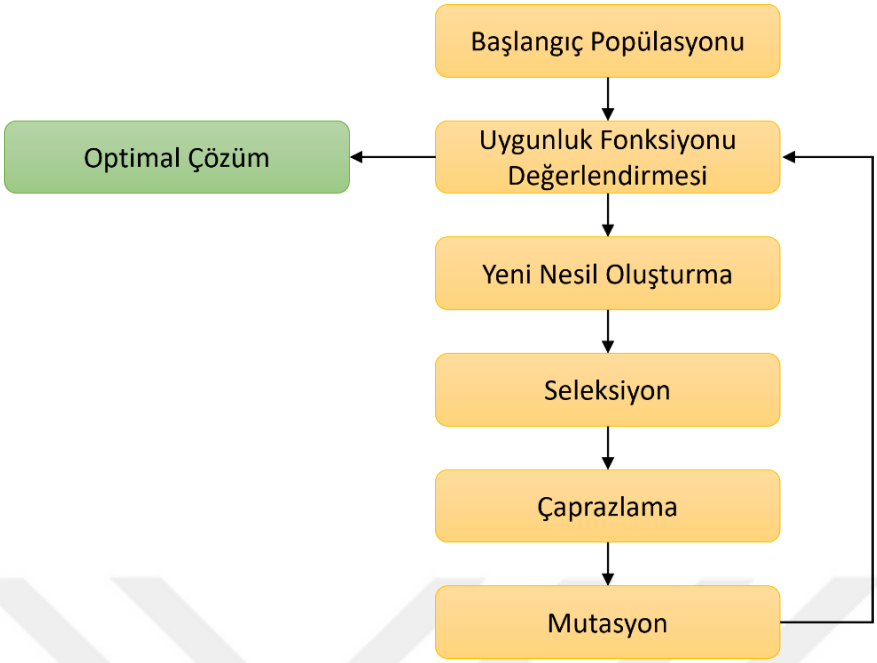
Daha sonra bu çalışması, 1960-1970 yılları arasında Holland'ın öğrencileri tarafından geliştirilmeye devam etmiştir. 1975 yılında Holland, "*Adaptation in Natural and Artificial Systems*" adlı eserini yayınlayarak bu alandaki ilk kapsamlı çalışmalardan birini ortaya koymuş, genetik algoritmaların temellerini atarak sistematik hale getirmiştir (Holland 1975). Bu dönemde, biyolojik evrimdeki adaptasyon ve doğal seçilim gibi süreçler, yapay sistemlere uygulama amaçlı teorik bir çerçeveye dönüştürülmüştür.

1970'ler ve 1980'lerde genetik algoritmalar daha yaygın bir şekilde kullanılmaya başlanmış ve özellikle David Goldberg'in 1989'da yayınladığı "*Genetic Algorithms in Search, Optimization, and Machine Learning*" adlı eseri ile geniş bir kitleye tanıtılmıştır. Goldberg'in çalışması, genetik algoritmaların daha geniş bir uygulama alanına taşınmasını sağlamış, mühendislik, biyoloji ve ekonomi gibi alanlarda kullanıma sunulmuştur (Goldberg 1989). Ayrıca bu dönemde genetik algoritmalar, yerel arama yöntemleri ve diğer optimizasyon teknikleriyle entegre edilerek daha efektif çözümler üretme imkânı elde edilmiştir. Bu hibrit algoritmalar, karmaşık sistemlerin optimizasyonunda önemli bir araç haline gelmiştir (De Jong 1975).

1990'lar sonrasında, genetik algoritmalar gelişmiş yapay zekâ yöntemleri ve büyük veri analitiği ile birleşerek daha karmaşık problemleri çözmede etkin bir araç haline gelmiştir. Özellikle makine öğrenimi ve yapay sinir ağları gibi alanlarla entegrasyonu, bu algoritmaların performansını üst seviyeye çıkarmıştır. Genetik algoritmalar, sadece bireysel problemlerin değil, büyük ve çok katmanlı sistemlerin optimizasyonunda da kullanılmaya başlanmıştır (Mitchell 1996). Günümüzde, bu algoritmalar kuantum bilgisayarlar gibi yeni teknolojilerle birleşerek daha güçlü ve verimli çözümler sunmayı hedeflemektedir. Sonuç olarak, genetik algoritmalar doğadaki evrimsel süreçlerin insan yapımı sistemlerde nasıl uygulanabileceğini gösteren etkili bir yöntem haline gelmiştir (Rechenberg 1973).

Genetik algoritmalar temelde birey (kromozom), popülasyon (population), uygunluk fonksiyonu (fitness function), seçim (selection), çaprazlama (crossover) ve mutasyon (mutation) gibi genetik faktörlerden oluşmaktadır. Çalışma sürecinde genellikle rastgele oluşturulan bireylerden oluşan bir başlangıç popülasyonu ile algoritma başlatılır. Her bireyin uygunluk değeri, çözülmek istenen probleme uygun bir fonksiyon ile hesaplanır. Buna uygunluk fonksiyonu adı verilmektedir ve her bir bireyin performansını ölçmek için kullanılır. Problem çözümüne ne kadar yakınsa, bireyin uygunluk değeri o kadar yüksektir. Uygunluk fonksiyonu yüksek olan bireyler bir sonraki nesil için seçilir. Seçim işlemi, genetik materyalin gelecek nesillere aktarılmasını sağlar. Çaprazlama ile yeni nesil bireyler oluşturulur, ardından bireylerde küçük mutasyonlar yapılarak popülasyon çeşitliliği sağlanır. Mutasyon, popülasyonun çeşitliliğini korumak amacıyla bireylerin genlerinde yapılan küçük rastgele değişikliklerdir ve yerel optimumlara takılmayı önlemeye yardımcı olur. Son adımda yeni nesil, orijinal popülasyondaki bireylerin yerine geçer ve uygunluk fonksiyonları hesaplanarak döngü tekrarlanır. Bu süreç, belirli bir uygunluk değeri sağlanana veya belirli bir iterasyon sayısına ulaşılana kadar devam eder ve bu şekilde optimizasyon işlemi tamamlanır.

Genetik algoritmanın temel işleyişini anlatan optimizasyon adımları sırasıyla Şekil 2.3'te gösterilmektedir.



Şekil 2.3. Genetik algoritmanın akış şeması

Genetik algoritmanın çalışma mantığını daha yakından kavrayabilmek adına bunu basit bir örnekle açıklamak faydalı olacaktır. Bu örnekte mesela kompozit bir yapının maksimum rijitliğini sağlamak için katmanların elyaf oryantasyon açılarını optimize etmek isteyelim. Bu durumda işlem adımları aşağıdaki gibi olacaktır:

a) **Kodlama**: Her bir bireyin geni = Kompozitin katman açıları dizisi olsun. Örneğin [0/45/-45/90/0] gibi.

b) **Uygunluk fonksiyonu**: Mesela maksimum sehim veya rijitlik hesaplanabilir. (Örneğin, analitik yöntemlerle, sonlu elemanlar metodu ile ya da Klasik Katmanlı Plaka Teorisi (CLPT) kullanılarak).

c) **Seçim**: Daha az sehim yapan dizilimler daha yüksek uygunluk değeri alır.

d) **Çaprazlama ve mutasyon**: Katman açıları değiştirilerek yeni bireyler üretilir.

e) **Optimum sonuç**: Maksimum uygunluk değerine sahip birey bulunana kadar döngü tekrarlanır.

Genetik algoritmalar, metasezgisel algoritmalar ailesinin bir üyesi olarak, diğer metasezgisel yöntemlerle birleştirilebilir veya karşılaştırılabilir. Örneğin GA'nın yerel optimizasyonda yavaş kalma problemini çözmek için benzetilmiş tavlama (SA) ile hibrit modeller oluşturulabilir ya da parçacık sürü optimizasyonu (PSO) ile birleştirilerek daha etkili çaprazlama stratejileri geliştirilebilir.

Genetik algoritmaların geniş bir uygulama yelpazesi bulunmakta olup mühendislik, biyoenformatik, yapay zekâ, finans ve lojistik gibi alanlarda etkili bir biçimde kullanılmaktadır. Örneğin, mühendislikte karmaşık sistem tasarımlarını

optimize etmek, lojistikte rota planlaması yapmak ve üretim sistemlerini iyileştirmek için yaygın şekilde uygulanmaktadır. Finansal analizde ise portföy optimizasyonu ve borsa trendlerinin modellenmesinde önemli bir araçtır. Ayrıca, biyoenformatik alanında DNA dizilim analizi ve protein yapısının tahmini gibi problemlerin çözümünde de rol oynar (Emel ve Taşkın 2002; Keklik ve Özcan 2023).

Genetik algoritmaların başarısı, farklı çözüm yaklaşımlarını bir araya getirerek zamanla daha uygun çözümler üretme yeteneğine dayanır. Bu, özellikle geleneksel yöntemlerle çözülmesi zor veya imkânsız olan problemlerde etkili bir alternatiftir. Algoritmanın doğal seçilim, çaprazlama ve mutasyon adımları, çözüm uzayında daha iyi sonuçlar elde etmeyi sağlar. Bu nedenle bu tez kapsamında yapılacak çalışmalarda optimizasyon algoritması olarak genetik algoritmanın kullanılması tercih edilmiştir.

2.6. Kompozitlerin Optimizasyonu

Kompozit malzemelerin optimizasyonu, talep edilen performans isterlerini en iyi şekilde karşılamak için bu malzemelerin özelliklerinin ve yapısal tasarımlarının ayarlanması sürecidir. Kompozitler, matris ve takviye bileşenlerinin bir araya getirilmesiyle oluşan malzemelerdir. Genellikle yüksek mukavemet, hafiflik, dayanıklılık ve tasarım esnekliği gibi avantajları nedeniyle tercih edilirler. Kompozitlerin optimizasyonu ise söz konusu bu özelliklerin farklı uygulamalara göre en uygun hale getirilmesi sürecini kapsamaktadır.

Aşağıda kompozitlerin optimizasyonunda dikkate alınması gereken temel faktörler özet olarak anlatılmaktadır.

a) Malzeme Seçimi

- **Matris malzemesi:** Polimer, metal, seramik veya karbon matrisli kompozitler arasında seçim yapılır. Matris malzemesi, takviye malzemesini bir arada tutan madde olup kompozitin çevresel dayanıklılığını ve şekil değiştirme özelliklerini belirler.

- **Takviye malzemesi:** Genellikle sürekli elyaf, kısa elyaf, parçacık veya nano boyutlu partiküller şeklinde olabilir. Karbon, cam, aramid, bor, alüminyum oksit ve silisyum karbür gibi değişik malzemelerden elde edilebilir. Takviye malzemeleri, kendilerine özgü mekanik dayanım özelliklerini kompozit yapıya kazandırarak güçlendirici vazifesi görür.

b) Katman yapısı ve dizilimler

- Kompozitlerde, takviye malzemesi genellikle çok katmanlı yapılar oluşturacak şekilde tasarlanır. Bu katmanların kalınlığı, sayısı ve elyaf oryantasyon açısı gibi özellikler malzemenin mukavemet, rijitlik ve darbe dayanımı gibi mekanik davranışlarını önemli ölçüde etkiler.

- Elyaf oryantasyon açılarının yönelimi optimizasyonu ve katman dizilimi optimizasyonu, kompozitlerin yük altında gösterdiği davranışı iyileştirmek için en yaygın kullanılan optimizasyon yöntemlerinin başında gelmektedir.

c) Geometri ve yapısal tasarım

- Kompozit malzemelerin geometrik özelliklerini tanımlayan kalınlık, uzunluk, genişlik gibi genel unsurların optimize edilmesi, malzemenin uygulandığı yapıdaki performansını artırır. Ayrıca bu optimizasyonla ağırlığı azaltma veya maliyeti düşürme gibi önemli avantajlar elde edilebilir.

d) Yük altındaki davranış ve mekanik özellikler

- **Mukavemet ve rijitlik:** Kompozitlerin esnekliği, yük altında nasıl bir deformasyon sergilediği ile yakından ilgilidir. Özellikle havacılık ve otomotiv gibi sektörlerde kompozitlerin yüksek mukavemetli ve hafif olması önem arz etmektedir.

- **Hasar ve yorulma davranışı:** Kompozitler, farklı yükleme koşulları altında farklı tipte hasarlar oluşturabilir. Delaminasyon olarak bilinen katmanlar arası ayrışma, elyafların kırılması gibi hasar tiplerinin engellenmesi için optimizasyon işlemleri gerçekleştirilir.

e) İmalat süreci

- Kompozit üretim yöntemleri (örneğin, hand lay-up olarak bilinen elle serim yöntemi, vakum infüzyon veya otoklavlama gibi) malzemenin nihai özelliklerini oldukça etkileyebilir. En uygun imalat yöntemi ve koşullarının belirlenmesi de optimizasyon sürecin bir parçasıdır.

f) Optimizasyon yöntemleri

- **Simülasyon tabanlı analizler:** Sonlu elemanlar analizi (FEA) ve diğer simülasyon araçları, malzemeye uygulanan yüklerin kompozit yapı içerisinde nasıl bir dağılım gösterdiğini belirlemek amacıyla kullanılır.

- **Matematiksel modelleme ve algoritmalar:** Genetik algoritmalar, parçacık sürü optimizasyonu, karınca kolonisi optimizasyonu gibi popülasyon tabanlı yöntemler, kompozitlerin optimum özelliklerini bulmak için en sık tercih edilen tekniklerin başında gelmektedir.

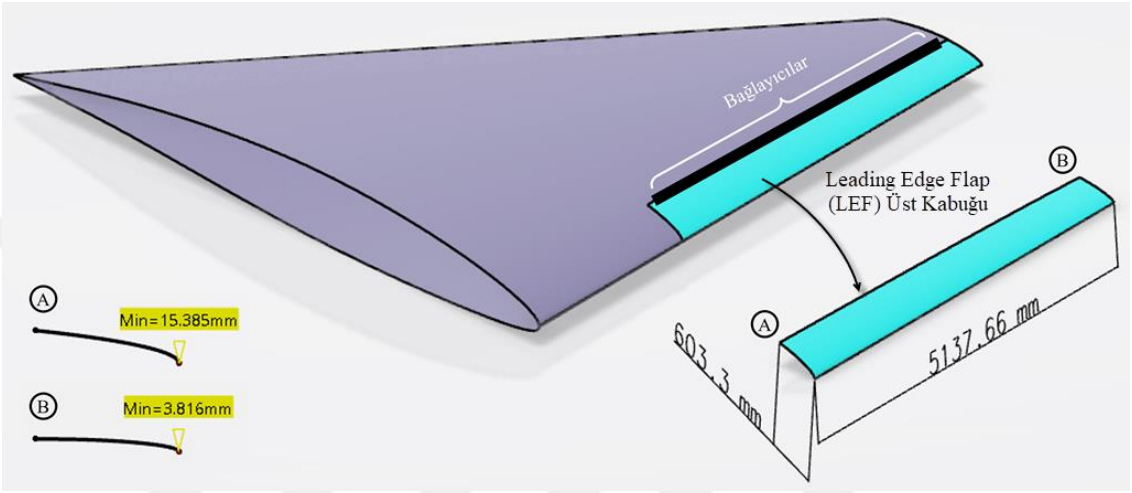
Kompozitlerin optimizasyonu, genellikle endüstriyel ve mühendislik uygulamaları için son derece önem arz etmektedir, çünkü hem malzeme özelliklerini iyileştirir hem de maliyetleri düşürerek verimliliği artırır. Literatür araştırması yapıldığında kompozit malzemelerin özellikle genetik algoritma ile optimizasyonuna dair birçok çalışma yürütüldüğü görülmektedir.

Wang vd. (2016) tarafından yürütülen çalışmada, kompozit dikey eksenli rüzgâr türbini kanatlarının ağırlığını minimize etmek amacıyla tek yönlü katmanların sayısı, kesme profil gövdelerinin (shear web) kalınlığı ve kiriş flanşlarının (spar cap) konumları optimize edilmiştir. Burada amaç fonksiyonunun optimizasyonu için genetik

algoritma uygulanmış ve stres, deformasyon, titreşim, burkulma ve katman yerleşimlerinin sürekliliği olmak üzere beş farklı kısıt hesaba katılmıştır. Elde edilen sonuçlar incelendiğinde, kanatlar optimize edildikten sonra ağırlıkta %17,4 oranında bir azalma olduğu tespit edilmiştir. Ayrıca basma gerilmesi en baskın kısıt olarak ortaya çıkmıştır. Bu nedenle ağırlığı daha da düşürmek için daha yüksek basma mukavemetine sahip yeni bir malzemenin seçilmesi gerektiği sonucuna varılmıştır. Barnes ve Morozov (2016), kompozit rüzgâr türbini kanatlarının iç geometrisini optimize etmek için genetik algoritma kullanmışlardır. Bu kapsamda kiriş flanşlarının genişliği, kesme profil gövdelerinin sayısı ve konumu ile hücum kenarı takviyesinin yeri optimize edilmiştir. Nihai sonuçlara göz atıldığında, bu yapılar optimize edildikten sonra ağırlıkta %3,5 ila %7,4 arasında bir azalma sağlanmıştır. Fagan vd. (2017), cam elyaf takviyeli polipropilenden yapılmış iki kompozit rüzgâr türbini kanadının uzunluğu boyunca kalınlığını optimize etmek için başka bir genetik algoritma tabanlı optimizasyon yaklaşımı kullanmıştır. Bu kanatlardan birinde ek karbon fiber katmanlar yer almaktaydı. Amaç fonksiyonu, ağırlığı en aza indirirken sertliği en üst düzeye çıkarmak şeklinde tanımlandı. Optimizasyon sonucunda, optimize edilmiş kanadın orijinal kanata göre %16 daha hafif ve %30 daha yüksek sertliğe sahip olduğu görülmüştür. Gutierrez-Delgadillo vd. (2015), rüzgâr türbini kanatları olarak kullanılan ankastre kompozit kutu kirişlerin dizilim (stacking) sırasını optimize etmek için genetik algoritmaya başvurmuştur. Amaç fonksiyonu ile söz konusu yapıların kesme gerinimini en aza indirmiştir. Lee vd. (2005), elyaf takviyeli kompozit ile tasarlanan helikopter kanatlarının bükülme deformasyonunu en aza indirmek için genetik algoritma kullanarak oryantasyon açılarını optimize etmeye çalışmıştır. Çalışmasında minimum atalet momenti, doğal frekans ve hasar olmak üzere üç kısıt dikkate alınmıştır. Dillinger vd. (2013), kompozit kanatlarda katmanların rijitlik matrislerini optimize etmek için genetik algoritma kullanarak ağırlığı en aza indirmeyi ve kanatçık verimliliğini en üst düzeye çıkarmayı hedeflemiştir. Çalışmada burkulma ve hasar yükleri kısıt olarak ele alınmıştır. Sonuç olarak, dengesiz tasarımların belirtilen hedefler açısından dengeli tasarımlardan daha yüksek performans gösterdiği ispatlanmıştır. GA tabanlı farklı bir optimizasyon prosedürü uygulayarak Herath vd. (2017), kompozit hidrofoillerin burulma performansını katman dizilimlerini optimize ederek iyileştirdi. Söz konusu yapılar üzerinde çeşitli deneysel testler gerçekleştirilmiş ve bu testler, FEM ile gerçekleştirilen analizleri doğrulamak için kullanılmıştır. Sonuçlar, hidrofoillerin burulma davranışında optimizasyon sonrası %200'lük bir artış yaşandığını ortaya koymuştur. Maes vd. (2019) tarafından yürütülen bir çalışmada, grid takviyeli kompozit panellerin ve kabukların ağırlığı FEM ve GA kombinasyonu kullanılarak minimize edilmiştir. Optimizasyon süreci, burkulma yükü ve yapısal dayanım ile sınırlandırılmıştır. Çalışmada kiriş sayısı, yüksekliği ve genişliği ile kabuk genişliği tasarım değişkenleri olarak tanımlanmıştır. Elde edilen sonuçlara göre, araştırmacılar deneme yanılma yöntemiyle elde edilen tasarımlara kıyasla %20 ağırlık kazancı elde ederek optimum bir tasarıma ulaşabilmişlerdir. Giriprasad vd. (2017), halka destekli katmanlı silindirik kabukların kritik burkulma yükünü maksimize etmek için genetik algoritma kullanmıştır. Tasarım değişkenleri olarak elyaf oryantasyon açılarını almış ve altı katmanlı kabukların optimum dizilimlerini elde etmişlerdir. Burada [0/0/90/45/0/90] açılı dizilime sahip kompozit yapının, optimal sonuçlara göre burkulmaya en az eğilim gösteren yapı olduğu tespit edilmiştir.

3. MATERYAL VE METOT

Bu tez çalışmasında sürekli elyaf takviyeli katmanlı kompozit bir yapının genetik algoritma ile optimizasyonu gerçekleştirilecek olup, söz konusu yapıda minimum sehim (deplasman) elde edilmesini sağlayan en uygun elyaf oryantasyon açılarının hesaplanması amaçlanmaktadır. Kompozit yapı örneği olarak süpersonik akış koşullarına uygun ve günümüz modern savaş uçaklarında yaygın olarak tercih edilen, standart bir aerodinamik profille tasarlanmış temsili bir trapez kanat modeli tasarlanacaktır (Şekil 3.1).



Şekil 3.1. 3B kanat modeli ve LEF bileşenine ait üst kabuk geometrisi

Şekil 3.1'de gösterilen, kanadın hücum kenarındaki eğrilik yarıçapı A kesitinde 15.385 mm'den B kesitinde 3.816 mm'ye azalarak giden ve "Leading Edge Flap" (LEF) olarak adlandırılan yapısal bileşenin üst kabuğu, karbon fiber esaslı kompozit bir malzeme kullanılarak farklı katman dizilimlerinde tasarlanacak ve ardından analiz işlemleri için gerekli olan sonlu elemanlar modeli oluşturulacaktır. GA ile optimizasyon sürecinde, tasarımı gerçekleştirilen katmanların elyaf oryantasyon açıları optimize edilecektir. Burada optimizasyon süreci, yapısal deformasyonun en aza indirgenmesini hedeflemekte olup, bu bağlamda, yapıda meydana gelebilecek en küçük sehim değerine karşılık gelen optimum katman oryantasyon açılarının elde edilmesi amaçlanmaktadır.

3.1. Kompozit Yapının Tasarımı ve Sonlu Elemanlar Modelinin Oluşturulması

Tasarım sürecinin ilk adımında LEF kabuğu için malzemenin tayin edilmesi işlemi yapılacaktır. Burada, havacılık ve uzay endüstrisinde yaygın olarak kullanılan yüksek performanslı bir kompozit malzeme olan M91-IM10 UD karbon prepreg malzemesi kullanılacaktır. Bu malzeme, karbon fiberlerin önceden reçine ile doyurulmuş (prepreg) bir formudur ve özellikle yüksek elastisite modülü ve ağırlık başına mukavemet oranı gerektiren havacılık ve uzay araçlarının primer yapılarında, rijitlik ve hafiflik gereksinimlerinin birlikte karşılanması gibi durumlarda sıklıkla tercih edilmektedir. Malzemenin UD (Unidirectional) tipinde olması, elyafların tek yönde dizildiği bir prepreg formunu ifade etmektedir. Bu da yapıların belirli yönlerde optimize edilmesi gereken yerlerde avantaj sağlamaktadır.

M91-IM10 UD karbon prepreg malzemesinin katman kalınlığı, üretici tarafından hazırlanan teknik bilgi formunda 0.184 mm olarak belirtilmiştir (Hexcel Corporation 2020). Bu malzemeye ait mühendislik sabitleri ise aşağıda Çizelge 3.1'de gösterilmektedir.

Çizelge 3.1. M91-IM10 UD karbon prepreg malzemesinin mekanik özellikleri (Guo vd. 2015)

Malzeme Özelliği	Birim	Değer
$\boldsymbol{E_1}$	GPa	176
$\boldsymbol{E_2}$	GPa	15
$\boldsymbol{G_{12}}$	GPa	5.6
$\boldsymbol{\nu_{12}}$	-	0.27
$\boldsymbol{X_t}$	MPa	3520
$\boldsymbol{X_c}$	MPa	1880
$\boldsymbol{Y_t}$	MPa	42.7
$\boldsymbol{Y_c}$	MPa	127
$\boldsymbol{S}$	MPa	105
$\boldsymbol{\rho}$	kg/m^3	1586

Malzeme seçiminin tamamlanmasının ardından kompozit katman diziliminin ve elyaf oryantasyon açılarının belirlenmesi gerekmektedir. Bunun için üç farklı katman dizilimine sahip kompozit kabuk tasarlanacak ve her biri için ayrı ayrı sonlu elemanlar analizi ve optimizasyon işlemleri gerçekleştirilecektir. Burada, GA ile yapılacak olan optimizasyonlarda elde edilen sonuçların, farklı katman dizilimlerine sahip kompozit kabukların FEM sonuçları ile kıyaslanması amaçlanmıştır.

Literatür araştırmalarına göre en sık kullanılan katman dizilimleri incelenmiş ve aşağıda gösterilen katman dizilimlerinin kullanılmasına karar verilmiştir:

a) $[(45/90/-45/0)_3]_s$ dizilimine sahip 24 katmanlı kompozit kabuk

b) $[(45/-45)_3]_s$ dizilimize sahip 12 katmanlı kompozit kabuk

c) $[(0/90)_2]_s$ dizilimine sahip 8 katmanlı kompozit kabuk

Her üç katman dizilimine ait elyaf oryantasyon açıları sırasıyla Çizelge 3.2, Çizelge 3.3 ve Çizelge 3.4'te gösterilmektedir.

Çizelge 3.2. $[(45/90/-45/0)_3]_s$ dizilimine sahip 24 katmanlı kompozit kabuğun katman dizilimi

Katman No	Elyaf Açısı	Malzeme Türü
1	45°	C/Epoksi
2	90°	C/Epoksi
3	-45°	C/Epoksi
4	0°	C/Epoksi
5	45°	C/Epoksi
6	90°	C/Epoksi
7	-45°	C/Epoksi
8	0°	C/Epoksi
9	45°	C/Epoksi
10	90°	C/Epoksi
11	-45°	C/Epoksi
12	0°	C/Epoksi
13	0°	C/Epoksi
14	-45°	C/Epoksi
15	90°	C/Epoksi
16	45°	C/Epoksi
17	0°	C/Epoksi
18	-45°	C/Epoksi
19	90°	C/Epoksi
20	45°	C/Epoksi
21	0°	C/Epoksi
22	-45°	C/Epoksi
23	90°	C/Epoksi
24	45°	C/Epoksi

Çizelge 3.3. $[(45/-45)_3]_s$ dizilimine sahip 12 katmanlı kompozit kabuğun katman dizilimi

Katman No	Elyaf Açısı	Malzeme Türü
1	45°	C/Epoksi
2	-45°	C/Epoksi
3	45°	C/Epoksi
4	-45°	C/Epoksi
5	45°	C/Epoksi
6	-45°	C/Epoksi
7	-45°	C/Epoksi
8	45°	C/Epoksi
9	-45°	C/Epoksi
10	45°	C/Epoksi
11	-45°	C/Epoksi
12	45°	C/Epoksi

Çizelge 3.4. $[(0/90)_2]_s$ dizilimine sahip 8 katmanlı kompozit kabuğun katman dizilimi

Katman No	Elyaf Açısı	Malzeme Türü
1	0°	C/Epoksi
2	90°	C/Epoksi
3	0°	C/Epoksi
4	90°	C/Epoksi
5	90°	C/Epoksi
6	0°	C/Epoksi
7	90°	C/Epoksi
8	0°	C/Epoksi

Bir sonraki aşama kompozit kabukların toplam kalınlıklarının belirlenmesi işlemidir. 24 katmanlı kompozit kabuk için toplam kalınlık şu şekilde elde edilmektedir:

$$Katman\ sayısı\ x\ Katman\ kalınlığı = \ 24\ x\ 0.184\ mm = 4.416\ mm$$

Optimizasyon sürecinde, 24 katmanlı kabuk için elde edilen bu katman kalınlığı diğer kabuklar için de sabit tutularak optimizasyon işlemleri gerçekleştirilecektir.

Burada katman kalınlığı, optimizasyon ile elde edilecek minimum yapısal deformasyonun üretim kolaylığı ve düşük maliyet gibi faktörler göz önünde tutularak en az sayıda katman ile sağlanması hedeflendiği için sabit tutulmuştur. Bu durumda 12 katmanlı kabukta katman kalınlığı 4.416 mm / 12 = 0.368 mm olarak elde edilir. 8 katmanlı kabuk için ise 4.416 mm / 8 = 0.552 mm olarak elde edilmektedir.

Genetik algoritmada her bir bireyin geni, kompozit katmanların elyaf oryantasyon açıları dizisi olarak kodlandığı için katman dizilimlerinin doğru olarak yapılması kritik bir öneme sahiptir. Bu durumda örneğin 24 katmanlı kabuk için;

$$\textit{Her bir bireyin geni} = [(45/90/-45/0)_3]_s$$

olarak kodlanmaktadır. Bu gen dizilimi, optimizasyon işlemlerinde tasarım değişkenleri olacak kullanılacaktır.

Kompozit yapının tasarım süreci bu şekilde tamamlanmaktadır. Bir sonraki aşama, sonlu elemanlar modelinin oluşturulması ve analize hazırlanması işlemidir.

GA ile optimizasyon işlemlerinde gerçekleştirilecek olan sonlu elemanlar analizlerinde çözücü olarak (FEA solver) açık kaynak kodlu ve ücretsiz Calculix yazılımı kullanılmaktadır. Calculix ile sonlu eleman modelleri oluşturulabilir, çözülebilir ve post işlemleri yapılabilir. Çözücü, doğrusal (linear) ve doğrusal olmayan (non-linear) statik ve dinamik hesaplamalar yapabilir. Isıl analiz, modal analiz ve burkulma gibi analizleri de desteklemektedir. İki ana bileşenden oluşmaktadır; çözücü (solver) olarak asıl hesaplamayı yapan Crunchix (CCX) kısmı ile pre ve post işlemlerini yapan Graphix (CGX) kısmıdır. Python dilinde, CCX bileşenini kullanarak sonlu elemanlar analizi yapılmasını sağlayan pygccx (ya da diğer adıyla pycalculix) ve ccx_dynamic isminde kütüphaneler geliştirilmiştir. Bu çalışmada optimizasyon işlemlerinde ccx_dynamic kütüphanesi kullanılacaktır.

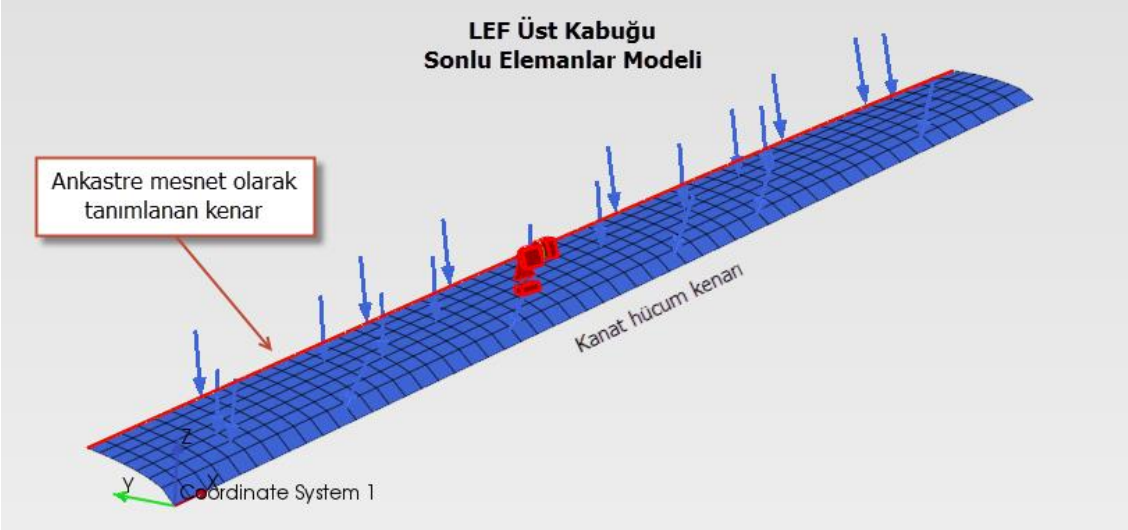
Tez çalışması kapsamında kullanılacak olan bir diğer yazılım yine açık kaynak kodlu ve ücretsiz olan PrePoMax programıdır. PrePoMax aslında çözücü olarak Calculix kullanan bir sonlu elemanlar analizi programıdır ve Calculix için kolay ve esnek bir kullanıcı ara yüzü sunmaktadır. Oldukça kolay bir şekilde sonlu elemanlar analizi yapılabilmesi ve optimizasyon sonuçlarının gerçek verilerle kıyaslanması için FEM analizlerinin gerekli olması sebebiyle bu tezde tercih edilmiştir. Calculix arayüzü olarak pre ve post işlemlerini çok rahat bir şekilde gerçekleştirebilmektedir.

Yukarıda tasarımı gerçekleştirilen 24 katmanlı kompozit kabuğun CAD modeli, Calculix arayüzünde içe aktarılarak kabuk (shell) elemanı şeklinde sonlu elemanlar tipi belirlenmiş ve FE geometrisi elde edilmiştir. Bu işlemin ardından geometriye uygun bir şekilde mesh atma işlemi gerçekleştirilmiş ve böylece sonlu elemanlar modelinin ilk hali oluşturulmuştur. Modelde, x ekseni kanadın hücum kenarını, y ekseni kanat genişliğini ve z ekseni de kanat yüksekliğini gösterecek şekilde lokal bir koordinat sistemi oluşturulmuştur. Kompozit kabuğun sağlıklı bir şekilde analiz edilebilmesi için Çizelge 3.2'de belirtilen katman diziliminin hangi eksene göre yapılacağı son derece önem teşkil etmektedir. İlaveten, söz konusu lokal eksen takımı oryantasyon açılarının optimizasyonunda referans eksen olarak kullanılacaktır.

Sonlu elemanlar modelinde kabuğun yapıya sabit bağlayıcılarla bağlandığı kenar, ankastre mesnet olarak tayin edilerek sınır şartı tanımlanmıştır. Böylece bu kenar boyunca her 3 eksende toplam yer değiştirme ($\Delta x = 0, \Delta y = 0, \Delta z = 0$) ve toplam dönme ($\theta x = 0, \theta y = 0, \theta z = 0$) hareketleri engellenerek tüm serbestlik dereceleri (DOF) kısıtlanmıştır. LEF komponenti normal şartlarda hava araçlarında kanadın kamburluğunu artırarak taşıma kuvvetini artırmaya yarayan hareketli bir uçuş kontrol yüzeyidir ve çoğu zaman yapıya çeşitli lokasyonlarda yer alan bağlantı parçalarıyla bağlanmaktadır. Ancak buradaki optimizasyon sürecinde kolaylık olması açısından ve ana amacın GA ile gerçekleştirilen optimizasyon işlemlerinin etkisini ortaya koymak olduğundan, LEF kabuğunun yapıya sabit bağlayıcılarla bağlandığı ve bu şekilde aerodinamik yüklere maruz kaldığı varsayılmıştır.

Kompozit kabuğa etkiyen aerodinamik yüklerin kabuk boyunca yayılı yük olarak davrandığı varsayılmış olup, 0.001 MPa büyüklüğünde seçilerek uygulanmıştır. Bu basınç yükü, optimizasyon sonucunda birbirleri arasında kıyaslama yapabilmek adına farklı katman dizilimlerine sahip üç farklı kompozit kabuk için aynı büyüklükte uygulanmıştır ve bu nedenle referans bir değer olarak belirlenmiştir.

Son olarak Calculix ara yüzünde kabuğun malzemesi M91-IM10 UD karbon prepreg olarak 24 katman şeklinde tayin edilerek sonlu elemanlar modeli elde edilir. Elde edilen sonlu elemanlar modeli Şekil 3.2'de gösterilmektedir. Aynı işlemler 12 ve 8 katmanlı kompozit kabuklar için de yapılarak sonlu elemanlar modelleri elde edilir.



Şekil 3.2. Kompozit LEF üst kabuğunun sonlu elemanlar (FE) modeli

Optimizasyon sürecinde, genetik algoritmaya performans verisi sağlayabilmek ve tasarım değişkenlerini buna uygun olarak güncelleyebilmek için kompozit kabukların sonlu elemanlar metodu (FEM) ile yükleme ve sınır koşulları altındaki mekanik davranışlarının hesaplanması gerekmektedir. Bu nedenle FEM analizlerine sıklıkla başvurulmaktadır. Bir sonraki kısımda, sonlu elemanlar metodunun matematiksel olarak gelişimi ve katmanlı kompozit yapılarda sonlu elemanlar metodunun temelini oluşturan Klasik Katmanlı Plaka Teorisi'nin (CLPT) ortaya çıkışı hakkında bir inceleme yapılacaktır.

3.2. Klasik Katmanlı Plaka Teorisi ve Sonlu Elemanlar Metodu

Kompozit yapıların analizinde kullanılan sonlu elemanlar metodu (FEM) ve Klasik Katmanlı Plaka Teorisi (CLPT) hakkında genel bir bilgi edinebilmek için çeşitli matematiksel yöntemlere göz atmakta fayda vardır. Bunlardan ilki virtüel yer değiştirmeler prensibidir.

Virtüel yer değiştirmeler prensibi, bir sistemin denge durumunda olduğunu ifade eden varyasyonel bir prensiptir. Bir sistemin denge halinde olması için, sistem üzerindeki iç ve dış kuvvetlerin virtüel işlerinin toplamının sıfır olması gerekir.

Matematiksel olarak virtüel yer değiştirme δu simgesi ile gösterildiğinde, virtüel yer değiştirmeler prensibi Denklem 3.1'deki gibi ifade edilmektedir.

$$\delta W = \int_V \sigma_{ij} \delta \varepsilon_{ij} dV - \int_V f_i \delta u_i dV - \int_S T_i \delta u_i dS = 0 \tag{3.1}$$

Burada;

- $\boldsymbol{\sigma_{ij}}$, gerilme tensörü,
- $\boldsymbol{\epsilon_{ij}}$, şekil değiştirme (deformasyon) tensörü,
- $\boldsymbol{f_i}$, cisim içindeki hacim kuvvetleri (örneğin yerçekimi),
- $\boldsymbol{T_i}$, yüzey üzerinde uygulanan yüzey kuvvetleri,
- $\boldsymbol{\delta u_i}$, virtüel yer değiştirmelerdir.

Bu ifade, elastik cisimlerin denge denklemlerini elde etmek için kullanılan bir varyasyonel prensiptir. Virtüel yer değiştirmeler prensibi, minimum potansiyel enerji prensibiyle doğrudan bağlantılıdır ve Euler-Lagrange denklemlerini türetmek için kullanılır.

Virtüel yer değiştirmeler prensibiyle ilişkili Euler-Lagrange denklemleri, özellikle elastisite teorisi ve sürekli ortam mekaniğinde kullanılır. Bu prensip, küçük deformasyona uğrayan bir cisim için denge denklemlerini türetmek amacıyla uygulanır.

Küçük deformasyonlar için, bir elastik cisimdeki potansiyel enerji fonksiyonu Π olmak üzere, elastik enerji ve dış yüklerden gelen potansiyel enerjilerin toplamı Denklem 3.2'deki gibidir.

$$\prod = \int_V U dV - \int_V f_i u_i dV - \int_S T_i u_i dS \tag{3.2}$$

Burada U, elastik potansiyel enerji yoğunluğudur. Minimum potansiyel enerji prensibine göre, denge durumunda bu fonksiyonun türevi Denklem 3.3 eşitliğinde belirtildiği gibi sıfır olmalıdır.

$$\frac{\delta \prod}{\delta u_i} = 0 \tag{3.3}$$

Bu koşul, Euler-Lagrange denklemlerine götürür ve böylece elastisite teorisinin temel denge denklemleri elde edilir. Bu durumda, Euler-Lagrange denklemleri Denklem 3.4'teki gibi düzenlenebilir.

$$\frac{\partial}{\partial x_j}\left(\frac{\partial U}{\partial \varepsilon_{ij}}\right) - f_i = 0 \tag{3.4}$$

Bu ifade, elastik bir cismin küçük deformasyonlar altındaki denge denklemlerini elde etmemizi sağlamaktadır.

Özetlemek gerekirse, virtüel yer değiştirmeler prensibi, elastik bir cismin denge denklemlerini elde etmek için kullanılır. Bu prensipten türetilen denklemler, potansiyel enerji prensibi ile birleşerek Euler-Lagrange denklemlerine yol açar. Euler-Lagrange denklemleri ise, elastisite teorisinde gerilme ve deformasyon arasındaki ilişkiyi belirlemek için kullanılır.

Diğer yandan kompozit kiriş, plaka ya da kabukların genel denge denklemlerini elde etmekte kullanılan kesin analitik veya varyasyonel yöntemler basit geometrik yapılar için çözüm ortaya koyabilirken, gerçek hayatta sıkça karşılaşılan değişken sınır koşullarına sahip, doğrusal olmayan ve karmaşık geometrilerin çözümünde yetersiz kalmaktadır. Bu nedenle bu durum yaklaştırma (approximation) yöntemlerini kullanmaya itmektedir.

Bu bağlamda sonlu elemanlar metodu (FEM), mühendislik ve uygulamalı bilimlerin çeşitli alanlarında ortaya çıkan diferansiyel ve integral denklemlerin çözümü için güçlü bir yaklaştırma tekniği olarak ortaya çıkmaktadır. Bu yöntem, klasik varyasyonel yöntemlerin bir genellemesidir.

Reddy (2004)'ye göre Denklem 3.4'te ifade edilen temel Euler-Lagrange denge denklemi, katı bir Ω_0 cismi üzerine etkiyen virtüel yer değiştirmeler δu_0, δv_0 ve δw_0 cinsinden yazıldığında Denklem 3.5, Denklem 3.6 ve Denklem 3.7 denge denklemleri elde edilir.

$$\delta u_0 : \frac{\partial N_{xx}}{\partial x} + \frac{\partial N_{xy}}{\partial y} = I_0 \frac{\partial^2 u_0}{\partial t^2} - I_1 \frac{\partial^2}{\partial t^2}\left(\frac{\partial w_0}{\partial x}\right) \tag{3.5}$$

$$\delta v_0 : \frac{\partial N_{xy}}{\partial x} + \frac{\partial N_{yy}}{\partial y} = I_0 \frac{\partial^2 v_0}{\partial t^2} - I_1 \frac{\partial^2}{\partial t^2}\left(\frac{\partial w_0}{\partial y}\right) \tag{3.6}$$

$$\begin{aligned}\delta w_0 : \frac{\partial^2 M_{xx}}{\partial x^2} + 2\frac{\partial^2 M_{xy}}{\partial y \partial x} + \frac{\partial^2 M_{yy}}{\partial y^2} + \mathcal{N}(w_0) + q \\ = I_0 \frac{\partial^2 w_0}{\partial t^2} - I_2 \frac{\partial^2}{\partial t^2}\left(\frac{\partial^2 w_0}{\partial x^2} + \frac{\partial^2 w_0}{\partial y^2}\right) + I_1 \frac{\partial^2}{\partial t^2}\left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial y}\right)\end{aligned} \tag{3.7}$$

$$\mathcal{N}(w_0) = \frac{\partial}{\partial x}\left(N_{xx}\frac{\partial w_0}{\partial x} + N_{xy}\frac{\partial w_0}{\partial y}\right) + \frac{\partial}{\partial y}\left(N_{xy}\frac{\partial w_0}{\partial x} + N_{yy}\frac{\partial w_0}{\partial y}\right) \tag{3.8}$$

Elde edilen bu denge denklemlerinde Ω, uzayda Γ yüzeyi ile çevrili bir bölgeyi, N_{xx}, N_{yy} ve N_{xy} büyüklükleri düzlem içi kuvvet bileşenlerini, M_{xx}, M_{yy} ve M_{xy} büyüklükleri moment bileşenlerini, Q büyüklüğü enine kuvvet bileşkesini ve I_0, I_1 ve I_2 büyüklükleri de atalet momentlerini ifade etmektedir. I_2, dönel eylemsizlik (rotary inertia) olarak da isimlendirilir ve çoğu zaman ihmal edilir.

Yukarıdaki virtüel yer değiştirmeler sırasıyla δu_0, δv_0 ve δw_0 katsayılarıyla çarpılıp Ω^e elementi boyunca integrali alındığında, Denklem 3.9, Denklem 3.10 ve Denklem 3.11 eşitlikleri ile gösterilen zayıf formlar elde edilir.

$$0 = \int_{\Omega^e} \delta u_0 \left[-\frac{\partial N_{xx}}{\partial x} - \frac{\partial N_{xy}}{\partial y} + I_0 \frac{\partial^2 u_0}{\partial t^2} - I_1 \frac{\partial^2}{\partial t^2}\left(\frac{\partial w_0}{\partial x}\right) \right] dxdy \tag{3.9}$$

$$0 = \int_{\Omega^e} \delta v_0 \left[-\frac{\partial N_{xy}}{\partial x} - \frac{\partial N_{yy}}{\partial y} + I_0 \frac{\partial^2 v_0}{\partial t^2} - I_1 \frac{\partial^2}{\partial t^2}\left(\frac{\partial w_0}{\partial y}\right) \right] dxdy \tag{3.10}$$

$$\begin{aligned} 0 = \int_{\Omega^e} \delta w_0 \Bigg[& -\frac{\partial^2 M_{xx}}{\partial x^2} - 2\frac{\partial^2 M_{xy}}{\partial y \partial x} - \frac{\partial^2 M_{yy}}{\partial y^2} - q - \frac{\partial}{\partial x}\left(\widehat{N}_{xx}\frac{\partial w_0}{\partial x} + \widehat{N}_{xy}\frac{\partial w_0}{\partial y}\right) \\ & - \frac{\partial}{\partial y}\left(\widehat{N}_{xy}\frac{\partial w_0}{\partial x} + \widehat{N}_{yy}\frac{\partial w_0}{\partial y}\right) + I_0\frac{\partial^2 w_0}{\partial t^2} \\ & - I_2\frac{\partial^2}{\partial t^2}\left(\frac{\partial^2 w_0}{\partial x^2} + \frac{\partial^2 w_0}{\partial y^2}\right) + I_1\frac{\partial^2}{\partial t^2}\left(\frac{\partial u_0}{\partial x} + \frac{\partial v0}{\partial y}\right) \Bigg] dxdy \end{aligned} \tag{3.11}$$

Zayıf form (weak form), diferansiyel denklemlerin çözümünü daha geniş bir fonksiyon uzayında tanımlayan ve özellikle sonlu elemanlar metodu gibi nümerik tekniklerde temel alınan bir matematiksel yaklaşımdır.

Zayıf formlarda yer alan $\widehat{N}_{xx}$, $\widehat{N}_{xy}$ ve $\widehat{N}_{yy}$ büyüklükleri düzlem içi kenar kuvvetleridir. N_{xx}, M_{xx} gibi stres ve moment bileşkeleri zaten yer değiştirmeler (u_0, v_0, w_0) cinsinden bilinmektedir. Bu arada virtüel yer değiştirmelerin (δu_0, δv_0, δw_0) zayıf formlarda ağırlık fonksiyonlarının rolünü üstlendiği not edilmelidir.

u_0 ve v_0 yer değiştirmeleri Lagrange interpolasyon fonksiyonları kullanılarak, w_0 ise Hermite interpolasyon fonksiyonları kullanılarak Ω^e elementi üzerinden aşağıdaki gibi yakınlaştırılmalıdır.

$$u_0(x,y,t) \approx \sum_{j=1}^{m} u_j^e(t)\psi_j^e(x,y) \tag{3.12}$$

$$v_0(x,y,t) \approx \sum_{j=1}^{m} v_j^e(t)\psi_j^e(x,y) \tag{3.13}$$

$$w_0(x,y,t) \approx \sum_{k=1}^{n} \Delta_k^e(t)\varphi_k^e(x,y) \tag{3.14}$$

Buradaki u_j^e ve v_j^e ifadeleri, Lagrange elementlerinin j'inci nodundaki (u_0, v_0) değerleridir. Δ_k^e ifadesi, w_0 ve onun türevlerinin x ve y cinsinden k'ıncı nodundaki değerleridir. ψ_j^e ve φ_k^e ifadeleri sırasıyla Lagrange ve Hermite interpolasyon fonksiyonlarını ifade etmektedir.

Bu yakınlaştırma metotları, yer değiştirmeler ve virtüel yer değiştirmeler için i'inci interpolasyon fonksiyonları ($\delta u_0 \sim \psi_i$, $\delta v_0 \sim \psi_i$, $\delta w_0 \sim \varphi_i$) cinsinden yazılıp yukarıdaki zayıf formlarda yerine yazıldığında sırasıyla aşağıdaki eşitlikler elde edilir.

$$0 = \sum_{j=1}^{m}\left(K_{ij}^{11}u_j^e + K_{ij}^{12}v_j^e + M_{ij}^{11}\ddot{u}_j^e\right) + \sum_{k=1}^{n}\left(K_{ik}^{13}\Delta_k^e + M_{ik}^{13}\ddot{\Delta}_k^e\right) - F_i^1 - F_i^{T1} \tag{3.15}$$

$$0 = \sum_{j=1}^{m}\left(K_{ij}^{21}u_j^e + K_{ij}^{22}v_j^e + M_{ij}^{22}\ddot{v}_j^e\right) + \sum_{k=1}^{n}\left(K_{ik}^{23}\Delta_k^e + M_{ik}^{23}\ddot{\Delta}_k^e\right) - F_i^2 - F_i^{T2} \tag{3.16}$$

$$0 = \sum_{j=1}^{m}\left(K_{kj}^{31}u_j^e + K_{kj}^{32}v_j^e + M_{kj}^{31}\ddot{u}_j^e + M_{kj}^{32}\ddot{v}_j^e\right) + \sum_{\ell=1}^{n}\left[\left(K_{k\ell}^{33} + G_{k\ell}^e\right)\Delta_\ell^e + M_{k\ell}^{33}\ddot{\Delta}_l^e\right] - F_k^3 - F_k^{T3} \tag{3.17}$$

Burada i = 1, 2, … m; k = 1, 2, …, n sayılarını ifade etmektedir. Rijitlik matrisi $K_{ij}^{\alpha\beta} = K_{ij}^{\beta\alpha}$ ve kütle matrisi $M_{ij}^{\alpha\beta} = M_{ij}^{\beta\alpha}$ (simetrik) katsayıları birbirine eşittir. F_i^α ve $F_i^{T\alpha}$ kuvvet vektörleri ise şu şekilde elde edilir:

$$K_{ij}^{11} = A_{11}S_{ij}^{xx} + A_{16}\left(S_{ij}^{xy} + S_{ij}^{yx}\right) + A_{66}S_{ij}^{yy} \tag{3.18}$$

$$K_{ij}^{12} = A_{12}S_{ij}^{xy} + A_{16}S_{ij}^{xx} + A_{26}S_{ij}^{yy} + A_{66}S_{ij}^{yx} \tag{3.19}$$

$$K_{ij}^{22} = A_{66}S_{ij}^{xx} + A_{26}\left(S_{ij}^{xy} + S_{ij}^{yx}\right) + A_{22}S_{ij}^{yy} \tag{3.20}$$

$$K_{ik}^{13} = -B_{11}R_{ik}^{xxx} - B_{12}R_{ik}^{xyy} - 2B_{26}R_{ik}^{xxy} - B_{16}R_{ik}^{yxx} - B_{26}R_{ik}^{yyy} - 2B_{66}R_{ik}^{yxy} \tag{3.21}$$

$$K_{ik}^{23} = -B_{16}R_{ik}^{xxx} - B_{26}R_{ik}^{xyy} - 2B_{66}R_{ik}^{xxy} - B_{12}R_{ik}^{yxx} - B_{22}R_{ik}^{yyy} - 2B_{26}R_{ik}^{yxy} \tag{3.22}$$

$$K_{k\ell}^{33} = D_{11}T_{k\ell}^{xxxx} + D_{12}\left(T_{k\ell}^{xxyy} + T_{k\ell}^{yyxx}\right) + 2D_{16}\left(T_{k\ell}^{xxxy} + T_{k\ell}^{xyxx}\right) + 2D_{26}\left(T_{k\ell}^{xyyy} + T_{k\ell}^{yyxy}\right) + 4D_{66}T_{k\ell}^{xyxy} + D_{22}T_{k\ell}^{yyyy} \tag{3.23}$$

$$G_{ij}^e = \int_{\Omega^e}\left[\hat{N}_{xx}\hat{S}_{ij}^{xx} + \hat{N}_{xy}\left(\hat{S}_{ij}^{xy} + \hat{S}_{ij}^{yx}\right) + \hat{N}_{yy}\hat{S}_{ij}^{yy}\right]dxdy \tag{3.24}$$

$$M_{ij}^{11} = \int_{\Omega^e} I_0\psi_i^e\psi_j^e\,dxdy \tag{3.25}$$

$$M_{ik}^{13} = -\int_{\Omega^e} I_1 \psi_i^e \frac{\partial \varphi_k^e}{\partial x} dxdy \tag{3.26}$$

$$M_{ij}^{22} = \int_{\Omega^e} I_0 \psi_i^e \psi_j^e dxdy \tag{3.27}$$

$$M_{ik}^{23} = -\int_{\Omega^e} I_1 \psi_i^e \frac{\partial \varphi_k^e}{\partial y} dxdy \tag{3.28}$$

$$M_{k\ell}^{33} = \int_{\Omega^e} \left[I_0 \varphi_k^e \varphi_\ell^e + I_2 \left(\frac{\partial \varphi_k^e}{\partial x} \frac{\partial \varphi_\ell^e}{\partial x} + \frac{\partial \varphi_k^e}{\partial y} \frac{\partial \varphi_\ell^e}{\partial y} \right) \right] dxdy \tag{3.29}$$

$$F_i^1 = \oint_{\Gamma^e} p_x \psi_i^e ds \tag{3.30}$$

$$F_i^2 = \int_{\Gamma_e} p_y \psi_i^e \, ds \tag{3.31}$$

$$F_k^3 = \int_{\Omega^e} q \varphi_k^e dxdy + \oint_{\Gamma^e} (Q_n \varphi_k^e + T_x \frac{\partial \varphi_k^e}{\partial x} + T_y \frac{\partial \varphi_k^e}{\partial y}) ds \tag{3.32}$$

$$F_i^{T1} = \int_{\Omega^e} (\frac{\partial \psi_i^e}{\partial x} N_{xx}^T + \frac{\partial \psi_i^e}{\partial y} N_{xy}^T) dxdy \tag{3.33}$$

$$F_i^{T2} = \int_{\Omega^e} (\frac{\partial \psi_i^e}{\partial x} N_{xy}^T + \frac{\partial \psi_i^e}{\partial y} N_{yy}^T) dxdy \tag{3.34}$$

$$F_k^{T3} = \int_{\Omega^e} \left(\frac{\partial^2 \varphi_k^e}{\partial x^2} M_{xx}^T + 2 \frac{\partial^2 \varphi_k^e}{\partial x \partial y} M_{xy}^T + \frac{\partial^2 \varphi_k^e}{\partial y^2} M_{yy}^T \right) dxdy \tag{3.35}$$

Burada N_{xx}^T, M_{xx}^T, vb. gibi ifadeler termal kuvvet ve moment bileşkelerini ifade etmektedir. Son olarak;

$$S_{ij}^{\xi\eta} = \int_{\Omega^e} \frac{\partial \psi_i^e}{\partial \xi} \frac{\partial \psi_j^e}{\partial \eta} dxdy \tag{3.36}$$

$$\hat{S}_{k\ell}^{\xi\eta} = \int_{\Omega^e} \frac{\partial \varphi_k^e}{\partial \xi} \frac{\partial \varphi_\ell^e}{\partial \eta} dxdy \tag{3.37}$$

$$R_{ik}^{\xi\eta\zeta} = \int_{\Omega^e} \frac{\partial \psi_i^e}{\partial \xi} \frac{\partial^2 \varphi_k^e}{\partial \eta \partial \zeta} dxdy, \tag{3.38}$$

$$T_{k\ell}^{\xi\eta\zeta\mu} = \int_{\Omega^e} \frac{\partial^2 \varphi_k^e}{\partial \xi \partial \eta} \frac{\partial^2 \varphi_\ell^e}{\partial \zeta \partial \mu} dxdy \tag{3.39}$$

eşitlikleri yazılabilir. Burada ξ, η, ζ ve μ değerleri x veya y'ye eşit olabilir. Buraya kadar elde edilen eşitlikler matris formu şeklinde yeniden düzenlenirse Denklem 3.40 bağıntısı elde edilir.

$$\begin{bmatrix} [K^{11}] & [K^{12}] & [K^{13}] \\ [K^{12}]^T & [K^{22}] & [K^{23}] \\ [K^{13}]^T & [K^{23}]^T & [K^{33}] \end{bmatrix} \begin{Bmatrix} \{u^e\} \\ \{v^e\} \\ \{\Delta^e\} \end{Bmatrix} + \begin{bmatrix} [0] & [0] & [0] \\ [0] & [0] & [0] \\ [0] & [0] & [G] \end{bmatrix} \begin{Bmatrix} \{u^e\} \\ \{v^e\} \\ \{\Delta^e\} \end{Bmatrix} + \begin{bmatrix} [M^{11}] & [0] & [M^{13}] \\ [0] & [M^{22}] & [M^{23}] \\ [M^{13}]^T & [M^{23}]^T & [M^{33}] \end{bmatrix} \begin{Bmatrix} \{\ddot{u}^e\} \\ \{\ddot{v}^e\} \\ \{\ddot{\Delta}^e\} \end{Bmatrix} = \begin{Bmatrix} \{F^1\} + \{F^{T1}\} \\ \{F^2\} + \{F^{T2}\} \\ \{F^3\} + \{F^{T3}\} \end{Bmatrix} \tag{3.40}$$

Bu ifade, Klasik Lamine Teorisi (Classical Laminate Theory, CLT) için sonlu elemanlar modelinin geliştirilmesini tamamlamaktadır. Geliştirilen bu sonlu elemanlar modeli, yer değiştirme cinsinden ifade edilen hareket denklemlerine dayandığı için yer değiştirme sonlu elemanlar modeli (displacement finite element model) olarak da tanımlanmaktadır. Son olarak belirtmek gerekir ki, Klasik Katmanlı Plaka Teorisi (Classical Laminated Plate Theory, CLPT) aslında Klasik Lamine Teorisi'nin bir uzantısıdır. Söz konusu geliştirilen bu teoriler ışığında, günümüz bilgisayar ve iş istasyonlarında kullanılan birçok sonlu elemanlar metodu (FEM) yazılımı sayesinde kompozit yapılar için analiz işlemleri başarıyla gerçekleştirilebilmektedir.

3.3. Katmanlı Kompozit Yapıların Genetik Algoritma ile Optimizasyonu

Gerçekleştirilecek olan optimizasyon işlemleri için Python programlama dilinde bu çalışmaya özel kodlar geliştirilmiştir. Burada yazılım dili olarak Python kullanılmasının sebebi, başta Calculix olmak üzere birçok uygulama için kullanılabilen genel amaçlı bir dil olması ve açık kaynak lisansları altındaki geliştiriciler için birçok kütüphane barındırması nedeniyle bu tezde tercih sebebi olmuştur. Söz konusu kütüphanelerden biri de Pymoo kütüphanesidir. Pymoo, İngilizce "Multi-objective Optimization in Python" ifadesinden türetilmiş olup, Python dilinde geliştirilmiş çok amaçlı optimizasyon algoritmalarını barındıran bir kütüphanedir. Bu algoritmalardan biri de bu tez kapsamında kullanılacak olan genetik algoritmadır.

Genetik algoritma ile Calculix çözücüsünün entegrasyonu sağlanarak sonlu elemanlar analizlerini yapabilmek ve optimizasyon işlemlerini gerçekleştirebilmek için kod içerisinde çeşitli fonksiyonlar geliştirilmiştir. Bu kodların tamamı yorum satırlarıyla birlikte bu tez dokümanında EK-1'de paylaşılmıştır. Ancak optimizasyon sürecine ait iş akışını daha iyi kavrayabilmek adına burada bazı kod satırlarını ve fonksiyonları açıklamak faydalı olacaktır.

Pymoo kütüphanesinde yer alan ve Python ile yapılan kodlamada kullanılan en önemli nesneler ile geliştirilen bazı temel fonksiyonlar aşağıda belirtilmiştir.

Problem nesnesi*:* *from pymoo.core.problem import ElementwiseProblem* satırı ile koda eklenir. Optimizasyon sürecinin temelini, bir optimizasyon probleminin

tanımlanması ve amaç fonksiyonunun belirlenmesi oluşturmaktadır. Bu sebeple bir *Problem* nesnesinin oluşturulması ve gerekli parametrelerin tayin edilmesi zorunludur. Pymoo kütüphanesi içerisinde çeşitli tiplerde problem nesneleri mevcut olup burada *ElementwiseProblem* tipi kullanılacaktır. Kod içerisinde bu tipte bir argümanının gönderildiği *Composite_Shell_Problem()* isminde bir sınıf nesnesi yaratılmıştır. Bu sınıf içerisinde, problem nesnesine ait zorunlu parametreler *__init__()* fonksiyonu içerisinde tanımlanmış olup bu parametreler Çizelge 3.5'te gösterilmektedir.

Çizelge 3.5. Problem nesnesine ait parametreler ve açıklamaları

Parametre	Değer	Açıklama
n_var	Elyaf oryantasyon açıları sayısı	Tasarım değişkeni sayısı (Integer)
n_obj	1	Amaç fonksiyonu sayısı (Integer)
n_ieq_constr	1	Eşitsizlik kısıtlamalarının sayısı (Integer)
xl	-90	Tasarım değişkenleri için alt sınır (Float)
xu	90	Tasarım değişkenleri için üst sınır (Float)

Burada *n_var* parametresi, ileride detaylı olarak açıklanacak olan katmanlı kompozit yapıdaki elyaf oryantasyon açılarının sayısı olarak belirlenecektir. Bu durumda oryantasyon açıları, tasarım değişkenleri olarak tayin edilecektir. *n_obj* parametresi 1 adet amaç fonksiyonunu, *n_ieq_constr* parametresi ise eşitsizlik içeren 1 adet kısıt fonksiyonunu belirtmektedir. *xl* ve *xu* parametreleri, oryantasyon açıları için alt ve üst limit değerlerini ifade etmekte olup optimizasyon esnasında bu açıların -90° ile +90° arasından seçilmesini sağlamaktadır.

Amaç fonksiyonu ve eşitsizlik içeren kısıt fonksiyonu, kod içerisinde sırasıyla *Objective_Function()* ve *Constraint_Function()* isimleriyle ayrı birer fonksiyon olarak tanımlanmıştır. Bu fonksiyonlar problem nesnesine ait *_evaluate()* fonksiyonunun içerisinden çağrılmaktadır. Burada;

- Amaç fonksiyonu → $f(x) = Maks(sehim)$
- Kısıt fonksiyonu → $g(x) = Tsai - Hill\ hata\ kriteri < 1$

olmak üzere $f(x)$ amaç fonksiyonu, sonlu elemanlar metodu ile gerçekleştirilen analiz sonucunda kompozit kabukta meydana gelen maksimum sehim değerini ifade etmektedir. Optimizasyon işlemi ile bu değer minimize edilecektir. Kısıt fonksiyonu olarak tanımlanan $g(x)$ fonksiyonu ile optimizasyon işleminin arama uzayı sınırlandırılmaktadır. Burada, sonlu elemanlar analizi ile gerçekleştirilen hesaplamalarda Tsai-Hill hata kriterinin 1'den küçük olması kısıtı getirilmektedir. Kompozitlerin farklı yönlerde farklı mekanik özellik gösteren anizotropik yapıda olması sebebiyle izotropik yapı için kullanılan klasik Von Mises kriteri yetersiz kalmaktadır. Von Mises kriterinden uyarlanmış Tsai-Hill kriteri, çok eksenli gerilme durumlarında

elyaf yönünde veya elyafa dik yönlerde oluşabilecek hasarları göz önünde bulundurarak bunları tahmin etmede kullanılan bir hata kriteridir. Matematiksel olarak Denklem 3.41'de gösterildiği gibi ifade edilmektedir.

$$\left(\frac{\sigma_1}{X_t}\right)^2 - \left(\frac{\sigma_1\sigma_2}{{X_t}^2}\right) + \left(\frac{\sigma_2}{Y_t}\right)^2 + \left(\frac{\tau_{12}}{S}\right)^2 < 1 \tag{3.41}$$

Bu denklemde yer alan ifadelerin açıklamaları ise şu şekildedir:

- $\boldsymbol{\sigma_1}$: Elyaf yönündeki normal gerilme
- $\boldsymbol{\sigma_2}$: Elyaf yönüne dik normal gerilme
- $\boldsymbol{\tau_{12}}$: Düzlem içi kayma gerilmesi
- $\boldsymbol{X_t}$: Elyaf yönündeki çekme mukavemeti
- $\boldsymbol{Y_t}$: Elyaf yönüne dik çekme mukavemeti
- $\boldsymbol{S}$: Düzlem içi kayma mukavemeti

Tsai-Hill kriterinde elde edilen sonucun < 1 olması durumunda yapı hasarsız olarak kabul edilirken, ≥ 1 olması durumunda ise ilgili katmanda hasar meydana geldiği kabul edilmektedir.

Optimizasyon esnasında amaç ve kısıt fonksiyonları ile elde edilen sonuçlar, kütüphane (*dictionary*) tipindeki *out* değişkenlerine aktarılmaktadır.

out["F"] = Objective_Function(Results)

out["G"] = Constraint_Function(Results)

Minimize fonksiyonu: *from pymoo.optimize import minimize* satırı ile koda eklenir. Pymoo kütüphanesindeki herhangi bir algoritmayı kullanarak optimizasyon işlemine başlayabilmek için *minimize()* olarak isimlendirilen bir fonksiyon ara yüzü kullanılmaktadır. Bu fonksiyon içerisinde problem nesnesi, algoritma nesnesi, *n_gen* parametresiyle nesil/iterasyon sayısı ve *seed* parametresiyle de rastgelelik (randomness) gibi önemli parametreler tayin edilmektedir. Pymoo kütüphanesinin en temel fonksiyonu olmakla birlikte optimizasyon sürecinin başlatılmasını sağlamaktadır. Kod içerisinde *Run_Minimizer()* fonksiyonu içerisinden çağrılmaktadır.

Algoritma nesnesi: *from pymoo.algorithms.moo.nsga2 import NSGA2* satırı ile koda eklenir. Pymoo kütüphanesi içerisinde, temel amacımız olan optimizasyon işleminin gerçekleşmesini sağlayan kısıtlı ya da kısıtsız, tek amaçlı ya da çok amaçlı olmak üzere çok sayıda algoritma yer almaktadır. Bu tez çalışmasında genetik algoritma kullanılacağından dolayı hangi tipte olduğu belirtilmelidir. Pymoo içerisinde çeşitli sayıda genetik algoritma tipleri mevcut olup bunlardan NSGA2 tipi kullanılacaktır. Genetik algoritma ile NSGA (Non-dominated Sorting Genetic Algorithm) hakkında biraz detaylı bilgi edindikten sonra aralarındaki temel farklardan bahsetmek faydalı olacaktır.

"Baskın Olmayan Sıralama Genetik Algoritması" olarak adlandırılan NSGA algoritması, genetik algoritmalara dayalı çok amaçlı optimizasyon problemlerini çözmek için kullanılan bir evrimsel algoritmadır. NSGA, birden fazla amaç

fonksiyonuna sahip optimizasyon problemlerinde Pareto optimal çözümleri bulmayı amaçlar. NSGA algoritmasının NSGA-II olarak geliştirilen farklı bir versiyonu bulunmakta olup, bu versiyonda NSGA algoritmasına göre sıralama ve yoğunluk mesafesi gibi hesaplamalar daha hızlı gerçekleştirilerek daha verimli hale getirilmiştir.

NSGA ve NSGA-II'nin temel ilkeleri şu şekildedir:

a) Pareto Önceliklendirme (Non-dominated Sorting)

- NSGA, bireyleri baskın olmayan (non-dominated) çözümlere göre sıralar.
- Bir çözüm, başka bir çözüm tarafından tüm hedeflerde aynı anda daha iyi olmadığı sürece baskın olmayan (non-dominated) olarak kabul edilir.
- Pareto sınırı (Pareto front), baskın olmayan çözümlerden oluşur ve en iyi çözümleri temsil eder.

b) Yoğunluk Mesafesi (Crowding Distance)

- NSGA, çözümleri çeşitlendirmek ve iyi dağılım sağlamak için yoğunluk mesafesi yöntemini kullanır.
- Bu, birbirine çok yakın çözümlerin önüne geçerek çeşitli çözümleri korumaya yardımcı olur. Amaç, çözümlerin sadece optimal olması değil, aynı zamanda amaç fonksiyonu uzayında yayılmış ve çeşitli olmalarını sağlamaktır.
- Yoğunluk mesafesi, her çözümün çevresindeki yoğunluğu ölçer. Daha az kalabalık bölgelerdeki çözümler daha avantajlı kabul edilir.

c) Turnuva Seçimi (Tournament Selection)

- Seçilen bireyler, Pareto seviyesi ve yoğunluk mesafesine dayalı olarak turnuva seçim yöntemiyle bir sonraki nesle aktarılır. Amaç, yeni nesil için ebeveyn bireyleri seçmektir.
- Popülasyondan rastgele belirli sayıda birey seçilir. Bu bireyler kendi aralarında bir turnuva yapar. En iyi uygunluk değerine sahip birey (ya da NSGA-II algoritmasında, öncelikle daha düşük pareto seviyesi ve sonra daha yüksek yoğunluk mesafesi) turnuvayı kazanır ve seçilir.

NSGA ile GA arasındaki en temel fark, optimizasyonun hedefleri ve çözüm seçim stratejileriyle ilgilidir.

Genetik algoritma genellikle tek bir amaç fonksiyonunu optimize etmek için kullanılır. Örneğin, bir problemi çözmek için maliyeti minimize etmek ya da performansı maksimize etmek gibi tek bir kriter belirlenir. NSGA ve NSGA-II, çok amaçlı optimizasyon için geliştirilmiştir. Birden fazla çelişkili amaç fonksiyonunu aynı anda optimize eder ve Pareto optimal çözümleri üretir. GA, tek bir uygunluk değeri üzerinden seçim yapar. Genellikle en iyi çözümü bulmaya çalışır. NSGA, baskın olmayan çözüm adı verilen bir yöntem kullanarak çözümleri Pareto ön sınırına göre

sıralar ve yoğunluk mesafesi yöntemi ile çeşitliliği korur. GA, genellikle elitizm ve mutasyon mekanizmaları ile çeşitliliği korumaya çalışır. Ancak, bazen yerel minimuma sıkışabilir. NSGA, yoğunluk mesafesi ile farklı bölgelerde çözümler üreterek geniş bir çözüm kümesi sunar. Genetik algoritma ve NSGA arasındaki temel farklar Çizelge 3.6'da özetlenmiştir.

Çizelge 3.6. Genetik algoritma ve NSGA arasındaki temel farklar.

Özellik	Genetik Algoritma (GA)	Baskın Olmayan Sıralama Genetik Algoritması (NSGA)
Amaç Fonksiyonu	Tek hedefli optimizasyon	Çok hedefli optimizasyon
Çözüm Kümesi	Tek bir en iyi çözüm elde edilir.	Pareto optimal çözümler kümesi elde edilir.
Seçim Kriteri	Tek bir uygunluk değeri üzerinden seçim yapılır.	Çözümler baskın olmayan çözüm ile sıralanır.
Evrim Mekanizması	Uygunluk fonksiyonu en iyi bireyleri belirler.	Pareto seviyeleri ve Yoğunluk Mesafesi kullanılır.
Çeşitlilik Kontrolü	Genellikle elitizm ve mutasyon operatörleri ile sağlanır.	Yoğunluk mesafesi ile çeşitlilik korunur.
Sonuçların Dağılımı	Tek bir noktaya yakınsama eğilimindedir.	Çeşitli Pareto optimal çözümler bulunur.

Sonuç olarak NSGA-II, hem genetik algoritma hem de NSGA'ya göre daha iyi hesaplama verimliliği ve çeşitlilik kontrolü sağlamakta olup, günümüzde çok amaçlı optimizasyon problemlerinde en yaygın kullanılan algoritmalardan biridir.

Kod içerisinde *algorithm* nesnesiyle oluşturulan NSGA2 algoritmasına girdi olarak, popülasyon büyüklüğünü belirten *pop_size* parametresi gönderilmektedir.

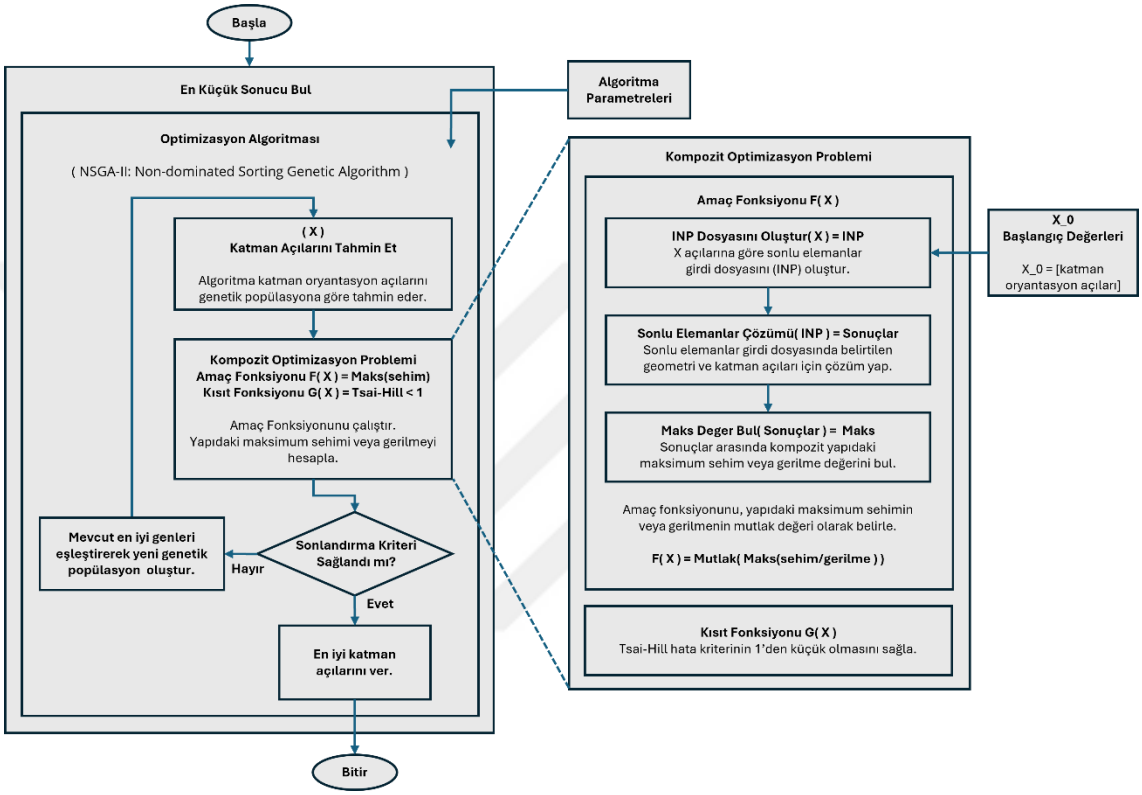
Calculix fonksiyonu: *Run_Calculix()* olarak isimlendirilen bu fonksiyon ile Calculix çözücüsü bir alt işlem (subprocess) olarak çağrılır ve sonlu elemanlar analizi işlemleri gerçekleştirilir. Fonksiyonun sonlanması için analiz işlemlerinin tamamlanması beklenmektedir.

Girdi dosyası fonksiyonu: *Generate_Inp_File()* fonksiyonu aracılığıyla Calculix çözücüsü için girdi dosyası hazırlanır. Optimizasyon sürecinde bu fonksiyona her iterasyonda elyaf oryantasyon açıları yeniden gönderilerek girdi dosyasının içeriği her defasında güncellenmektedir. Burada aynı zamanda kompozit yapının katman kalınlığı *h_layer* parametresiyle tanımlanmaktadır.

Tsai-Hill hata kriteri fonksiyonu: *Tsai_Hill_Cr()* fonksiyonu ile her bir kompozit katman için Tsai-Hill hata kriteri hesaplanmaktadır. Bu fonksiyonda aynı zamanda kompozit malzemenin mukavemet değerleri tayin edilmektedir.

Sonuç nesnesi: Optimizasyon işleminin tamamlanmasının ardından *minimize()* fonksiyonu ile *res_min* isminde bir sonuç nesnesi geri döndürülür.

Optimizasyon sürecine ait akış şeması Şekil 3.3'te gösterilmektedir.



Şekil 3.3. Optimizasyon sürecine ait akış şeması

Akış şemasında gösterilen işlem maddeleri detaylı olarak şu şekilde sıralanmaktadır:

a) Başlangıç adımında sonlu elemanlar analizi için gerekli olan girdi dosyası oluşturulur.

   Girdi dosyası, Calculix üzerinden sonlu elamanlar analizini başlatan ccx_dynamic kütüphanesi için zorunlu bir parametre olduğundan dolayı en başta bu dosyanın hazırlanması gerekmektedir. Bu işlem için CAD ortamında tasarlanan kompozit yapıya ait üç boyutlu model geometrisi, Calculix ara yüzünde *STEP* formatında içe aktarılarak (import) burada sonlu elemanlar modeli elde edilir. Bu model üzerinde uygun bir mesh yapısı oluşturulur ve ardından malzeme özellikleri, katman dizilimi, sınır koşulları ve yük koşulları tanımlanarak sonlu elemanlar analizine hazır hale getirilir. Ardından Calculix ara yüzünün dışa aktarma (export) özelliği ile *Calculix *.inp* formatında girdi dosyası elde edilir.

b) Optimizasyon süreci başlatılarak Calculix ile sonlu elemanlar analizi gerçekleştirilir.

   Python kodunda algoritma parametrelerinin tanımlandığı *Run_Minimizer()* fonksiyonu çağrılarak rastgele oluşturulan bireylerden oluşan bir başlangıç popülasyonu ile optimizasyon süreci başlatılır. Kompozit optimizasyon problemi içerisinden amaç fonksiyonu çalıştırılır ve Calculix ile birinci adımda hazırlanan başlangıç parametrelerine sahip girdi dosyası kullanılarak sonlu elemanlar analizi başlatılır. Calculix çözücüsü *Run_Calculix()* fonksiyonu tarafından bir alt işlem olarak çalıştırılır ve sonlu elamanlar analizi gerçekleştirilir. Her iterasyonda analiz işlemlerinin tamamlanması beklenmektedir. Analiz sonucunda mesh yapısındaki her bir düğüm noktası (node) için koordinat, sehim ve stres sonuçları elde edilir ve sonuçlar arasında kompozit yapıdaki maksimum sehim değeri bulunur.

c) Genetik algoritma ile optimizasyon işlemi gerçekleştirilir.

   Analiz işleminin tamamlanmasının ardından elde edilen sonuçlar başlangıç popülasyonuyla birlikte GA'ya gönderilerek uygunluk değerlendirme adımı ile optimizasyon algoritması çalıştırılır. Burada her bir bireyin uygunluk değeri, amaç fonksiyonu ile hesaplanarak daha az sehim yapan bireyler daha yüksek uygunluk değeri alır. Uygunluk fonksiyonu yüksek olan bireyler bir sonraki nesil için seçilir. Ardından çaprazlama ve mutasyon işlemleri ile yeni nesil bireyler oluşturularak bir önceki popülasyondaki bireylerin yerine geçer. Katman oryantasyon açıları değiştirilerek uygunluk fonksiyonları tekrar hesaplanır ve döngü bu şekilde devam eder. Bu işlem sonlandırma kriterinde belirtilen nesil/iterasyon sayısına ulaşana kadar devam eder.

d) Optimum sonucun elde edilmesiyle optimizasyon işlemi tamamlanır.

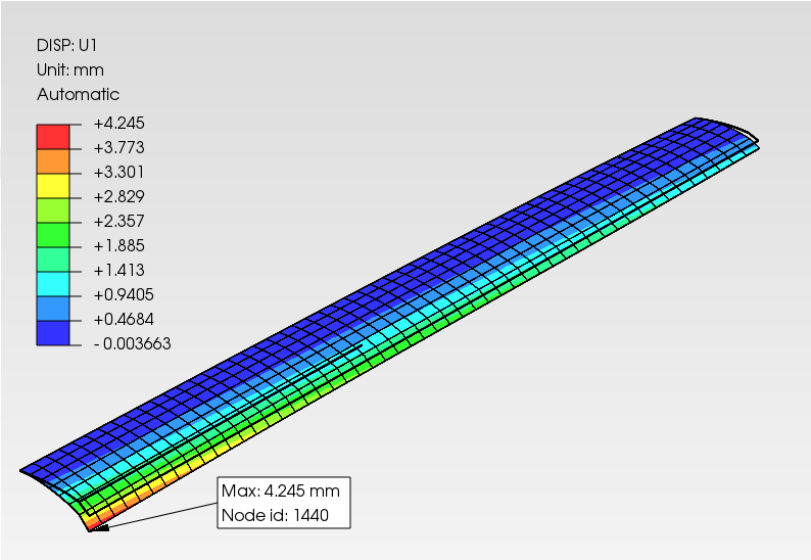
   n_gen değişkeniyle belirtilen nesil/iterasyon sayısına ulaşılmasıyla birlikte sonlandırma kriteri sağlanır ve ardından çözüm uzayı içerisindeki en küçük sehim değerine karşılık gelen en uygun katman açılarının elde edilmesiyle optimizasyon işlemi tamamlanır.

Burada optimizasyon işlemi en yüksek uygunluk değerine sahip optimal çözümü sunmaktadır. Ancak elde edilen optimizasyon sonucunun, farklı katman dizilimlerine sahip kompozit kabukların sonlu elemanlar analizi sonuçlarıyla kıyaslanması gerekmektedir. Böylece aslında GA'nın tam olarak ne kadar katkıda bulunup bulunmadığı daha kolay anlaşılacaktır. Bundan sonraki adımlarda kompozit kabukların sonlu elemanlar analizleri yapılacak ve ardından genetik algoritma ile optimizasyon işlemleri gerçekleştirilerek karşılaştırma yapılacaktır.

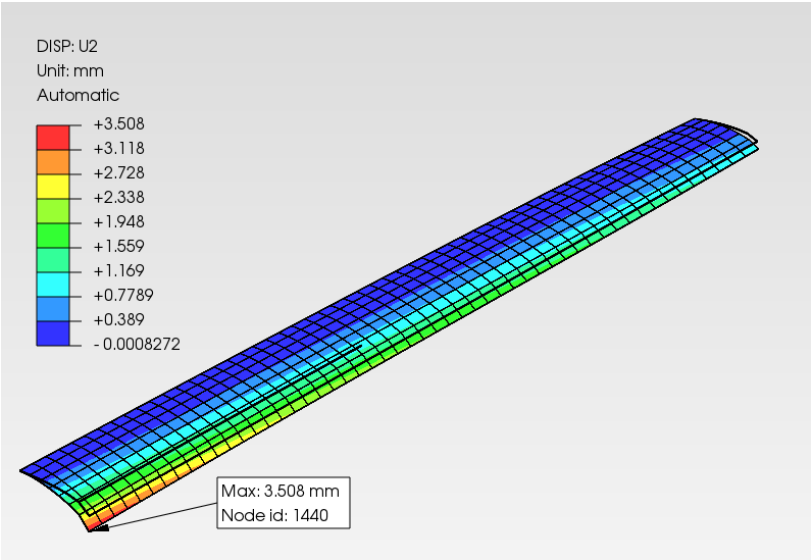
4. BULGULAR VE TARTIŞMA

4.1. Katmanlı Kompozit Yapıların Sonlu Elemanlar Analizi

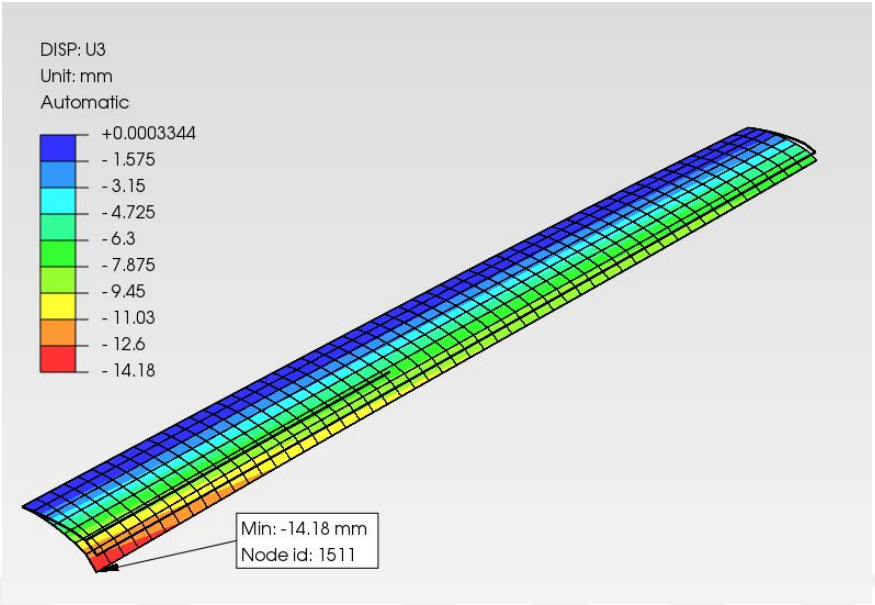
Genetik algoritma ile optimizasyon işlemlerine başlamadan önce kompozit kabukların sonlu elemanlar analizleri gerçekleştirilecektir. Buradaki temel amaç, elde edilen analiz sonuçlarının optimizasyon ile elde edilen sonuçlarla kıyaslanmasıdır. Calculix ara yüzü ile üç farklı katman dizilimine sahip kompozit kabuk için sonlu elemanlar analizleri koşturulduktan sonra x (U1), y (U2) ve z (U3) yönlerinde elde edilen maksimum yer değiştirmeler sırasıyla Şekil 4.1, Şekil 4.2 ve Şekil 4.3'te gösterilmektedir.



Şekil 4.1. $[(45/90/-45/0)_3]_s$ dizilimine sahip 24 katmanlı kompozit LEF üst kabuğunun U1 yönündeki maksimum yer değiştirmesi

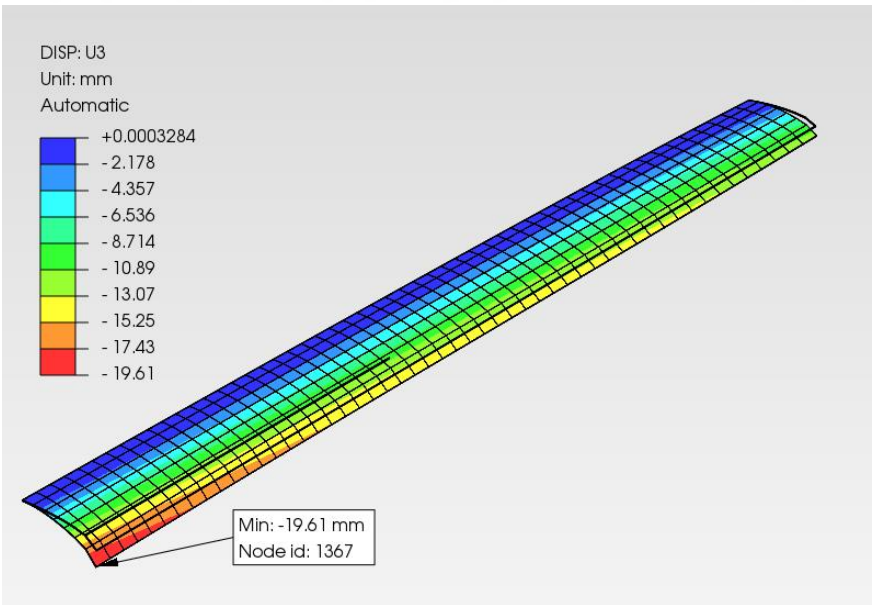


Şekil 4.2. $[(45/90/-45/0)_3]_s$ dizilimine sahip 24 katmanlı kompozit LEF üst kabuğunun U2 yönündeki maksimum yer değiştirmesi

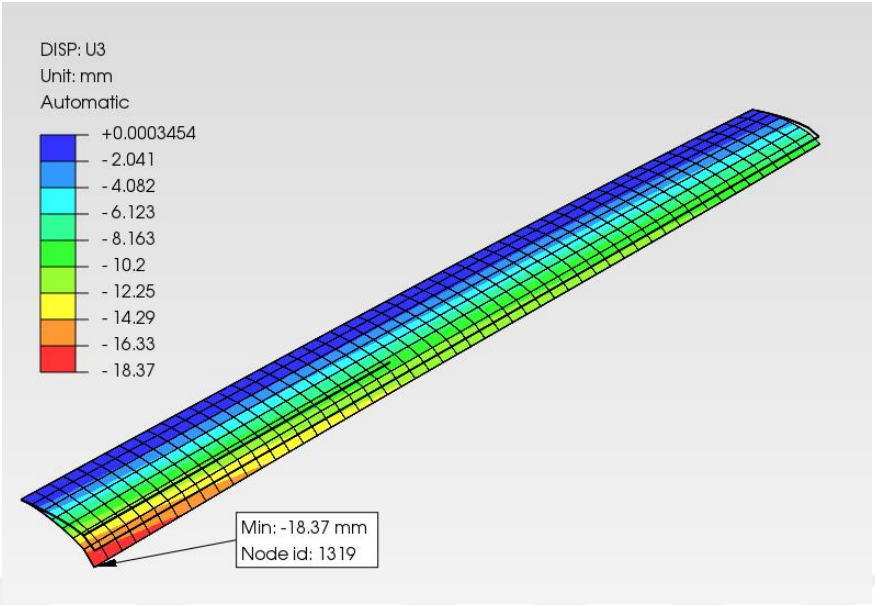


Şekil 4.3. $[(45/90/-45/0)_3]_s$ dizilimine sahip 24 katmanlı kompozit LEF üst kabuğunun U3 yönündeki maksimum yer değiştirmesi

Elde edilen sonuçlara göre maksimum yer değiştirmenin kanat köküne yakın tarafta, hücum kenarında ve -z yönünde (U3) 14.18 mm olduğu görülmektedir. Analiz işlemi, $[(45/-45)_3]_s$ dizilimine sahip 12 katmanlı ve $[(0/90)_2]_s$ dizilimine sahip 8 katmanlı kompozit kabuk için koşturulduğunda, maksimum yer değiştirmeler yine U3 yönünde sırasıyla 19.61 mm ve 18.37 mm olarak elde edilmiştir (Şekil 4.4 ve Şekil 4.5).



Şekil 4.4. $[(45/-45)_3]_s$ dizilimine sahip 12 katmanlı kompozit LEF üst kabuğunun U3 yönündeki maksimum yer değiştirmesi



Şekil 4.5. $[(0/90)_2]_s$ dizilimine sahip 8 katmanlı kompozit LEF üst kabuğunun U3 yönündeki maksimum yer değiştirmesi

Elde edilen analiz çıktılarına göre maksimum yer değiştirmeler U3 yönünde elde edildiği için optimizasyon işlemlerinde bu yöndeki değerler optimize edilmiştir. U1 ve U2 yönündeki yer değiştirmeler ihmal edilebilir seviyede olduğundan hesaba katılmamıştır.

4.2. Genetik Algoritma ile Optimizasyon İşlemlerinin Başlatılması

Bu kısımda Python kodu koşturularak GA ile optimizasyon işlemleri gerçekleştirilecektir. Optimizasyon işlemleri, her defasında farklı parametrelerle koşturulacak ve elde edilen sonuçlar tablo halinde paylaşılacaktır. Burada farklı parametreler test edilerek aynı zamanda GA'nın elde ettiği sonuçların keskinliği tespit edilmeye çalışılacaktır. Örneğin, başlangıç popülasyonu farklı olan optimizasyon sonuçları arasında ne kadar fark olduğu ya da farklı nesil/iterasyon sayılarının optimizasyon sonucunu ne kadar etkilediği gibi sorulara cevap aranacaktır. Bu nedenle optimizasyon sürecini etkileyen en kritik parametrelerin belirlenmesi gerekmektedir. Bu parametreler şu şekilde seçilmiştir:

- **pop_size** : Popülasyon sayısını belirtmektedir.
- **n_gen** : Nesil / iterasyon sayısını belirtmektedir.
- **n_var0** : Tasarım değişkeni (oryantasyon açıları) sayısını belirtmektedir.

Seçilen bu kritik parametreler kod içerisinde *Run_Minimizer()* fonksiyonu ve *Composite_Shell_Problem(ElementwiseProblem)* sınıfı içerisinde tanımlanmaktadır. Optimizasyon sürecini daha iyi irdeleyebilmek adına aşağıda bu parametreleri içeren kodun ilgili kısımları gösterilmektedir.

```
class Composite_Shell_Problem(ElementwiseProblem):

    def __init__(self):
        self.ni = 0 # iteration counter
        n_var0 = 24 # Number of composite layers
        xl = np.ones(n_var0)*-90 # Lower Th Limit
        xu = np.ones(n_var0)*90  # Upper Th Limit
        super().__init__(n_var=n_var0, n_obj=1, n_ieq_constr=1, xl=xl, xu=xu)


def Run_Minimizer():
    """ Run FEA for given [x] orientation values."""

    from pymoo.algorithms.moo.nsga2 import NSGA2
    problem = Composite_Shell_Problem()
    algorithm = NSGA2( pop_size = 50, )
    res_min = minimize(problem,
                algorithm,
                ('n_gen', 30),
                seed = 10,
                verbose=False,
                save_history=True)
```

Daha önce belirtildiği üzere kritik parametreler, her bir optimizasyon işleminde belirli değer aralıklarında değiştirilerek işlemler gerçekleştirilmiştir.

Optimizasyonun ilk aşamasında, 24 katmanlı kompozit kabuk için işlemler gerçekleştirilmiştir. Burada *n_gen* parametresiyle ifade edilen nesil/iterasyon sayısı 30 olacak şekilde sabit tutulmuş, *pop_size* ile ifade edilen popülasyon sayısı 10'dan başlayarak 100'e kadar artırılarak toplamda 10 adet optimizasyon işlemi gerçekleştirilmiştir. Bu aşamada, değişken popülasyon sayısının diğer parametrelere göre optimizasyon sonuçlarına olan etkisi incelenmek istenmiştir. *n_gen* parametresi ile gösterilen nesil/iterasyon sayısı deneysel verilere göre belirlenmiş ve 30 şeklinde bir başlangıç değeri uygun görülmüştür. *n_var0* parametresiyle belirtilen tasarım değişkeni sayısı, katman oryantasyon açılarının sayısına eşit olacak şekilde 24 olarak alınmıştır.

Optimizasyon parametreleri ve elde edilen optimizasyon sonuçları Çizelge 4.1'de gösterilmektedir.

Çizelge 4.1. Değişken popülasyon sayısına ve sabit nesil/iterasyon sayısına sahip 24 katmanlı kompozit kabuğun optimizasyon sonuçları

Özellik	**Değer**									
Katman Dizilimi	$[(45/90/-45/0)_3]_s$									
FEM ile elde edilen maksimum sehim	14.18 mm									
Optimizasyon Parametresi	**Değerler**									
pop_size	10	20	30	40	50	60	70	80	90	100
n_gen	30	30	30	30	30	30	30	30	30	30
n_var0	24	24	24	24	24	24	24	24	24	24
Optimizasyon Sonuçları										
Tsai-Hill Hata Kriteri	0.36	0.61	0.4	0.6	0.35	0.38	0.29	0.38	0.28	0.29
Optimizasyon sonrası sehim (mm)	10.57	8.67	8.43	8.19	8.12	7.92	8.22	7.94	7.85	7.81
pop_size Değişimine Göre Elde Edilen Optimum Oryantasyon Açıları (°)										
10	[75.1 54.3 76.4 86.8 18.6 51.7 -4.9 -86.2 46.5 -67.2 -30.4 -22.0 -8.8 72.5 -58.5 -87.1 62.8 57.1 -7.7 -20.9 -81.9 63.7 -85.7 42.2]									
20	[83.9 81.5 79.7 86.6 69.1 -89.6 -43.4 62.2 45.5 0.1 8.1 9.2 -4.5 7.2 -67.4 58.3 68.6 80.5 64.6 61.2 -83.3 39.9 -88.1 80.3]									
30	[63.5 81.3 72.8 86.9 70.2 -85.3 -21.4 75.6 -16.5 55.8 -33.6 -12.0 25.9 3.5 -39.6 28.2 -80.6 62.9 86.4 61.3 65.1 67.7 71.0 79.1]									
40	[83.5 74.9 84.6 82.7 76.4 -88.8 66.9 7.5 -0.8 62.5 -16.0 -9.8 23.7 51.5 -36.3 21.9 73.0 69.7 72.1 73.8 73.8 73.7 -89.7 -89.8]									
50	[78.7 77.5 76.6 82.6 64.3 78.6 80.8 64.2 70.4 -22.4 -79.5 5.0 -3.9 -4.0 20.4 -89.0 62.9 -89.1 62.1 62.6 74.9 75.4 72.8 -90.0]									

(**Devamı Arkada**)

Çizelge 4.1'in devamı

60	[78.5 76.7 83.5 80.1 79.4 82.7 69.2 87.8 -7.4 -16.4 -14.6 25.4 30.8 4.4 -16.3 88.4 77.3 85.4 61.6 74.2 73.2 76.2 76.1 79.7]
70	[76.2 76.7 79.2 83.1 68.8 76.1 21.7 -87.3 -87.5 -15.1 10.9 -17.4 -3.0 50.4 74.8 20.3 -20.1 65.8 69.8 -85.5 79.2 -89.0 69.5 78.8]
80	[75.7 81.8 76.3 83.3 79.3 81.1 54.8 77.8 -2.6 -79.3 -11.2 -3.5 -21.8 -3.2 16.4 21.6 75.3 87.5 61.9 70.9 77.5 81.9 75.1 75.4]
90	[76.8 77.6 75.4 76.5 66.3 71.8 69.6 79.7 -5.8 18.2 8.8 -20.9 -13.3 -6.5 22.8 31.7 72.5 70.6 65.1 76.3 74.1 78.0 73.9 80.0]
100	[76.0 78.2 75.9 78.3 77.8 74.8 80.0 74.7 14.4 -12.9 15.4 19.2 8.3 -21.7 -10.9 23.7 72.1 74.9 78.1 81.1 76.0 80.9 78.8 72.5]

Sonuçlar incelendiğinde, genetik algoritma ile optimizasyon işleminin son derece başarılı sonuçlar ürettiği tespit edilmiştir. Burada, kompozit kabuk için sonlu elemanlar analizinde elde edilen U3 yönündeki 14.18 mm olan maksimum sehim miktarının, 10. optimizasyonda 7.81 mm'ye kadar düştüğü görülmektedir. Bu şekilde yapısal deformasyonda %45 gibi çok ciddi bir düşüş elde edilmiştir. Çizelge 4.1'de aynı zamanda her bir optimizasyon işleminde -90° ile +90° arasından seçilen ve minimize edilen sehim miktarına karşılık gelen elyaf oryantasyon açıları paylaşılmıştır. Burada, elde edilen oryantasyon açıları herhangi bir müdahale edilmeden hesaplandığı gibi ondalıklı olarak bırakılmıştır. Özellikle havacılık sektöründe kompozit parçaların tasarımında, üretim kolaylığı sağlaması açısından genellikle 0°, 30°, 45°, 60° ve 90° şeklinde standartlaşan oryantasyon açıları tercih edilmektedir. Ancak bu çalışmanın amaçlarından biri de genetik algoritma ile gerçekleştirilen optimizasyon işlemlerinde yalın olarak ne tür sonuçlar elde edildiğini göstermektir. Aksi takdirde, Python kodu içerisinde söz konusu oryantasyon açılarının çıktı formatı değiştirilerek arama uzayı değiştirilebilir (örneğin, oryantasyon açılarının sadece 5'in katları olması gibi), istenilen şekilde üretim kolaylığı sağlayan oryantasyon açılarının elde edilmesi sağlanabilir.

Burada optimizasyon işlemlerinin ne kadar sürdüğü hakkında bilgi vermek de faydalı olacaktır. Bu elbette ki optimizasyonu koşturan bilgisayar ya da iş istasyonun performansına bağlı olarak değişse de özellikle popülasyon sayısının giderek artması ciddi anlamda performans ihtiyacı ortaya koymuştur. Söz konusu optimizasyonun 10. adımında sonuçlar yaklaşık 4 saatlik bir zaman dilimi sonucunda elde edilmiştir. Bu adımda GA ile toplamda 2999 adet iterasyon koşturulmuş ve optimizasyon tamamlanmıştır. Gerçekleştirilen her bir iterasyonda, Calculix çözücüsü tarafından sonlu elemanlar analizi yapıldığı ve Tsai-Hill hata kriterinin sağlanıp sağlanmadığının kontrol edildiği hatırlatılmalıdır.

Optimizasyon işleminin ikinci aşamasında yine 24 katmanlı kompozit kabuk için işlemler gerçekleştirilmiştir (Çizelge 4.2). Ancak burada popülasyon sayısı sabit tutulmuş, bu sefer nesil/iterasyon sayısı belirli aralıklarla artırılmıştır. Sabit bir

popülasyon sayısı belirleyebilmek için birinci optimizasyonda elde edilen sehim sonuçlarına göre yakınsama grafiğine bakılmıştır (Şekil 4.6).

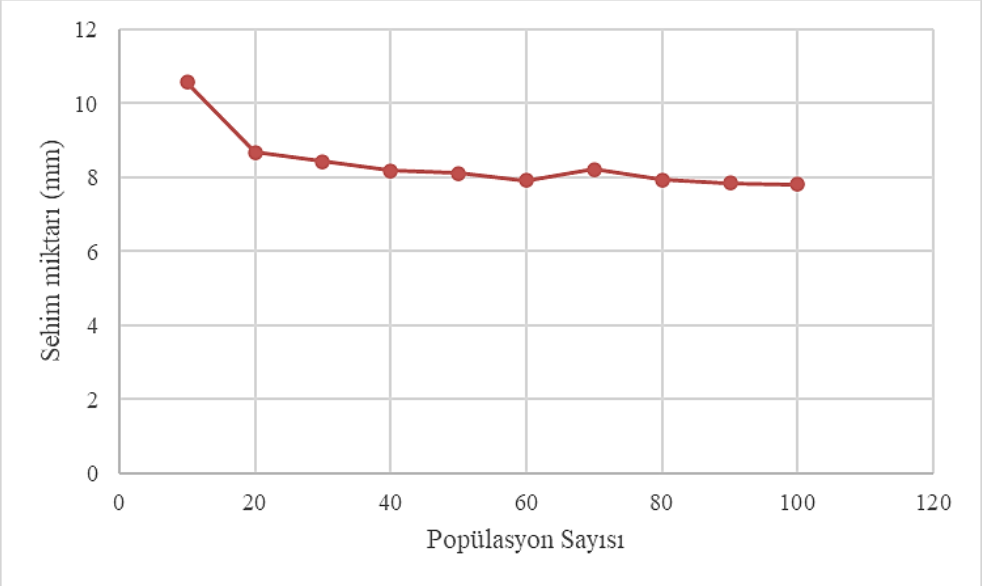
Çizelge 4.2. Değişken nesil/iterasyon sayısına ve sabit popülasyon sayısına sahip 24 katmanlı kompozit kabuğun optimizasyon sonuçları

Özellik	**Değer**									
Katman Dizilimi	$[(45/90/-45/0)_3]_s$									
FEM ile elde edilen maksimum sehim	14.18 mm									
Optimizasyon Parametresi	**Değerler**									
pop_size	50	50	50	50	50	50	50	50	50	50
n_gen	5	10	15	20	25	30	35	40	45	50
n_var0	24	24	24	24	24	24	24	24	24	24
Optimizasyon Sonuçları										
Tsai-Hill Hata Kriteri	0.48	0.33	0.33	0.45	0.34	0.35	0.25	0.33	0.47	0.35
Optimizasyon sonrası sehim (mm)	11.07	9.19	8.6	8.3	8.18	8.12	8.06	7.97	7.95	7.92
n_gen Değişimine Göre Elde Edilen Optimum Oryantasyon Açıları (°)										
5	[61.0 69.5 -81.6 82.3 64.6 -61.8 83.5 67.9 -77.3 -61.5 -55.9 -21.7 43.6 59.0 24.8 -88.4 62.3 -34.9 62.0 -54.9 -70.9 6.9 68.9 -74.2]									
10	[77.7 80.1 83.6 86.9 64.3 79.1 81.7 -78.9 80.6 54.0 -88.5 -27.4 25.2 2.2 87.8 55.6 62.3 -72.5 44.9 52.3 1.5 73.0 68.9 -87.7]									
15	[77.7 81.0 77.1 87.1 64.3 79.1 68.3 64.4 75.2 -23.6 -89.2 -22.4 -2.0 -3.9 35.6 -83.4 60.5 -74.4 62.4 52.0 -84.6 75.2 60.7 -87.9]									
20	[81.5 80.9 76.3 88.0 64.3 85.7 81.7 64.4 71.7 -24.2 -76.0 4.3 -2.3 0.5 35.5 -88.7 62.9 -71.2 62.1 63.6 69.4 78.0 72.8 -87.8]									

(**Devamı Arkada**)

Çizelge 4.2'nin devamı

25	[78.4 76.5 76.6 89.5 64.3 83.0 82.8 64.2 70.4 -24.8 -48.1 13.1 -9.5 -4.0 21.0 -89.5 62.5 -88.7 63.4 61.4 75.7 75.1 71.4 -89.3]
30	[78.7 77.5 76.6 82.6 64.3 78.6 80.8 64.2 70.4 -22.4 -79.5 5.0 -3.9 -4.0 20.4 -89.0 62.9 -89.1 62.1 62.6 74.9 75.4 72.8 -90.0]
35	[75.3 77.2 76.1 80.0 64.6 74.5 83.1 63.9 72.1 -4.0 -33.1 7.0 -5.9 -4.0 13.8 -89.0 61.1 -89.1 63.5 64.1 74.9 75.1 71.5 -89.9]
40	[78.0 77.2 85.4 80.3 76.7 75.8 72.9 74.3 60.7 -4.3 -16.1 7.0 1.5 -4.0 20.7 -89.0 61.1 -88.5 62.9 73.3 74.9 75.1 71.5 -89.4]
45	[82.2 77.2 76.2 80.0 77.0 76.0 79.6 75.5 72.3 -4.0 -15.9 7.0 1.4 -3.7 20.8 -89.0 60.5 -89.1 62.9 73.6 74.9 75.1 71.5 -89.9]
50	[78.7 77.2 76.2 80.0 76.8 78.3 79.6 77.4 71.8 -3.1 -6.6 7.0 1.4 -1.5 18.2 -89.0 61.5 -89.9 62.9 72.8 74.9 77.7 71.7 -89.9]

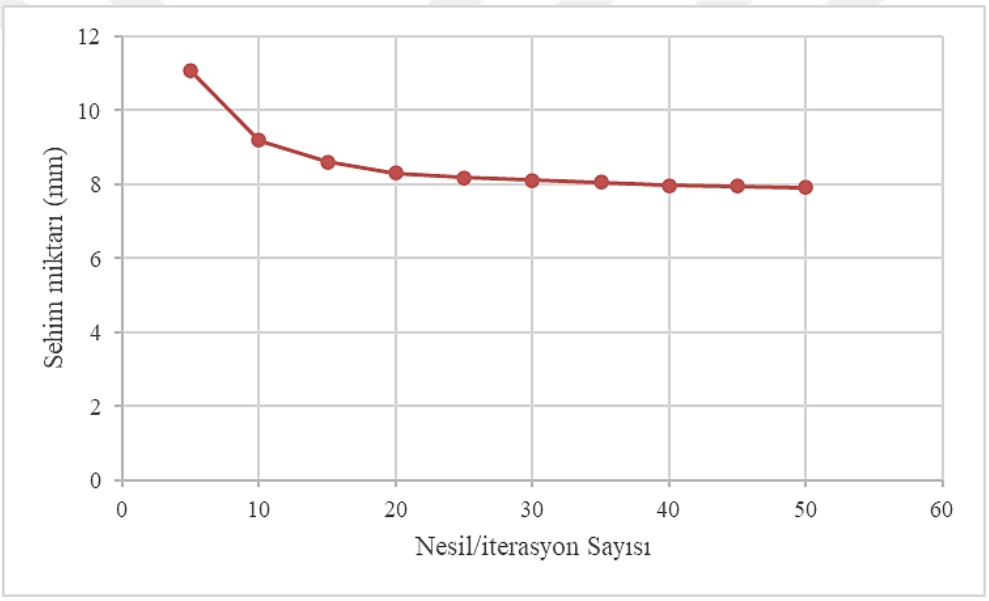


Şekil 4.6. İlk optimizasyonda elde edilen sehim sonuçlarının artan popülasyon sayısına göre yakınsama grafiği

Şekil 4.6'daki grafik incelendiğinde, ilk optimizasyon aşamasında elde edilen verilere göre popülasyon sayısının 50 değerinden sonra sehim değerlerinde çok fazla değişiklik olmaması sebebiyle ikinci optimizasyonda 50 olarak sabitlenmiştir. Burada nesil/iterasyon sayısı 5'ten 50'ye kadar belirli aralıklarla artırılarak diğer parametrelere göre optimizasyon sonuçlarına olan etkisi incelenmek istenmiş ve toplamda 10 adet optimizasyon işlemi gerçekleştirilmiştir. Tasarım değişkeni sayısında değişiklik olmadığı için yine 24 olarak alınmıştır.

Optimizasyon sonuçları incelendiğinde, 10. optimizasyon işleminde sehim miktarının 7.92 mm'ye kadar düştüğü görülmektedir. İlk optimizasyonun aynı adımında elde edilen 7.81 mm'lik sonuç ile kıyaslandığında arada %1.41'lik küçük bir fark olduğu görülmektedir. Performans açısından optimizasyon işlemlerinin süresi incelendiğinde ise, 10. adımda 2499 adet iterasyon koşturulduğu görülmüştür. Bu değer bir önceki optimizasyonun son adımında elde edilen 2999 adetten oldukça düşük görünmektedir. Buradan, popülasyon sayısını düşük tutarak ve nesil/iterasyon sayısını artırarak optimizasyon yapıldığında daha yüksek bir performans ile hemen hemen benzer bir sehim değerine erişildiği sonucu ortaya çıkmaktadır. Çizelge 4.2'de her bir optimizasyon işleminde elde edilen katman oryantasyon açıları görülmektedir.

Optimizasyonun üçüncü aşamasında, 12 katmanlı kompozit kabuk için işlem gerçekleştirilmiştir. Bu aşamada popülasyon sayısı yine 50 olarak sabit tutulmuştur. Referans bir nesil/iterasyon sayısı belirleyebilmek için ikinci optimizasyonda elde edilen sehim sonuçlarına göre yakınsama grafiğine bakılmıştır (Şekil 4.7).



Şekil 4.7. İkinci optimizasyonda elde edilen sehim sonuçlarının artan nesil/iterasyon sayısına göre yakınsama grafiği

Şekil 4.7'deki grafik incelendiğinde, ikinci optimizasyonda elde edilen verilere göre nesil/iterasyon sayısında 30 değerinden sonra sehim miktarlarında çok fazla değişiklik olmadığı tespit edilmiştir. Bu nedenle 12 katmanlı kabuğun optimizasyonunda bu değer referans alınmıştır. Tasarım değişkeni sayısı, katman oryantasyon açılarının sayısına eşit olarak alınmıştır. Bu kısımda sadece 1 kez optimizasyon işlemi gerçekleştirilmiştir. Bunun sebebi optimizasyon için gerekli *pop_size* ve *n_gen* referans değerlerinin artık sabitlenmiş olmasıdır.

12 katmanlı kompozit kabuk için optimizasyon parametreleri ve elde edilen optimizasyon sonuçları Çizelge 4.3'te gösterilmektedir.

Çizelge 4.3. Sabit popülasyon ve nesil/iterasyon sayısına sahip 12 katmanlı kompozit kabuğun optimizasyon sonuçları

Özellik	**Değer**
Katman Dizilimi	$[(45/-45)_3]_s$
FEM ile elde edilen maksimum sehim	19.61 mm
Optimizasyon Parametresi	**Değerler**
pop_size	50
n_gen	30
n_var0	12
Optimizasyon Sonuçları	
Tsai-Hill Hata Kriteri	0.22
Optimizasyon sonrası sehim (mm)	7.8
Elde Edilen Optimum Oryantasyon Açıları (°)	
[74.0 77.7 74.5 79.5 4.7 20.1 2.7 -12.5 64.4 80.2 72.7 75.8]	

Sonuçlar incelendiğinde sehim miktarının 19.61 mm'den 7.8 mm'ye düştüğü görülmektedir. Burada yaklaşık %60 oranında bir azalma elde edilmiştir. 24 katmanlı ilk optimizasyon sonucu ile kıyaslandığında elde edilen kazancın %45'ten %60'a yükseldiği görülmektedir. Ayrıca bu sonuca göre, katman sayısı yarıya düşürüldüğünde ve Çizelge 4.3'te elde edilen oryantasyon açıları kullanıldığında, 24 katmanlı kompozit kabuğun optimizasyonunda elde edilen minimum sehim miktarı ile hemen hemen aynı değerin elde edildiği ortaya çıkmıştır.

Optimizasyon işleminin performansına göz atıldığında ise toplamda 1499 adet iterasyon ile optimizasyon işleminin tamamlandığı görülmüştür. Burada iterasyon sayısındaki ciddi düşüşün en önemli sebebinin, tasarım değişkeni sayısının 24'ten 12'ye düşmesi olduğu anlaşılmıştır. Bu sonuç ile optimizasyon sürecinin performansını etkileyen en önemli parametrenin *n_var0* olarak ifade edilen tasarım değişkenleri sayısı olduğu ortaya çıkmaktadır.

Optimizasyonun son aşamasında, 8 katmanlı kompozit kabuk için işlem gerçekleştirilmiştir. Burada, bir önceki adımda belirlenen referans değerlere istinaden

yine popülasyon sayısı 50 ve nesil/iterasyon sayısı 30 olarak alınmıştır. Tasarım değişkeni sayısı, katman oryantasyon açılarının sayısına eşit olarak alınmıştır. Elde edilen optimizasyon sonuçları Çizelge 4.4'te gösterilmektedir.

Çizelge 4.4. Sabit popülasyon ve nesil/iterasyon sayısına sahip 8 katmanlı kompozit kabuğun optimizasyon sonuçları

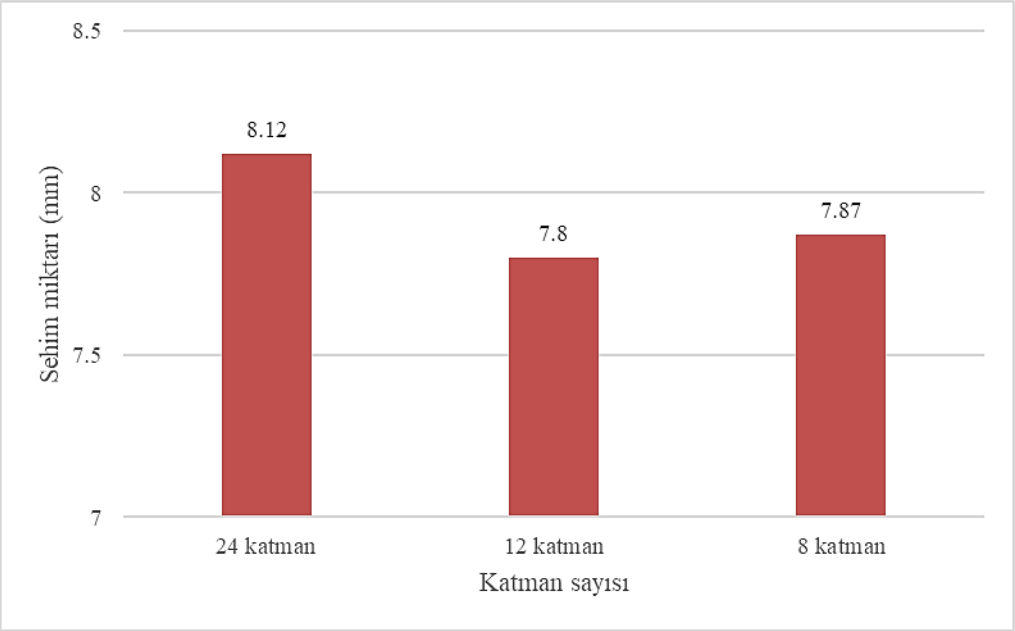
Özellik	**Değer**
Katman Dizilimi	$[(0/90)_2]_s$
FEM ile elde edilen maksimum sehim	18.37 mm
Optimizasyon Parametresi	**Değerler**
pop_size	50
n_gen	30
n_var0	8
Optimizasyon Sonuçları	
Tsai-Hill Hata Kriteri	0.32
Optimizasyon sonrası sehim (mm)	7.87
Elde Edilen Optimum Oryantasyon Açıları (°)	
[77.8 74.1 -89.4 10.5 -2.0 -88.6 73.3 75.0]	

Elde edilen sonuçlar incelendiğinde, sehim miktarının 18.37 mm'den 7.87 mm'ye düştüğü görülmektedir. Burada yaklaşık %57 oranında bir azalma elde edilmiştir. 12 katmanlı kompozit kabuğun optimizasyonunda elde edilen %60'lık oran ile kıyaslandığında arada sadece %3'lük bir fark olduğu ve hemen hemen benzer bir sehim miktarı elde edildiği görülmüştür. 24 katmanlı kompozit kabuğun ilk optimizasyon sonucu ile kıyaslandığında ise optimize edilen sehim miktarındaki kazancın %45'ten %57'ye yükseldiği tespit edilmiştir.

Optimizasyon işleminin performansı incelendiğinde toplamda 1499 adet iterasyon ile optimizasyon işleminin tamamlandığı görülmüştür. Bu sayı 12 katmanlı kompozit kabuğun optimizasyonunda gerçekleştirilen iterasyon sayısı ile bire bir aynıdır. Bunun sebebinin, hem 12 hem de 8 katmanlı kompozit kabuğun optimizasyon

işlemlerinde *pop_size* ve *n_gen* parametrelerinin aynı değerlere sahip olmasından kaynaklandığı anlaşılmaktadır.

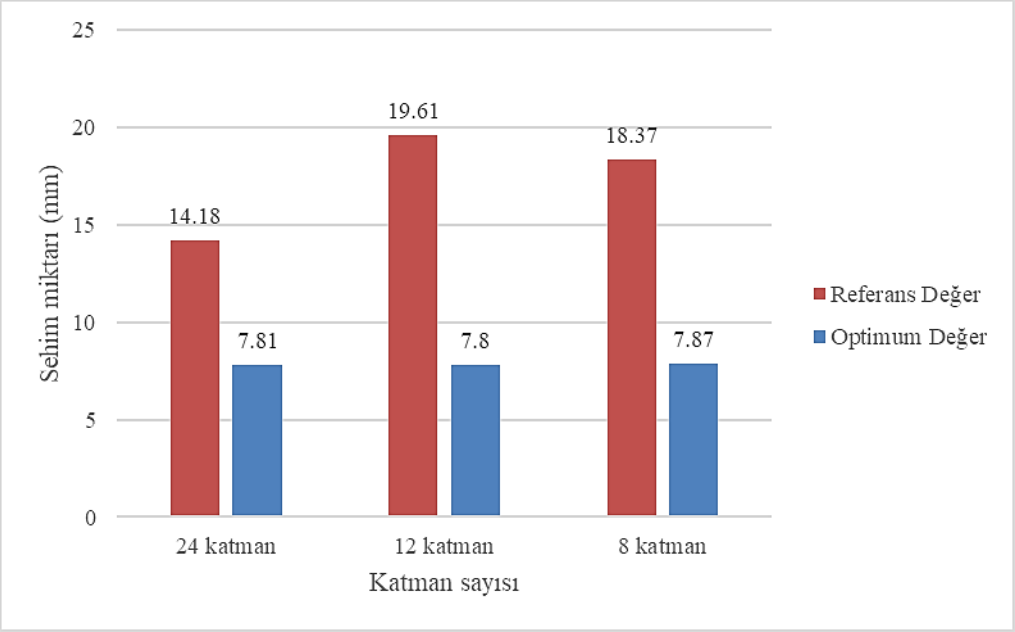
Üç farklı katman dizilimi ile elde edilen optimizasyon sonuçlarının aynı parametreler altında birbirleriyle kıyaslanması, farklı bir bakış açısı kazandıracaktır. Bu sebeple benzer koşullar altında karşılaştırma yapabilmek adına 24, 12 ve 8 katmanlı kompozit kabuklar için *pop_size*=50 ve *n_gen*=30 değerlerine karşılık gelen sehim miktarları Şekil 4.8'de gösterilmektedir.



Şekil 4.8. Üç farklı katman diziliminin *pop_size*=50 ve *n_gen*=30 değerlerindeki optimizasyon sonuçlarının karşılaştırılması

Şekil 4.8'deki grafik incelendiğinde, aynı optimizasyon parametrelerine sahip üç farklı kabuk içerisinde en küçük sehim miktarının 12 katmanlı kompozit kabuk için elde edildiği, ancak genel olarak aralarında çok büyük bir fark olmadığı sonucu ortaya çıkmaktadır.

Kompozit kabukların FEM analizlerinde elde edilen sehim miktarları referans alınarak optimizasyonlarda elde edilen minimum sehim miktarları karşılaştırıldığında, en çok düşüşün 12 katmanlı kabukta gerçekleştiği ve 19.61 mm'den 7.8 mm'ye düşerek %60 oranında bir düşüş elde edildiği görülmüştür. Diğer katmanlara bakıldığında ise 24 katmanlı kabukta sehim miktarının 14.18 mm'den 7.81 mm'ye düştüğü, 8 katmanlı kabukta ise 18.37 mm'den 7.87 mm'ye düştüğü görülmüştür. Bu durum incelendiğinde, optimizasyon sonucunda kompozit kabuklarda sehim miktarlarının azalmasının sağlanmasıyla birlikte yük taşıma kapasitelerinin ve buna bağlı olarak kompozit kabukların yorulma (fatigue) ömrünün artırıldığı sonucuna ulaşılmıştır. Üç farklı katman diziliminin FEM analizlerinde elde edilen sehim miktarları ile optimizasyon sonucunda elde edilen minimum sehim miktarlarının karşılaştırılması Şekil 4.9'da gösterilmektedir.



Şekil 4.9. Üç farklı katman diziliminin FEM analizleri ve optimizasyon sonuçlarının karşılaştırılması

Son olarak yukarıda gerçekleştirilen tüm optimizasyon işlemlerinde genetik algoritmanın, kompozit kabuklarda meydana gelebilecek en küçük sehim miktarına karşılık gelen en uygun katman oryantasyon açılarının hesaplanmasında oldukça başarılı sonuçlar ortaya koyduğu görülmüştür.

5. SONUÇLAR

Kompozit malzemelerin hafiflik, yüksek mukavemet, korozyon direnci, uzun ömürlülük gibi üstün özelliklerinin yanı sıra özellikle yakıt verimliliği ve performans artışı sağlaması gibi faktörler nedeniyle havacılık sektöründe kullanımı kritik öneme sahiptir. Her geçen yıl uçaklarda kullanımı giderek yaygınlaşan bu malzemelerin performanslarını artırmaya yönelik önemli çalışmalardan biri de gelişen teknolojik koşullarla birlikte optimizasyon yöntemlerinin yaygın olarak kullanılmaya başlanmasıdır. Optimizasyon teknikleriyle kompozit malzemeler için en uygun malzeme seçimi, optimum katman dizilimi, yük altındaki davranış ve mekanik özelliklerinde iyileştirme, üretim sürecini hızlandırabilecek en uygun imalat yönteminin belirlenmesi gibi karar alma süreçlerinde oldukça başarılı sonuçlar elde edilerek maliyet ve verimlilik üst seviyeye çıkarılmaktadır.

Bu tez çalışmasında, kompozitlerin optimizasyonunda en çok tercih edilen yöntemlerden biri olan Genetik Algoritma (GA) ile optimizasyon işlemleri gerçekleştirilmiştir. Burada, sürekli elyaf takviyeli katmanlı kompozit bir yapıda meydana gelebilecek minimum sehim miktarına karşılık en uygun elyaf oryantasyon açılarının hesaplanması ve böylece yapısal deformasyonun en aza indirgenmesi amaçlanmıştır.

Bu kapsamda ilk olarak bu çalışmaya özel Python programlama dilinde kodlar geliştirilmiş, bu kodlar aracılığıyla genetik algoritmanın Calculix sonlu elemanlar çözücüsü ile entegrasyonu sağlanmıştır. Bu kısımda optimizasyon problemi oluşturularak amaç ve kısıt fonksiyonları tanımlanmıştır. Amaç fonksiyonu, kompozit yapıdaki maksimum sehim miktarı ve kısıt fonksiyonu da Tsai-Hill hata kriteri olacak şekilde belirlenmiştir. Çalışmanın ikinci kısmında, kompozit yapı örneği olarak savaş uçaklarının kanadında sıkça kullanılan Leading Edge Flap (LEF) yapısal komponentinin üst kabuğunu temsilen $[(45/90/-45/0)_3]_s$, $[(45/-45)_3]_s$ ve $[(0/90)_2]_s$ dizilimlerine sahip sırasıyla 24, 12 ve 8 katmanlı üç farklı kompozit kabuk tasarımı gerçekleştirilmiş ve tüm kabuklar için sonlu elemanlar modelleri oluşturulmuştur. Kompozit kabuklarda havacılıkta sıkça tercih edilen M91-IM10 UD karbon prepreg malzemesi kullanılmıştır. Ardından her üç kabuk için optimizasyon sonuçlarıyla kıyaslayabilmek adına sonlu elemanlar analizleri yapılarak maksimum sehim miktarları elde edilmiştir. Elde edilen sonuçlar incelendiğinde en yüksek sehim değerlerinin kabuklarda -z (U3) yönünde elde edildiği görülmüş ve bu sebeple optimizasyonla bu değerlerin minimize edilmesine karar verilmiştir. Çalışmanın son aşamasında genetik algoritma ile optimizasyon işlemleri gerçekleştirilmiştir. Bu kısımda 24, 12 ve 8 katmanlı her üç kompozit kabuk için optimizasyonlar koşturularak daha önce bu kabuklar için FEM yöntemiyle elde edilen maksimum sehim miktarları minimize edilmiştir.

Elde edilen sonuçlar incelendiğinde ilk olarak genetik algoritma ile optimizasyon işlemlerinin son derece başarılı sonuçlar ortaya koyduğu görülmektedir. 24 katmanlı kompozit kabukta FEM ile elde edilen maksimum sehim miktarının optimizasyonla 14.18 mm'den 7.81 mm'ye düştüğü ve yaklaşık %45 oranında bir azalma elde edildiği görülmüştür. Yine 12 katmanlı kabukta sehim miktarının 19.61 mm'den 7.8 mm'ye ve 8 katmanlı kabukta 18.37 mm'den 7.87 mm'ye düşerek sırasıyla yaklaşık %60 ve %57 oranlarında bir azalma elde edildiği görülmüştür. Ayrıca burada gerçekleştirilen optimizasyonlarda sehim miktarlarında azalmanın sağlanmasıyla

birlikte kompozit kabukların yük taşıma kapasitelerinin de artırıldığı tespit edilmiştir. Optimizasyonla daha az sayıda katman kullanarak benzer sehim sonuçlarının elde edilmesinin görülmesi üzerine aynı performansı gösteren daha az sayılı katman kullanılması ile kompozit kabukların hem imalat kolaylığı hem de imalat ekonomisi sağlanmıştır. Sonuçlar, genetik algoritmanın kompozit kabuklardaki sehim miktarlarının düşürülmesinde ne kadar etkili bir yöntem olabileceğini kanıtlamaktadır. Ancak burada söz konusu kompozit kabukların basit yükleme koşulları altında tasarlandığının da altı çizilmelidir. Gerçek hayatta yapıya uygulanan yüklerin daha kompleks olduğu, yüklerin aktarılacağı mesnet noktalarının farklı şekillerde tasarlanabileceği bir gerçektir. Diğer yandan sonuçlara göz atıldığında, tahminlerin GA ile optimizasyon işlemlerinin yukarıda bahsedilen konularda mutlaka katkı sağlayacağı yönünde olduğu ve bu konuda daha ayrıntılı inceleme yapılması gerektiği söylenebilir.

Optimizasyonla elde edilen minimum sehim miktarlarına karşılık gelen elyaf oryantasyon açıları incelendiğinde, elde edilen açıların hassas, ondalıklı hatta eksi açılarda olduğu görülmektedir. Bu açıların 0°, 30°, 45°, 60° ve 90° gibi üretim kolaylığı sağlayan ve standartlaşmış oryantasyon açılarından oldukça farklı olduğu ve bu sebeple uygulanabilir olmadığı düşünülebilir. Ancak bu tez çalışmasının hedeflerinden biri de genetik algoritmanın çözüm uzayında herhangi bir müdahale olmadan yalın halde çalıştırıldığında ne tür sonuçlar ortaya koyabileceğini göstermektir. Aksi takdirde oryantasyon açılarının çıktı formatı Python kodu içerisinde istenildiği gibi şekillendirilebilir; örneğin sadece 5'in katları şeklinde olması sağlanabilir ya da aralık değerleri değiştirilebilir. Bu çalışmada elyaf oryantasyon açılarının sadece -90° ve +90° arasında olması gerektiği belirtilmiştir. Diğer taraftan, üretim yöntemi olarak farklı yönlerde ve hassas açılarda elyaf yerleşimleri yapabilen otomatik fiber yerleştirme makineleri (AFP) kullanılarak kompozit malzemelerin tasarımları gerçekleştirilebilmektedir. Bu makineler vasıtasıyla havacılık ve uzay alanında uçak kanatları, gövde panelleri, roket motor kaplamaları ve füze parçaları gibi birçok kompozit parçanın tasarımları yüksek hızlarda üretilebilmektedir. Dolayısıyla optimizasyonla elde edilen hassas oryantasyon açıları, AFP makineleriyle gerçekleştirilen kompozit üretim süreçlerinde rahatlıkla uygulanabilir. Diğer yandan buradaki en önemli sonuçlardan birinin ağırlık kazancı olarak ortaya çıkabileceği tahmin edilmektedir. Bu tez çalışmasındaki optimizasyonlarda kabuk kalınlıkları sabit tutularak en küçük sehim miktarları elde edilmiştir. Bu durumda, oryantasyon açılarıyla birlikte katman kalınlığı da aynı anda optimize edilerek ağırlık optimizasyonu yapılabileceği ve bu şekilde daha az katman sayısı elde edilebileceği tahmini yapılmaktadır. Sonuç olarak bu çalışmada, yapay zekâ temelli optimizasyon yöntemlerinin kompozit malzemelerin optimizasyonunda özellikle performans ve maliyet açısından önemli katkılar sağlayabileceği başarılı bir şekilde ortaya konmuştur.

6. KAYNAKLAR

Agarwal, B.D., Broutman, L.J. and Chandrashekhara, K. 2006. Analysis and Performance of Fiber Composites, Wiley, New Jersey.

Ashby, M.F. and Jones, D.R.H. 2012. Chapter 1-Engineering Materials and Their Properties. In: Engineering Materials 1: An Introduction to Properties, Applications and Design, Butterworth-Heinemann, Burlington, MA.

Baker, A.A., Dutton, S. and Kelly, D. 2004. Composite Materials for Aircraft Structures. American Institute of Aeronautics and Astronautics, Inc., Reston, Virginia.

Barnes, R.H. and Morozov, E.V. 2016. Structural optimisation of composite wind turbine blade structures with variations of internal geometry configuration. Composite Structures, vol. 152, pp. 158-167.

Campbell, F.C. 2010. Structural Composite Materials. ASM International, Materials Park, Ohio.

Daniel, I. M. and Ishai, O. 2006. Engineering Mechanics of Composite Materials (2nd ed.). Oxford University Press, New York.

Dantzig, G.B. 1963. Linear Programming and Extensions. Princeton University Press, Princeton.

De Jong, K.A. 1975. Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Dissertation, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Dillinger, J.K.S., Klimmek, T., Abdalla, MM. and Gurdal, Z. 2013. Stiffness Optimization of Composite Wings with Aeroelastic Constraints. Journal of Aircraft: devoted to aeronautical science and technology, 50(4), pp. 1159-1168.

Dorigo, M. and Stützle, T. 2004. Ant Colony Optimization. The MIT Press, Cambridge.

Dreo, J. and Candan, C. 2011. Different classifications of metaheuristics. https://commons.wikimedia.org/wiki/File:Metaheuristics_classification.svg [Son erişim tarihi: 11.05.2025].

Emel, G.G. ve Taşkın, Ç. 2002. Genetik algoritmalar ve uygulama alanları. Bursa Uludağ Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 21, 1, ss. 129-152.

Giriprasad, S., Rao, D.S. and Naidu, D.M. 2017. Design and Optimization of Laminated Hybrid Composite Cylinder with Stiffeners Subjected to Buckling Load.

Glover, F. 1989. Tabu search—Part I. ORSA Journal on Computing, vol. 1, issue 3, 190-206.

Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA.

Guo, S., De Los Monteros, J. E. and Liu, Y. 2015. Gust Alleviation of a Large Aircraft with a Passive Twist Wingtip. Aerospace, 2(2), 135-154.

Gutierrez-Delgadillo, D., Fragoudakis, R., Zimmerman, M. and Saigal, A. 2015.

Cantilever boxbeam application of composite stacking sequence optimization using adaptive genetic algorithm. In: TMS 2015 144th annual meeting and exhibition. Springer, Cham, pp. 705–712.

Fagan, E.M., Leen, S.B., de la Torre, O. and Goggins, J. 2017. Experimental investigation, numerical modelling and multi-objective optimisation of composite wind turbine blades. Journal of Structural Integrity and Maintenance, 2(2), pp. 109–119.

Gay, D., Hoa, S.V. and Tsai, S.W. 2003. Composite Materials Design and Application. CRC Press LLC, Boca Raton.

Hearle, J. W. S. 2001. High Performance Fibers. CRC Press, Woodhead Publishing Limited, Cambridge, England.

Herath, M.T., Prusty, B.G., Phillips, A.W. and John, N.S. 2017. Structural strength and laminate optimization of self-twisting composite hydrofoils using a Genetic Algorithm. Composite Structures, vol. 176, pp. 359-378.

Hexcel Corporation. 2020. HexPly M91 product data sheet. https://hexcel.com/user_area/content_media/raw/HexPly_M91_global_DataSheet.pdf [Son erişim tarihi: 13.05.2025].

Hillier, F.S. and Lieberman, G.J. 2001. Introduction to Operations Research. McGraw-Hill, New York.

Holland, J.H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI.

Hull, D. and Clyne, T. 1996. An Introduction to Composite Materials. 2nd Edition, Cambridge University Press, Cambridge.

Keklik, G. ve Özcan, B.D. 2023. Genetik algoritmaların işleyişi ve genetik algoritma uygulamalarında kullanılan operatörler. Osmaniye Korkut Ata Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 6, 1, ss. 1052-1066.

Kennedy, J. and Eberhart, R. 1995. Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, pp. 1942-1948, Perth, WA, Australia.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. 1983. Optimization by simulated annealing. Science, vol. 220, issue 4598, pp. 671-680.

Lee, Y.J., Lin, C.C., Ji, J.C. and Chen, J.S. 2005. Optimization of a composite rotor blade using a genetic algorithm with local search. Journal of Reinforced Plastics and Composites, 24(16), pp. 1759–1769.

Luenberger, D.G. and Ye, Y. 2008. Linear and Nonlinear Programming. Springer.

Maes, V.K., Pavlov, L. and Simonian, S.M. 2019. An efficient semi-automated optimisation approach for (grid-stiffened) composite structures: Application to Ariane 6 Interstage. Composite Structures, vol. 209, pp. 1042-1049.

Mallick, P.K. 2007. Fiber-Reinforced Composites: Materials, Manufacturing, and Design. CRC Press, Boca Raton.

Mitchell, M. 1996. An Introduction to Genetic Algorithms. The MIT Press.

Nemhauser, G.L. and Wolsey, L.A. 1988. Integer and Combinatorial Optimization. John Wiley and Sons.

Rashed, R. (1994). The Development of Arabic Mathematics: Between Arithmetic and Algebra. Kluwer Academic Publishers, Dordrecht.

Rechenberg, I. 1973. Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart.

Reddy, J.N. 2004. Mechanics of Laminated Plates and Shells: Theory and Analysis. 2nd Edition, CRC Press, Boca Raton.

Strong, B.A. 2008. Fundamentals of Composites Manufacturing: Materials, Methods and Applications. Society of Manufacturing Engineers, Dearborn, Michigan.

Struik, D.J. 1948. A Concise History of Mathematics. Dover Publications, New York.

Talbi, E.G. 2009. Metaheuristics: From Design to Implementation. John Wiley and Sons.

Wang, L., Kolios, A., Nishino, T., Delafin, PL. and Bird, T. 2016. Structural optimisation of vertical-axis wind turbine composite blades based on finite element analysis and genetic algorithm. Composite Structures, vol. 153, pp. 123-138.

Zeyrek, B. Y., Aydoğan, B., Dilekcan, E. and Ozturk, F. 2022. Review of Thermoplastic Composites in Aerospace Industry. International Journal on Engineering Technologies and Informatics, 3(1), pp. 1–6.

Anonymous 1: NEOS Optimization Guide. https://neos-guide.org/guide/types/ [Son erişim tarihi: 11.05.2025].

7. EKLER

EK-1: Optimizasyon için geliştirilen Python kodu

```
#%% Imports

import subprocess
import numpy as np
from matplotlib import pyplot as plt
import pyvista as pv

from pymoo.optimize import minimize
from pymoo.core.problem import Problem
from pymoo.core.problem import ElementwiseProblem
from pymoo.termination import get_termination

from typing import TextIO

# Path and File Name
path = # The file path where the Python code is run
inp_name = # The name of the input file (.inp) for Calculix
full_name = f"{path}\\{inp_name}"

#%% Functions

def Stress_VonMiss(S):
    """Calculate Von-Misses Average Stress"""
    if S.shape == (6,):
        s_vm = ( (S[0]-S[1])**2 + (S[1]-S[2])**2 + (S[0]-S[2])**2 + 3*( S[3]**2 +
S[4]**2 + S[5]**2 ) )**0.5
    else:
        s_vm = ( (S[:, 0]-S[:, 1])**2 + (S[:, 1]-S[:, 2])**2 + (S[:, 0]-S[:, 2])**2 + 3*( S[:,
3]**2 + S[:, 4]**2 + S[:, 5]**2 ) )**0.5
    return s_vm

def Tsai_Hill_Cr(stresses):
    """ Calculate the Tsai-Hill failure criterion for a composite layer. """

    # Material strengths (example values for a carbon/epoxy composite)
    strengths = {
        'X_t': 3520.0,  # Tensile strength in 1-direction (MPa)
        'X_c': 1880.0,  # Compressive strength in 1-direction (MPa)
        'Y_t': 42.7,    # Tensile strength in 2-direction (MPa)
        'Y_c': 127.0,   # Compressive strength in 2-direction (MPa)
        'S_12': 105.0,   # Shear strength in 1-2 plane (MPa)
    }

    sigma_11, sigma_22, sigma_33, tau_12, tau_13, tau_23 = stresses
```

```
    # Determine whether to use tensile or compressive strengths based on stress sign
    X = strengths['X_t'] if sigma_11 >= 0 else strengths['X_c']
    Y = strengths['Y_t'] if sigma_22 >= 0 else strengths['Y_c']
    S_12 = strengths['S_12']

    # Calculate Tsai-Hill index
    term1 = (sigma_11 / X) ** 2
    term2 = (sigma_22 / Y) ** 2
    term3 = (sigma_11 * sigma_22) / (X ** 2)  # Interaction term 1-2
    term4 = (tau_12 / S_12) ** 2

    tsai_hill_index = (term1 + term2 - term3 + term4 )

    return tsai_hill_index

#%% Plot Functions

def Plot_Part(Results):
    """Scatter Plot of Part Nodes"""
    coords = Results[0]
    fig = plt.figure()
    ax = fig.add_subplot(111, projection= "3d")
    xyz = []
    for i, node in enumerate(coords[-1]):
        X, Y, Z = coords[-1][node]
        xyz.append((X, Y, Z))
    xyz = np.array(xyz)
    ax.scatter( xyz[:,0], xyz[:,1], xyz[:,2], color="k")
    plt.show( block=True )
    return xyz

def Plot_Elements(Results):
    """Plot Elements Having Max Stress"""
    pv.global_theme.cmap = 'jet'

    R_coord = Results[0][-1]
    R_elems = Results[1][-1]
    R_disps = Results[2][-1]
    R_stres = Results[3][-1]

    # Get Stress
    stress_vals, stress_maxs, stress_ids, stress_keys = _get_values_and_find_max(
R_stres )

    # Nodal Connection List for S8R Element
    Faces = [
        [ 8, 1, 13], [13, 5, 16], [16, 4, 12], [12, 0,  8], [8,  13, 16], [16, 12,  8],
```

```
    [ 9, 2, 14], [14, 6, 17], [17, 5, 13], [13, 1,  9], [9,  14, 17], [17, 13,  9],
    [10, 2, 14], [14, 6, 18], [18, 7, 15], [15, 3, 10], [10, 14, 18], [18, 15, 10],
    [11, 3, 15], [15, 7, 19], [19, 4, 12], [12, 0, 11], [11, 15, 19], [19, 12, 11],
    [18, 7, 19], [19, 4, 16], [16, 5, 17], [17, 6, 18], [18, 19, 16], [16, 17, 18],
    [10, 3, 11], [11, 0,  8], [ 8, 1,  9], [ 9, 2, 10], [10, 11,  8], [ 8,  9, 10],
      ]

Cell = [ ]
for row in Faces:
   len(row)
   Cell.append( [len(row),] + row )

# Find Elements Having Max Sress
elem_keys = [ ]
for el_key in R_elems:
   for s_key in stress_keys:
      if any(R_elems[el_key] == s_key):
         elem_keys.append(el_key)
elem_keys = list(set(elem_keys))

# Include Neighbour Elements
ngbr_el_keys = []
ngbr_range = 10
for el_key in elem_keys:
   el_n = int(el_key[0])
   for i in range(-ngbr_range, 25*ngbr_range+1):
      ngbr_el_keys.append( ( str(el_n+i), ) )
ngbr_el_keys = list(set(ngbr_el_keys))

# Plot 3D PyVista
plotter = pv.Plotter()
def_scale = 1
s_bar = {"title":"S_vm\n[MPa]", "vertical":True, "title_font_size":25,}

sm_Nodes = []
sm_Disps = []
sm_Stres = []
for el_key in ngbr_el_keys:
   try:
      el_nodes = R_elems[ el_key ]
   except:
      pass
   Points = [ ]
   Stress = [ ]
   Deform = [ ]
   for node in el_nodes.astype(int):
      x = float(R_coord[node][0])
      y = float(R_coord[node][1])
```

```
            z = float(R_coord[node][2])
            Points.append([ x, y, z ])

            dx = float(R_disps[node][0])
            dy = float(R_disps[node][1])
            dz = float(R_disps[node][2])
            Deform.append([ dx, dy, dz ])

            s11 = float(R_stres[node][0])
            s22 = float(R_stres[node][1])
            s33 = float(R_stres[node][2])
            s12 = float(R_stres[node][3])
            s23 = float(R_stres[node][4])
            s13 = float(R_stres[node][5])
            Stress.append([ s11, s22, s33, s12, s23, s13 ])

        Points = np.array(Points)
        Deform = np.array(Deform)
        Stress = np.array(Stress)

        # mesh_undeformed = pv.PolyData( Points, Cell )
        # plotter.add_mesh( mesh_undeformed, opacity=0.2, show_edges=True,
line_width=1 )

        color = Stress_VonMiss(Stress)
        mesh_deformed = pv.PolyData( Points + Deform * def_scale, Cell )
        plotter.add_mesh( mesh_deformed, opacity=1.0, scalars=color,
scalar_bar_args=s_bar  )

        sm_Nodes.append(Points)
        sm_Disps.append(Deform)
        sm_Stres.append(Stress)

    plotter.show_axes()
    plotter.show()

    return sm_Nodes, sm_Disps, sm_Stres

def Plot_History(res_min):
    """ Plot History of F, G, X """
    hist_ni = np.array([e.evaluator.n_eval for e in res_min.history])
    hist_FG = np.array([ [ e.opt[0].F, e.opt[0].G ] for e in res_min.history])
    hist_X = np.array([ e.opt[0].X for e in res_min.history])

    plt.figure()
    plt.title("F(X)")
    plt.plot(hist_ni, hist_FG[:,0], "*-")
    plt.xlabel("n_iter")
```

```
    plt.ylabel("max Z_Disp. [mm]")
    plt.show()

    plt.figure()
    plt.title("G(X)")
    plt.plot(hist_ni, hist_FG[:,1]+1, "*-" )
    plt.xlabel("n_iter")
    plt.ylabel("Tsai-Hill Index")
    plt.show()

    plt.figure()
    plt.title("[X]")
    L_mrk = ["*", "o", "s", "d", "x", 6, 7, 8, 9, ]
    n_mrk = len(L_mrk)
    for i in range(np.size(hist_X, 1)):
        plt.plot(hist_ni, hist_X[:,i], "--", marker=L_mrk[i % n_mrk], label=f"Lyr-{i}")
    plt.xlabel("n_iter")
    plt.ylabel("Th [deg]")
    plt.legend()
    plt.show()

    return hist_ni, hist_FG, hist_X

def _get_all_nodes(R_coords):
    """ Get All Nodes """
    nodes = []
    node_coords = []
    for node in R_coords:
        X, Y, Z = R_coords[node]
        nodes.append(node)
        node_coords.append((X, Y, Z))

    nodes = np.array(nodes)
    node_coords = np.array(node_coords)

    return nodes, node_coords

def _get_values_for_nodes(node_set, res_dict):
    """Get values for given node-set"""

    # Get Values and Keys From Dict.
    values = []
    keys = []
    for node in node_set:
        value = res_dict[node]
        values.append(value)
        keys.append(node)
    values = np.array(values)
```

```
    keys = np.array(keys)

    # Get Max Components
    c_ids = []
    c_maxs = []
    for col in range(np.size(values, 1)):
        max_c_id = np.argmax(np.abs(values[:, col]))
        max_c = values[max_c_id, col]
        c_maxs.append(max_c)
        c_ids.append(max_c_id)
    c_maxs = np.array(c_maxs)
    c_keys = keys[c_ids]

    return values, c_maxs, c_ids, c_keys

def _get_values_and_find_max(res_dict):
    """Get values for given node-set"""

    # Get Values and Keys From Dict.
    values = []
    keys = []
    for key in res_dict:
        value = res_dict[key]
        values.append(value)
        keys.append(key)
    values = np.array(values)
    keys = np.array(keys)

    # Get Max Components
    c_ids = []
    c_maxs = []
    for col in range(np.size(values, 1)):
        max_c_id = np.argmax(np.abs(values[:, col]))
        max_c = values[max_c_id, col]
        c_maxs.append(max_c)
        c_ids.append(max_c_id)
    c_maxs = np.array(c_maxs)
    c_keys = keys[c_ids]

    return values, c_maxs, c_ids, c_keys

#%% FEA Functions FRD

def _read_coords_block(f:TextIO, line_split:list[str]):
    """Parse Coordinates of nodes from FRD result file."""

    step_time = float(1.0)
    component_names:list[str] = ["X", "Y", "Z"]
```

```
    values:dict[int, list[float]] = {}
    entity_name = 'COORD'

    while line := f.readline():
        line_split = line.split()
        line_type = line_split[0]

        if line_type == '-1':
            nid, components = _read_component_line(line, 12)
            values[nid] = components
        elif line_type == '-3':
            break

    # Get number of components from arbitrary dict item
    if values == {}:
        no_comp = 0
        return "empty"
    else:
        no_comp = len(next(iter(values.values())))
        # print("   >", entity_name, no_comp, "OK")
        # convert dict[int, list[float]] -> dict[int, np.NDarray]
        values_arr = {nid:np.array(v, dtype=float) for nid, v in values.items()}

        return (entity_name, no_comp, step_time, tuple(component_names[:no_comp]),
values_arr)

def _read_elements_block(f:TextIO, line_split:list[str]):
    """Parse Elements of nodes from FRD result file."""

    step_time = float(1.0)
    component_names:list[str] = ["node "*2]
    values:dict[int, list[float]] = {}
    entity_name = 'ELEMENT'

    while line := f.readline():
        line_split = line.split()
        line_type = line_split[0]

        if line_type == '-1':
            nid =   line_split[1],
        elif line_type == '-2':
            values[nid] = line_split[1:]
            line2 = f.readline()
            line2_split = line2.split()
            values[nid] += line2_split[1:]
        elif line_type == '-3':
            break
```

```
    # Get number of components from arbitrary dict item
    if values == {}:
        no_comp = 0
        return "empty"
    else:
        no_comp = len(next(iter(values.values())))
        # print("   >", entity_name, no_comp, "OK")
        # convert dict[int, list[float]] -> dict[int, np.NDarray]
        values_arr = {nid:np.array(v, dtype=float) for nid, v in values.items()}

        return (entity_name, no_comp, step_time, tuple(component_names[:no_comp]),
values_arr)

def _read_result_block(f:TextIO, line_split:list[str]):
    """Parse result values of nodes from FRD result file."""

    step_time = float(line_split[2])
    component_names:list[str] = []
    values:dict[int, list[float]] = {}
    entity_name = ''

    while line := f.readline():
        line_split = line.split()
        line_type = line_split[0]

        # Following if statements in optimized order
        # Most common case first
        if line_type == '-1':
            nid, components = _read_component_line(line, 12)
            values[nid] = components

        elif line_type == '-5':
            component_names.append(line_split[1])

        elif line_type == '-4':
            entity_name = line_split[1]

        elif line_type == '-3':
            break

        elif line_type == '-2': # Rarest case
            _, components = _read_component_line(line, 12)
            values[nid] += components # type: ignore
            #nid comes from the prvious '-1' line

    # Get number of components from arbitrary dict item
    if values == {}:
        no_comp = 0
```

```
        return "empty"
    else:
        no_comp = len(next(iter(values.values())))
        # print("   >", entity_name, no_comp, "OK")
        # convert dict[int, list[float]] -> dict[int, np.NDarray]
        values_arr = {nid:np.array(v, dtype=float) for nid, v in values.items()}

        return (entity_name, no_comp, step_time, tuple(component_names[:no_comp]),
values_arr)

def _read_component_line(line:str, len_comp_str:int) -> tuple[int, list[float]]:
    """Parse lines from FRD result file."""
    try: nid = int(line[3:13])
    except ValueError: nid = 0 # Happens if line is a continuation line, rare

    tmp, n =  line[13:].rstrip(), len_comp_str
    comps = [tmp[i:i+n] for i in range(0, len(tmp), n)]
    comps = [float(c) for c in comps]

    return nid, comps

def Read_FRD_File(full_name):
    """This function reads and parses result file which is generated by Calculix"""

    # print("\n*** Reading FEA Results. ***")
    # print("   > FRD :", full_name+".frd")
    Results = []

    with open(full_name+".frd") as f:
        while line := f.readline():
            line_split = line.split()

            if line_split[0] == '2C':
                #print("Reading Coordinates")
                rs = _read_coords_block(f, line_split)
                if not rs == "empty":
                    Results.append(rs)

            if line_split[0] == '3C':
                #print("Reading Elements")
                rs = _read_elements_block(f, line_split)
                if not rs == "empty":
                    Results.append(rs)

            if line_split[0] == '100CL':
                #print("Reading Results")
                rs = _read_result_block(f, line_split)
                if not rs == "empty":
```

```
                Results.append(rs)

    return Results

#%% FEA Functions (INP and RUN)

def Generate_Inp_File(x):
    """This function prepares input file for Calculix.
    X: contains layer orientations in degrees.
    inp_file_01: Contains node and mesh informations. Must be prepared manually.
    inp_file_02: Contains orientation informations. It is modified here.
    inp_file_03: Contains B.C. and other informations. Must be prepared manually.
    Returns: input file full path

    Creates orientaion card.
    Creates shell-sections card.
    Merges all files and writes input file for Calculix.

    !! File names, paths and layer thicknesses are hard coded here. !!
    !! Modify as needed. Attention on whitespaces.                  !!"""

    # Hard Coded Parameters
    # Layer Thickness
    h_layer = 4.416/len(x)

    inp_file_01 = open(full_name+"_01.txt", "r").readlines()
    inp_file_02 = open(full_name+"_02.txt", "r").readlines()
    inp_file_03 = open(full_name+"_03.txt", "r").readlines()

    # print("\n-----------------------------------------\n")

    # th = x in radians
    orient_str = "**\n"
    for i, th_i in enumerate(x):

        ab = ( 3200.22, -3882.17, 0.0,  1857.83, -1900.0, 77.24 )

        str_i = f"*ORIENTATION, NAME=OR_{i}, SYSTEM=RECTANGULAR \n
{ab[0]:0.6f}, {ab[1]:0.6f}, {ab[2]:0.6f}, {ab[3]:0.6f}, {ab[4]:0.6f}, {ab[5]:0.6f}\n"
        str_i += f" 3, {th_i}\n** \n"
        orient_str += str_i

    # print(orient_str)

    str_i = "*SHELL SECTION, , COMPOSITE, Elset=Element_Set-1 \n"
    orient_str += str_i

    for i, th_i in enumerate(x):
```

```
        str_i = "{1:0.6f}, , composite, OR_{0} \n".format(i, h_layer)
        orient_str += str_i

    inp_file_02 = [orient_str,]

    inp_file = inp_file_01 + inp_file_02 + inp_file_03

    with open(full_name+".inp", "w") as inp_file_w:
        for line in inp_file:
            inp_file_w.write(line)

    return full_name

def Run_Calculix(full_name, out=False):
    """This function calls Calculix over subprocess.
    Waits for completion of FEA solver."""

    sp_calculix = subprocess.run(["ccx_dynamic.exe", full_name], shell=False,
capture_output=True, text=True)
    if out == True:
        print(sp_calculix.stdout)

    if not sp_calculix.stderr == "":
        print(sp_calculix.stderr)
        success = False
    else:
        # print("*** FEA Calculation Finished. ***")
        success = True

    return success, sp_calculix.stdout, sp_calculix.stderr

def Run_FEA_Main(x):
    """ Run FEA for given [x] orientation values."""

    # Generate_Inp_File
    full_name = Generate_Inp_File(x)
    # Run FEA
    Run_Calculix(full_name, out=False)
    # Get Resluts
    Results = Read_FRD_File(full_name)

    return Results

#%% Optimization Functions

def Objective_Function(Results):
    """This function defines objective function for optimization.
    This function must return a float number which corresponds
```

```
    to the optimization score is going to be minimized."""

    # # Get Results
    # R_coord = Results[0][-1]
    # R_elems = Results[1][-1]
    R_disps = Results[2][-1]
    R_stres = Results[3][-1]

    # Get Displacements and Stress
    disp_vals, disp_maxs, disp_ids, disp_keys = _get_values_and_find_max( R_disps )
    stress_vals, stress_maxs, stress_ids, stress_keys = _get_values_and_find_max(
R_stres )

    # Define objective function. Use absolute values.
    objective_function =  np.abs(disp_maxs[2])

    return objective_function

def Constraint_Function(Results):
    """This function defines constraint function for optimization."""
    # # Get Results
    # R_coord = Results[0][-1]
    # R_elems = Results[1][-1]
    # R_disps = Results[2][-1]
    R_stres = Results[3][-1]

    # Get Stress
    stress_vals, stress_maxs, stress_ids, stress_keys = _get_values_and_find_max(
R_stres )

    th_cr_max = 0
    el_max = 0
    S_el_max = 0
    for el in stress_ids:
        S_el = stress_vals[el,:]
        th_cr = Tsai_Hill_Cr(S_el)
        if th_cr_max < th_cr:
            th_cr_max = th_cr
            el_max = el
            S_el_max = S_el

    # th_cr_max must be smaller than 1
    # const_fun should be smaller than 0
    const_fun = th_cr_max-1  # (th_cr_max-1) <= (0)  # format about zero

    if th_cr_max >= 1:
        S_el_vm = Stress_VonMiss(S_el_max)
        print(f"\n   *** Tsai_Hill ({th_cr_max:+5.3f}) Failure at El: {el_max:4d} *** ")
```

```
        print(f"   S_vm: {S_el_vm:+8.3f}    Sij:", S_el_max, "\n")

    return const_fun

class Composite_Shell_Problem(ElementwiseProblem):

    def __init__(self):
        self.ni = 0 # Iteration counter
        n_var0 = 12 # Number of composite layers
        xl = np.ones(n_var0)*-90 # Lower Th Limit
        xu = np.ones(n_var0)*90  # Upper Th Limit
        super().__init__(n_var=n_var0, n_obj=1, n_ieq_constr=1, xl=xl, xu=xu)

    def _evaluate(self, x, out, *args, **kwargs):
        print("vvv-------------------- iteration :",self.ni,"--------------------vvv")
        Results = Run_FEA_Main(x)
        out["F"] = Objective_Function(Results)
        out["G"] = Constraint_Function(Results)

        # Print-Outs
        # Change explanations if needed.
        self.ni += 1
        print("   ")
        print(f">>> Objectives   : F(x) = {out['F']:+6.3f}    Max Z-Disp")
        print(f">>> Constraints  : G(x) = {out['G']:+6.3f}    Tsai_Hill =
{out['G']+1:6.3f}")
        print( ">>> Orientations : ")
        print( "    x = [", end="")
        for x_i in x:
            print(f" {x_i:0.3f} ", end="")
        print("] [deg]")
        print("   ")

def Run_Minimizer():
    """ Run FEA for given [x] orientation values."""

    from pymoo.algorithms.moo.nsga2 import NSGA2

    problem = Composite_Shell_Problem()

    algorithm = NSGA2( pop_size = 50, )

    res_min = minimize(problem,
                algorithm,
                ('n_gen', 30),
                seed = 10,
                verbose=False,
                save_history=True)
```

```
    # Optimization Results
    X_opt = res_min.X
    F_opt = res_min.F
    G_opt = res_min.G

    print("\n---------------------------\n" )
    print("*** Optimization Finished. ***\n" )
    print("   > Optimized X :" )
    print( "\n   x = [", end="")
    for x_i in X_opt:
        print(f" {x_i:0.1f} ", end="")
    print("] [deg]\n")
    print("   > Optimized F :", F_opt )
    print("   > Optimized G :", G_opt )
    print("   > Tsai_Hill   :", G_opt+1 )
    print("\n*** Optimization Finished. ***" )
    print("\n---------------------------\n" )

    return X_opt, F_opt, G_opt, res_min, problem, algorithm

#%% Main Script
# Run Optimization
X_opt, F_opt, G_opt, res_min, problem, algorithm = Run_Minimizer()

#%% Plot Script

Results = Run_FEA_Main(X_opt)

sm_Nodes, sm_Disps, sm_Stres = Plot_Elements(Results)

hist_ni, hist_FG, hist_X = Plot_History(res_min)
```

ÖZGEÇMİŞ

Fatih ÖZTÜRK

ÖĞRENİM BİLGİLERİ

Yüksek Lisans 2022-2025	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Makine Mühendisliği Anabilim Dalı, Antalya
Lisans 2002-2007	Karadeniz Teknik Üniversitesi Mühendislik Fakültesi, Makine Mühendisliği Bölümü, Trabzon

MESLEKİ VE İDARİ GÖREVLER

Başmühendis 2022-Devam Ediyor	TUSAŞ-Türk Havacılık ve Uzay Sanayii Antalya Yapısal Mühendislik Ekibi, Antalya
Tasarım Mühendisi 2017-2022	TUSAŞ-Türk Havacılık ve Uzay Sanayii KAAN Projesi, Ankara
CAD Uzmanı 2008-2017	TUSAŞ-Türk Havacılık ve Uzay Sanayii PLM Ekibi, Ankara