

**UÇAK İTKİ SİSTEMLERİNİN TASARIM VE PERFORMANSININ SEZGİSEL
YÖNTEMLERLE OPTİMİZASYONU**

ALTUĞ PİŞKİN

DOKTORA Tezi

Uçak Gövde Motor Bakım Anabilim Dalı

Danışman: Prof. Dr. Önder TURAN

(İkinci Danışman: Doç. Dr. Tolga BAKLACIOĞLU)

Eskişehir

Eskişehir Teknik Üniversitesi

Lisansüstü Eğitim Enstitüsü

Haziran 2022

17/06/2022

DANIřMAN ONAYI

Daniřmanlıęını yrttęm Doktora ęrencisi Altuę PİřKİN, UÇAK İTKİ SİSTEMLERİNİN TASARIM VE PERFORMANSININ SEZGİSEL YNTEMLERLE OPTİMİZASYONU bařlıklı tez alıřmasını tamamlamıřtır. Hazırlamıř olduęu tez tarafımda incelenmiř ve ęrencinin tez savunma sınavına alınması bilimsel ve etik aıdan uygun grlmřtir.

Tez Daniřmanı

Prof. Dr. nder TURAN

ÖZET

UÇAK İTKİ SİSTEMLERİNİN TASARIM VE PERFORMANSININ SEZGİSEL YÖNTEMLERLE OPTİMİZASYONU

Altuğ PİŞKİN

Uçak Gövde Motor Bakım Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Haziran 2022

Danışman: Prof. Dr. Önder TURAN

(İkinci Danışman: Doç. Dr. Tolga BAKLACIOĞLU)

Bu tez kapsamında; Karınca kolonisi ve Parçacık sürü optimizasyon algoritmalarından faydalanılarak çok boyutlu ve çok hedefli hibrit bir optimizasyon algoritması ACOPSO (Ant Colony Optimization – Particle Swarm Optimization) geliştirilmiştir. Bu algoritma ile özellikle havacılıkta kullanılan itki sistemlerinin tasarım ve optimizasyonunda ihtiyaç duyulan doğrusal olmayan ve birçok yerel çözüme sahip zorlayıcı problemlerin çözülmesi amaçlanmıştır. İkinci aşamada bu kod kullanılarak turbojet, turboşaft, turboprop ve turbofan motorların performans optimizasyonu yapılmıştır. Özellikle gaz türbinli motorların tasarım dışı noktalardaki uyumluluk ve performans tasarımının iyileştirilebilmesine yönelik problemler çözülmüştür.

Tasarım çalışmasına örnek olarak; ticari olarak yaygın bir kullanıma sahip olan turboprop motorunun, serbest türbinli ve ayırık akışlı turbofan varyantlarının modellenmesi ve hesaplamaları yapılmıştır. Öncelikle tek şaftlı bir turboprop motorun kalkış, seyir ve dolanma noktalarında tasarım dışı hesaplamaları ACOPSO ile gerçekleştirilmiştir. Ardından, bu motor serbest türbinli turboşaft konfigürasyona dönüştürülmüştür. Daha sonra bu serbest türbinli motorun ayırık akışlı bir turbofan motora dönüştürülmesi ile elde edilecek maksimum itki bulunmuştur. Literatürden bulunan bir fan haritası bu optimum fan basınç oranı, by-pass oranı ve hava debisine göre ölçeklenmiş ve tasarım dışı noktalarda fan performansı hesaplanmıştır.

Anahtar Sözcükler: Karınca Kolonisi Optimizasyonu, Parçacık Sürüsü Optimizasyonu, Turbojet, Turboprop, Turboshaft, Turbofan

ABSTRACT

OPTIMIZATION OF AIRCRAFT PROPULSION SYSTEM DESIGN AND PERFORMANCE BY MEANS OF METAHERUSTIC METHODS

Altuğ PİŞKİN

Department of Aircraft Airframe and Powerplant Maintenance

Eskişehir Technical University, Institute of Graduate Programs, June 2022

Supervisor: Prof. Dr. Önder TURAN

(Co-Supervisor: Assoc. Prof. Dr. Tolga BAKLACIOĞLU)

Within the scope of this thesis; a novel multi-dimensional and multi-objective hybrid optimization algorithm ACOPSO (Ant Colony Optimization – Particle Swarm Optimization) has been developed by utilizing the Ant Colony and Particle Swarm Optimization algorithms. With this algorithm, it is targeted to solve non-linear, multi-modal challenging optimization problems, especially encountered in the aerospace propulsion system design and optimization. In the second phase, performance optimization of turbojet, turboshaft, turboprop and turbofan engines is performed using this code. Particularly, problems related to the gas turbine engine off-design matching and performance improvement have been investigated.

As an example to the design study; a well known commercial turboprop engine and its turboshaft and turbofan variants have been modeled and their performance calculations have been performed. At first, single shaft turboprop engine off-design calculations at take-off, cruise and loiter conditions have been realized with ACOPSO. Next, this engine has been converted to the free turbine turboshaft configuration. Afterwards, the maximum thrust that can be obtained by converting this turboshaft engine into an unmixed flow turbofan engine is achieved. A fan performance map obtained from the literature and scaled using the optimum fan pressure ratio, by-pass ratio and air mass flow rates, and off-design operating conditions of the turbofan engine have been calculated.

Keywords: Ant Colony Optimization, Particle Swarm Optimization, Turbojet,
Turboprop, Turboshaft, Turbofan

TEŞEKKÜR

Tez çalışmamın konu seçiminden başlayıp, hazırlanmasında, yönlendirilmesinde ve değerlendirilmesinde bana her konuda yol gösteren ve yardımcı olan değerli danışman hocalarım Sayın Doç Dr. Tolga BAKLACIOĞLU'na ve Sayın Prof. Dr. Önder TURAN'a teşekkürü bir borç bilirim.

Değerli düşünceleri ve önerileriyle tezimin daha iyi bir noktaya gelmesini sağlayan tüm jüri üyelerine ayrı ayrı teşekkürlerimi sunarım.

Altuğ PİŞKİN

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan “bilimsel intihal tespit programı”yla tarandığını ve hiçbir şekilde “intihal içermediğini” beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Altuğ PİŞKİN

İÇİNDEKİLER

Sayfa

BAŞLIK SAYFASI	i
JÜRİ VE ENSTİTÜ ONAYI.....	ii
DANIŞMAN ONAYI	iii
ÖZET	iv
ABSTRACT.....	v
TEŞEKKÜR	vi
ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ.....	vii
İÇİNDEKİLER	viii
TABLolar DİZİNİ	xi
ŞEKİLLER DİZİNİ	xiii
SİMGELER VE KISALTMALAR DİZİNİ	xvi
1. GİRİŞ	1
1.1. Sivil Havacılıkta Uçak İtke Sistemleri	1
1.2. Tezin Amacı ve Kapsamı	1
1.3. Tezin Özgünlüğü	3
1.4. Tezin Güncelliği.....	4
1.5. Tezin Ekonomik ve Sosyal Katkıları.....	4
2. SEZGİSEL YÖNTEMLER	6
2.1. Karınca Kolonisi Optimizasyonu.....	8
2.2. Parçacık Sürüsü Optimizasyonu	9
2.3. Levy Uçuşu.....	12
2.4. ACO ve PSO Melez (Hibrit) Yöntemleri	12
2.5. Yeni Bir Hibrit Optimizasyon Yöntemi: ACOPSO	13
3. DOĞRULAMA ÇALIŞMALARI	17
3.1. Karınca Kolonisi Optimizasyon Yöntemi Doğrulaması	17
3.2. Türbin Kanatçıklarını Dizerek Dengeleme Yapılması	20

3.3. Regresyon Yöntemi İle Eğri Modellemesi	25
3.3.1. Goldsection optimizasyonu yöntemi ile eğri uydurma.....	28
3.3.2. Karınca kolonisi optimizasyonu ile eğri uydurma	29
3.3.3. Goldsection ve ACO sonuçlarının karşılaştırılması	30
3.4. Parçacık Sürüsü Optimizasyonu Yöntem Doğrulaması.....	33
3.4.1. Küresel konveks fonksiyon	33
3.4.2. Yumurta Kabı (Egg-holder) Fonksiyonu	34
3.5. ACOPSO Doğrulama Çalışmaları.....	37
3.5.1. Yumurta kabı fonksiyonu.....	37
3.5.2. Rosenbrock-Watson regresyon problemi.....	37
3.5.3. Bukin fonksiyonu no “f6”	38
3.5.4. Basınçlı kap tasarım problemi	39
3.5.5. Ankastre giriş tasarım problemi.....	40
4. UÇAK MOTORU İTKİ SİSTEMLERİ	42
4.1. Motor Tasarım Süreçleri.....	44
4.2. Motor Performans Optimizasyonu (Literatür incelemesi)	48
5. TERMODİNAMİK MOTOR PERFORMANS MODELİ	57
6. İTKİ SİSTEMİ PERFORMANS OPTİMİZASYONU	71
6.1. Tasarım Noktası Çevrim Optimizasyonu	72
6.2. Tasarım Noktası ve Tasarım Noktası Dışı Çevrim Optimizasyonu	78
6.2.1. NPSS turbojet modellemesi	79
6.2.2. Jetcat P400 turbojet modellemesi	84
6.3. Motor Varyantlarının Tasarımı	88
6.4. Tek Şaftlı Turboprop Motorunun Modellemesi	92
6.5. Serbest Türbinli Varyant Motorun Modellenmesi	95
6.6. Turbofan Motoru Varyantının Modellemesi.....	98
6.7. Turbofan Motoru Tasarım Dışı Hesaplamaları.....	104
7. SONUÇ VE ÖNERİLER.....	108
KAYNAKÇA.....	111
EKLER	120

ÖZGEÇMİŞ



TABLÖLAR DİZİNİ

Sayfa

Tablo 3.1. Karınca kolonisi optimizasyon algoritması	18
Tablo 3.2. Analitik ve bulunan minimumların karşılaştırması	19
Tablo 3.3. Spline matrisi hesaplama adımları	25
Tablo 3.4. Spline katsayı matrisi	26
Tablo 3.5. MSE özet tablosu	33
Tablo 3.6. R özet tablosu	33
Tablo 3.7. Yakınsamış sonuçlar	34
Tablo 3.8. Uygulanan PSO algoritması	36
Tablo 3.9. Sonuçlar ve karşılaştırma tablosu	37
Tablo 3.10. Rosenbrock-Watson veri tablosu	38
Tablo 3.11. Rosenbrock-Watson regresyon problemi karşılaştırma tablosu	38
Tablo 3.12. Global minimum ve ACOPSO ile bulunan en iyi değer	38
Tablo 3.13. Optimizasyon tanımlama denklemleri	39
Tablo 3.14. Basıncılı kap optimizasyon sonuçları karşılaştırması	40
Tablo 3.15. Optimizasyon tanımlama denklemleri	41
Tablo 3.16. Ankastre giriş optimizasyon sonuçları	41
Tablo 4.1. Uçaklar ve kullanılan motor tipleri	42
Tablo 4.2. Motor seçiminde/tasarımında dikkate alınan A, B, C ve D noktaları	43
Tablo 4.3. Tasarım parametreleri	48
Tablo 4.4. Girdi parametreleri [58]	49
Tablo 4.5. Tasarım kısıtları [58]	49
Tablo 4.6. Maksimum net itki için optimum parametreler [59]	49
Tablo 4.7. Minimum özgül yakıt tüketimi için optimum parametreler [59]	50
Tablo 4.8. Optimizasyon parametreleri ve sınırları [60]	50
Tablo 4.9. Aşınmış motora ait parametre çarpanları	54
Tablo 5.1. Newton Raphson hesaplama matrisi parametreleri [73]	69
Tablo 6.1. Optimizasyon tanımlama denklemleri	72
Tablo 6.2. Sabit alınan parametreler	72
Tablo 6.3. Turbofan motor optimizasyonu analiz sonuç özeti	73
Tablo 6.4. Optimizasyon tanımlama denklemleri	74
Tablo 6.5. Sabit alınan parametreler	74

Tablo 6.6. Optimizasyon sonuçlarının karşılaştırılması.....	75
Tablo 6.7. Sabit alınan parametreler	75
Tablo 6.8. Değiştirilen verimler ile optimizasyon sonuçları.....	75
Tablo 6.9. Durum 1 Optimizasyon sonuçlarının karşılaştırılması	76
Tablo 6.10. Durum 2 Optimizasyon sonuçlarının karşılaştırılması	77
Tablo 6.11. Durum 3 Optimizasyon sonuçlarının karşılaştırılması	78
Tablo 6.12. A4 ve A8 parametrik analiz sonuçları özet tablosu	81
Tablo 6.13. Optimizasyon tanımlama denklemleri	83
Tablo 6.14. El hesabı ile optimizasyon sonucu karşılaştırması	83
Tablo 6.15. NPSS, el hesapları ve optimize edilmiş alanlarla bulunan sonuçlar.....	83
Tablo 6.16. Jetcat motoru üreticisi tarafından yayınlanmış veriler [76,77]	84
Tablo 6.17. P400-RX-G Katalog bilgileri [78]	85
Tablo 6.18. Referans çalışmada sunulan test verisi [76].....	85
Tablo 6.19. Çok hedefli tasarım gereksinimleri.....	85
Tablo 6.20. Optimizasyon tanımlama denklemleri	86
Tablo 6.21. El hesabı ile bulunan ve optimizasyon ile elde edilen alanların kıyaslaması.....	86
Tablo 6.22. Motor karşılaştırma tablosu	88
Tablo 6.23. T800-LHT-800 teknik özellikleri [81].....	91
Tablo 6.24. Tek şaftlı turboprop motor performans tablosu (Ghorbanian, et al., 2019) (AIAA, 2021)[84,85]	92
Tablo 6.25. Ölçekleme noktaları	93
Tablo 6.26. Optimizasyon tanımlama denklemleri	94
Tablo 6.27. Hesaplanan ve verilen çalışma noktası performans değerleri.....	95
Tablo 6.28. Optimizasyon tanımlama denklemleri	97
Tablo 6.29. Tek ve iki şaftlı motor performans sonuçlarının karşılaştırması	97
Tablo 6.30. Test verileri ile hesaplamaların karşılaştırılması	98
Tablo 6.31. Optimizasyon tanımlama denklemleri	100
Tablo 6.32. Türbin alanlarının değiştirilmesinin etkisi.....	103
Tablo 6.33. Seçilen fan tasarım noktası değerleri	104
Tablo 6.34. Optimizasyon tanımlama denklemleri	105
Tablo 6.35. Turbofan motor optimizasyon hesaplama sonuçları	106

ŞEKİLLER DİZİNİ

Sayfa

Şekil 2.1. Sezgisel yöntemlerin tarihi gelişimi [6]	7
Şekil 2.2. Karıncaların yiyecek arama davranışları	8
Şekil 2.3. Karıncaların yiyecek yolu üzerinde yoğunlaşmaları	8
Şekil 2.4. Parçacıkların pozisyonları, hız vektörleri ve etki eden ivme vektörleri	10
Şekil 2.5. ACOPSO algoritması	14
Şekil 2.6. Parçacık ve yiyecek kaynağı nesnelerinin özellikleri	15
Şekil 2.7. ACOPSO şematik gösterim (14 parçacık 4 yiyecek kaynağı)	16
Şekil 3.1. Birinci ve ikinci turdaki seçimler	18
Şekil 3.2. Optimizasyon sonucunda ulaşılan noktalar	19
Şekil 3.3. İterasyon bilgi penceresi görüntüsü	20
Şekil 3.4. Disk üzerinde kanatçıkların dizilimi	20
Şekil 3.5. Kanatçık merkez kaç kuvvetleri ve toplamda hesaplanan “d” vektörü	21
Şekil 3.6. $f(b1,b2)= d $ grafiksel gösterimi	22
Şekil 3.7. Turlar sırasında karıncaların yerlerinin değişimi	22
Şekil 3.8. Kanatçıkların ilk dizilişleri	23
Şekil 3.9. Optimizasyon algoritmaları ile elde edilen sonuçların karşılaştırılması	24
Şekil 3.10. Spline eğrisini oluşturan segmentlerin gösterimi	25
Şekil 3.11. Deneysel titanyum ısı verisi	27
Şekil 3.12. Spline eğri uydurma adımları	28
Şekil 3.13. Goldsection optimizasyon yöntemi	28
Şekil 3.14. Karınca kolonisi optimizasyon yönteminin uygulanması	30
Şekil 3.15. D1 ilk diziliş (NS=7)	31
Şekil 3.16. D1-O1 optimizasyon sonucu (NS=7)	31
Şekil 3.17. D1-O2 optimizasyon sonucu (NS=7)	31
Şekil 3.18. D2 ilk diziliş (NS=7)	32
Şekil 3.19. D2-O1 optimizasyon sonucu (NS=7)	32
Şekil 3.20. D2-O2 optimizasyon sonucu (NS=7)	32
Şekil 3.21. Konveks fonksiyon grafiği	34
Şekil 3.22. Yumurta kabı fonksiyonu renklendirilmiş grafiği	35
Şekil 3.23. Parçacıkların hareketleri ve yakınsamasını gösteren ekran görüntüleri	35
Şekil 3.24. Bukin fonksiyonu no “f6”	39

Şekil 3.25. Basıncılı kap şematik gösterimi.....	40
Şekil 3.26. Ankastre giriş şematik gösterimi.....	40
Şekil 4.1. Havacılıkta yaygın olarak kullanılan uçak tipleri.....	42
Şekil 4.2. Tipik bir ticarî turbofan uçuş profili [45]	43
Şekil 4.3. Helikopter motoru çalışma noktaları [46]	44
Şekil 4.4. Tipik gaz türbini tasarım adımları [48]	47
Şekil 4.5. Klâsik bir turbojet motorun bileşenleri	48
Şekil 4.6. İki şaftlı turbojet motoru şematik gösterimi ve enstrümantasyon noktaları [63].....	51
Şekil 4.7. Gaz akış yolu analizinde kullanılan tipik sensörler [69]	55
Şekil 4.8. TEMPER programı akış diyagramı [69]	56
Şekil 5.1. Bir turbojet motorunun istasyon numaraları	57
Şekil 5.2. Turbojet motoru T-s diyagramı	57
Şekil 5.3. Ayrık akışlı turbofan motor şematik gösterimi	62
Şekil 5.4. Turbojet motoru bileşenlerinin performans haritaları ve eğrileri [69]	64
Şekil 5.5. Kompresör ve sistem çalışma basınç eğrileri [70]	65
Şekil 5.6. Sabit bir N devrindeki türbin çalışma eğrisi.....	66
Şekil 5.7. Turbojet motorda türbin ve egzoz eşleşmesi [71]	66
Şekil 5.8. GENENG programı akış şeması(Koenig & Fishbach, 1972) [73]	68
Şekil 5.9. Gaz üretici (a) ve serbest türbin (b) eşleştirme akış diyagramı [49].....	70
Şekil 6.1. Optimizasyon programı ile turbojet fonksiyonu arasındaki etkileşim	71
Şekil 6.2. Durum 1 Optimizasyon parametrelerinin iterasyon adımları ile değişimi	76
Şekil 6.3. Durum 2 Optimizasyon parametrelerinin iterasyon adımları ile değişimi	77
Şekil 6.4. Durum 3 Optimizasyon parametrelerinin iterasyon adımları ile değişimi	78
Şekil 6.5. NPSS örnek motor termodinamik modeli [75].....	79
Şekil 6.6. NPSS örneğine ait kompresör haritası[75].....	80
Şekil 6.7. NPSS örneğine ait türbin haritası [75]	80
Şekil 6.8. Türbin ve egzoz alanı parametrik değişimi	81
Şekil 6.9. S400SX3 kompresör haritası üzerinde çalışma noktaları [76,78]	84
Şekil 6.10. Hesaplanan değerler ile test verilerinin karşılaştırması.....	87
Şekil 6.11. Hesaplanan çalışma noktalarının türbin haritası üzerinde gösterimi.....	87
Şekil 6.12. Genel amaçlı türbin eğrisi	87
Şekil 6.13. Tek şaftlı turboprop motor kesiti (Garrett Service Manual, 2021)[79].....	89

Şekil 6.14. Serbest türbinli turboşaft motor kesiti (Becker & Frounfelker, 1992)	
[80]	89
Şekil 6.15. İki şaftlı ayırık akışlı turbofan motor kesiti (Davis, Beale, Lt. Col. Boyer, & Maj. O'Dowd, 2006) [81]	89
Şekil 6.16. İki kademeli santrifüj kompresör performans haritası [81]	90
Şekil 6.17. Örnek turboprop motorun farklı güçteki modelleri (Frignac & Privoznik, 1979) [83]	92
Şekil 6.18. Tek şaftlı turboprop bir motorun şematik gösterimi	93
Şekil 6.19. Ölçeklenmiş kompresör haritası	94
Şekil 6.20. Kompresör çalışma çizgisi ve iterasyon noktaları	95
Şekil 6.21. Türbin çalışma çizgisi ve iterasyon noktaları	95
Şekil 6.22. Serbest türbinli bir turboşaft motorun şematik gösterimi	96
Şekil 6.23. Tek şaftlı, iki şaftlı model sonuçları ve test verisi	98
Şekil 6.24. İki şaftlı, ayırık akışlı bir turbofan motoru şematik gösterimi	99
Şekil 6.25. İtke ve T4 değişimi	101
Şekil 6.26. FPR ve BPR değişimi	102
Şekil 6.27. TSFC Değişimi	102
Şekil 6.28. İtke ve T4 değişimi	102
Şekil 6.29. FPR ve BPR değişimi	103
Şekil 6.30. TSFC değişimi	103
Şekil 6.31. JT15D Fan karakteristiği [86]	104
Şekil 6.32. Ölçeklenmiş fan performans haritası	105

SİMGELER VE KISALTMALAR DİZİNİ

A	: Alan (m^2)
F	: İtke kuvveti (N veya kgf)
g	: Global best position (Bulunan en iyi pozisyon)
HPC	: High Pressure Compressor (Yüksek Basınç Kompresörü)
HPT	: High Pressure Turbine (Yüksek Basınç Türbini)
LHV	: Lower Heating Value (Yakıtın alt ısı değeri J/kg)
LM	: Lagrange Multipliers (Lagranj çarpanları)
LPT	: Low Pressure Turbine (Düşük Basınç Türbini)
$\dot{m}$	: Kütleli debi (kg/s)
M	: Mach sayısı
MFP	: Mass Flow Parameter (Kütleli akış parametresi)
P	: Toplam basınç (Pa)
R	: Evrensel gaz sabiti (J/mol/K)
SHP	: Shaft Horse Power (Şaft Beygir Gücü)
SFC	: Specific Fuel Consumption (Özgöl Yakıt Sarfıyatı g/s/kN)
T	: Toplam sıcaklık (K)
TSFC	: Thrust Specific Fuel Consumption (İtke başına düşen özgöl yakıt sarfıyatı g/s/kN)
V	: Hız (m/s)
VSV	: Variable Stator Vanes (Değişken açılı sabit kanatçıklar)
ϕ	: İvmelenme katsayısı
γ	: İzentropik gaz genleşme faktörü
w	: Kütleli hava debisi (kg/s)
ω	: Eylemsizlik ağırlığı
η	: Verim

1. GİRİŞ

1.1. Sivil Havacılıkta Uçak İtke Sistemleri

Gaz türbinli itke sistemleri sivil ve askeri havacılık alanında çok büyük bir öneme sahiptir. Uçaklar ve helikopterler yolcu ve yük taşıyabilmek, havada kalabilmek için motorlar tarafından üretilen şaft gücüne ve itkiye ihtiyaç duymaktadırlar. Gerek çevreye salınan emisyonlar, gerekse işletme ekonomisi açısından tüm motorların mümkün olan en düşük yakıt sarfiyatı ile çalışması istenmektedir. Havacılık endüstrisi bunu sağlayabilmek için uzun menzilli ve ağır uçaklarda yüksek irtifada çalışabilen, yüksek by-pass oranlı, yüksek yanma sıcaklığı ve yüksek sıkıştırma oranı gibi teknolojik olarak en son seviyelerde turbofan motorları kullanmaktadır. Orta menzilli uçaklarda ise düşük hızlarda çalışan daha verimli turboprop ve turboşaft motorlar kullanılmaktadır. Bunlar kullandıkları uçuş bloklarına ve çevreye göre en uygun basınç oranı ve yanma odası çıkış sıcaklığına sahip motorlardır. En uygun çalışma noktası belirlenirken düşük yakıt sarfiyatı, maksimum kalkış gücü, minimum motor ağırlığı, düşük bakım maliyetleri, egzotik olmayan alaşımlar kullanılması gibi seçenekler değerlendirilmektedir. Daha çok manevra kabiliyeti ve yüksek hızlara ulaşma ihtiyacı olan askeri motorlarda ise düşük by-pass oranlı, art-yanmalı, yüksek gaz sıcaklığında çalışabilen egzotik süper alaşımlara sahip yüksek özgül itkiye sahip motorlar kullanılmaktadır.

Sürdürülebilir havacılık için bio-yakıtların kullanılması, daha verimli motorların tasarlanması, dağıtılmış itke sistemleri ve elektriksel itke sistemleri çalışılmaktadır. Bununla birlikte günümüzde 50-60 yıl önce imal edilmiş uçak motorları hala kullanılmaktadır. Uçak işletmecileri, ellerindeki motorların daha iyi anlaşılması ve bunların daha verimli kullanılması için araştırmalar yapmaktadırlar. Mevcut uçakların daha verimli kullanılabilmesi için üzerlerindeki motorların karakteristiklerinin belirlenmesine ve detaylı bilgilere ihtiyaç duyulmaktadır. Üreticiler tarafından bu motorların performans programları sağlanmasına rağmen detaylı bilgiler verilmemektedir.

1.2. Tezin Amacı ve Kapsamı

Bu tezin amacı, Uçak itke sistemlerinin tasarım ve performansının sezgisel yöntemlerle optimizasyonudur. Tez iki ana aşamadan oluşmaktadır;

- i) Çok parametrelili ve çok hedefli bir sezgisel optimizasyon kodunun geliştirilmesi,

- ii) İtke sistemi fonksiyonları geliştirilerek bunların geliştirilen koda entegre edilmesi.

GasTurb, NPSS gibi ticarî programlar ile motor performans hesaplamaları yapılabilmektedir. Ayrıca motor üreticileri tarafından geliştirilen özgün kodlar ve üniversiteler tarafından geliştirilen performans kodları da mevcuttur. NASA tarafından eskiden geliştirilen bazı kodlar dışındaki kodların hiç biri açık kaynaklı değildir. Optimizasyon kodları, bazı programlama dillerinde açık kütüphane olarak verilmektedir. Araştırmacıların çoğu Matlab gibi ticarî yazılımlar kullanmaktadırlar. Bu teze özgü olarak Karınca Koloni Optimizasyonu prototip kodları Excel VBA programlama dili kullanılarak geliştirilmiştir.

Optimizasyon kodu ile motor performans hesaplamaları yapacak programın entegre olarak ve hızlı çalışabilmesi gerektiğinden ötürü, optimizasyon kodu ve uçak motoru performans fonksiyonları Code::Blocks 20.03 ortamında C++ programlama dili kullanılarak geliştirilmiştir. Grafik ve çizimlerin oluşturulması için SFML grafik kütüphanesinden ve internetteki açık kaynaklardan [1,2] faydalanılmıştır. Hazır olarak verilen bir optimizasyon kütüphanesinden faydalanılmamıştır.

Çalışma kapsamında sezgisel optimizasyon yöntemleri kullanarak uçak motorlarının performansının öncelikle modellenmesi ve sonrasında da geliştirilmesi amaçlanmıştır.

İlk olarak literatür araştırması ile geleneksel ve sezgisel yöntemler incelenmiştir. Sezgisel yöntemlerden karınca kolonisi optimizasyon yöntemi, basit fonksiyonlara uygulanarak elde edilen sonuçlar incelenmiştir. Karınca kolonisi optimizasyon yönteminin; daha çok ayrışık olarak adlandırılan ve ziyaret edilecek noktalar arası yolların belli olduğu veya seçeneklerin sınırlı olduğu problemler için uygun olduğu görülmektedir. (Singiresu, 2009) tarafından verilen örnekten yola çıkarak bu yöntem sayısal problemlere uygun hale getirilmiştir [3]. Daha sonra bir ve iki boyutlu optimizasyon uygulamaları gerçekleştirilmiş; sonraki dönemlerde ise, çok parametrelili uygulamalar yapılması konusunda çalışmalar yürütülmüştür. İki'den fazla parametre için mevcut algoritma verimsizleştiğinden dolayı; bu çalışmada başka metotlar da denenmiştir.

Uçak itki sistemlerinin performansının ve tasarımının optimizasyonu için ikiden fazla sayıda parametrenin kullanılması söz konusu olduğu göz önünde bulundurularak; bu çalışmada ayrıca, parçacık sürüsü optimizasyonu algoritması incelenmiştir. İlk olarak geliştirilen algoritma; seçilen test fonksiyonları [4] üzerinde uygulanmış ancak yeterli seviyede başarı elde edilememiştir. Literatürde, karınca kolonisi ve parçacık sürüsü optimizasyonu algoritmalarının birlikte kullanıldığı hibrit (melez) algoritmalar bulunmaktadır. Bu

durumdan esinlenilerek, geliştirilen karınca kolonisi optimizasyon parametreleri ile parçacık sürüsü optimizasyon algoritması birleştirilmiştir. Bu sayede daha başarılı bir algoritma elde edilmiştir.

Örnek karşılaştırma fonksiyonları ve mühendislik problemleri üzerinde başarılı sonuçlar elde edilerek turbojet ve turbofan uçak motorlarının performansını hesaplayan alt programlar hazırlanmıştır. Doğrulama için öncelikle literatürde mevcut motor test verileri kullanılarak broşür verileri veya tasarım noktasında performansını eşleştirecek kompresör, türbin ve egzoz büyüklükleri araştırılmıştır. Daha sonra bu bileşenler sabitlenerek farklı çalışma noktalarında motorun üretmiş olduğu itki veya güç hesaplanmıştır.

Son olarak bir turboprop motoruna ait kompresör haritası ve türbin karakteristik eğrileri kullanılarak türbin ve egzoz alanları hesaplanmıştır. Bu alanlar ile kalkış, seyir ve havada dolanma sırasındaki motor performansları hesaplanmıştır. Aynı motorun serbest türbinli olması durumu için aynı çalışma noktalarındaki performans değerleri hesaplanarak karşılaştırılmıştır. Daha sonra bu motorun iki şaftlı bir turbofan motora çevrilmesi halinde en yüksek itkiyi verecek konfigürasyon araştırılmıştır. Bu şartlara uygun şekilde bir fan performans haritası kullanılarak turbofan motorunun kalkış, seyir ve havada dolanma sırasındaki performansı hesaplanmıştır.

İkinci ve üçüncü bölümlerde optimizasyon kodunun geliştirilmesi sırasında yapılan çalışmalar ve geliştirilen kodun karşılaştırma fonksiyonları üzerindeki yeterliliği değerlendirilmiştir. Dördüncü bölümde, literatürde bulunan motor tasarım süreçleri ve performans optimizasyon çalışmaları anlatılmıştır. Beşinci bölümde motor termodinamik modeli ve kullanılan formüller açıklanmıştır. Altıncı bölümde turbojet, turboprop ve turbofan motor fonksiyonları geliştirilmiş ve bunların literatürdeki mevcut değerler ile karşılaştırması yapılmıştır. Bu motor fonksiyonları, on-design ve off-design parametrelerinin araştırılmasına imkân verecek şekilde hazırlanmıştır. Örneğin, turbojet bir motorda genel amaçlı bir kompresör haritası ve genel amaçlı bir türbin haritası kullanılarak tasarım noktası dışındaki bölgelerde bileşenlerin nasıl bir cevap vereceği görülebilmektedir. Son olarak yedinci bölümde sonuçlar ve çalışmanın devamı niteliğinde konu önerilerinde bulunulmuştur.

1.3. Tezin Özgünlüğü

Tez kapsamında;

- i. Karınca kolonisi optimizasyon yöntemi ve parçacık sürüsü optimizasyonu yöntemini kullanan pek çok özgün yazılım geliştirilmiştir.

- ii. Daha sonra bu iki yöntemin birlikte kullanıldığı ve literatürde olmayan yeni bir optimizasyon algoritması geliştirilmiştir. Literatürde bulunan zorlayıcı test fonksiyonları ve mühendislik problemleri kullanılarak bu algoritmanın başarısı gösterilmiştir.
- iii. Havacılıkta kullanılan itki sistemlerinin optimizasyonunun yapılabilmesi için; turbojet, turboprop, turboşaft ve turbofan motor hesaplama kodları yazılmış ve optimizasyon kodu ile entegre edilmiştir. Bu programlar sayesinde, tasarım noktasında ve tasarım noktası dışında performans hesaplamaları yapılabilmiştir. Literatürde tasarım dışı hesaplama yapabilen programlar genellikle ticarî veya kurumlara özel kodlardır. Bu açıdan bakıldığında, geliştirilen programın tasarım dışı hesaplama metodolojisi açısından fayda sağlayacağı düşünülmektedir.

Optimizasyon kodunun kullanılması ile hem itki sistemlerinin hesaplamaları yapılmakta hem de en uygun çalışma alanları bulunabilmektedir. Turbojet motorların bileşenlerinin bazılarının ölçeklenmesi ile yeni türev motorlar tasarlanması ve performans hesaplamalarının yapılması bu program sayesinde kolaylaşmış olacaktır.

1.4. Tezin Güncelliği

Mevcut uçak motorlarının verimliliklerinin araştırılması, motorların detaylı performans modellerinin oluşturulması ile mümkün olabilmektedir. Bunlar için bu motorların detay bilgilerine ihtiyaç duyulmaktadır. Ancak motordan motora olan değişkenlikler, kullanım sırasında aşınma etkileri ve birçok etken nedeni ile incelenen motorlardan anlamlı veriler çıkarılmasını zorlaştırmaktadır. Optimizasyon programları sayesinde motor performans eğrilerinin oluşturulması sağlanabileceği gibi bu değişkenliklerin nedenleri de araştırılabilecektir.

Tezde geliştirilen kodun iki şekilde fayda sağlayacağı değerlendirilmektedir;

- Havacılıkta mevcut olarak kullanılan uçak motorlarının modellenmesi ve bunların performans hesaplarının yapılması.
- Yeni itki sistemlerinin tasarımında en iyi verim ve en ekonomik kullanım noktalarına göre tasarım yapılması.

1.5. Tezin Ekonomik ve Sosyal Katkıları

- Mühendisliğin birçok alanında olduğu gibi uçak motoru tasarımında da çok disiplinli optimizasyon çalışmaları önemli bir yer tutmaktadır. Motor performans

hesaplamaları bütün tasarım çalışmalarının en başında yer almaktadır. Bu nedenle bu aşamada en uygun çalışma noktasının bulunması çok büyük önem taşımaktadır.

- Bilindiği gibi ülkemizde özgün havacılık itki sistemlerinin geliştirilmesine yönelik büyük bir motivasyon bulunmaktadır. Dünyada var olan itki sistemlerinin anlaşılabilmesi ve bunlarda daha ileri seviyede verimli ve ekonomik motorlar tasarlanabilmesi için optimizasyona ve özgün performans kodlarına ihtiyaç duyulmaktadır. Bu tezde geliştirilen optimizasyon kodu sayesinde çok karmaşık olmayan performans kodlarının kullanımı ile bu hesaplamalar yapılabilmektedir.
- Optimizasyon kodu çok boyutlu ve doğrusal olmayan kısıtlı optimizasyon problemlerinin çözümünde kullanılabilmektedir. Örnek olarak bir turboprop motorun türevlerinin performans hesaplamaları yapılmıştır ancak bunun ötesinde birçok tasarım probleminde kullanılması mümkündür.

2. SEZGİSEL YÖNTEMLER

Optimizasyon yöntemleri genel olarak geleneksel ve geleneksel olmayan yöntemler olarak sınıflandırılabilir [2,5]. Sezgisel yöntemler geleneksel olmayan, yeni yöntemler sınıfına girmektedir. Bunlar içerisinde Genetik Algoritma, Tavlama Simülasyonu, Parçacık Sürüsü Optimizasyonu, Karınca Kolonisi Optimizasyonu, Bulanık Sistemlerin Optimizasyonu ve Sinir Ağları Esaslı Optimizasyon başlıkları altında birçok yöntem bulunmaktadır. Genetik algoritma ile tavlama simülasyonu, tasarım uzayı içindeki minimumları büyük olasılıkla bulma yeteneğine sahip olan, diğer bir deyişle öne çıkan stokastik yöntemlerdir. Genetik algoritma, canlıların evrimsel olarak çoğalmalarını ve gelişmelerini taklit etmektedir. Girdi parametreleri gen haritası şeklinde kodlanarak ilk kuşak oluşturulmakta daha sonra bu kuşaktan başarılı olanlar yeni kuşağa aktarılmaktadır. Mutasyon mekanizması ile kuşaklar içerisinde olan genler değiştirilmekte böylece farklı genlerin oluşması sağlanmaktadır. Tavlama simülasyonu ise metallerin eriyip katılaşması sırasında oluşan çekirdekleşme işlemine benzemektedir. Metalin sıcaklığı yüksek iken atomların çekirdekleri kolaylık oluşup çözülebilmektedir. Sıcaklık düştükçe katılaşma artacağından atomlar serbestçe hareket edemezler. Sıcaklık hızlı şekilde düşerse tanecikler içerisinde tane kusurları oluşacaktır. Bu nedenle sıcaklığın düşürülmesi için makul bir oran belirlenir. Arama fonksiyonu olarak Boltzmann olasılık fonksiyonu ve yapay sıcaklık parametresi kullanılır.

Parçacık sürü optimizasyonu, doğadaki böcek, arı, kuş ve balık sürülerinin davranışını esas almaktadır. Sürü, tasarım uzayı içerisindeki belirli sayıdaki rastgele noktalardan oluşmaktadır. Her noktanın bir başlangıç konumu ve bir hız vektörü vardır. Daha sonra sürü içerisindeki bireyler, aralarından en iyi değeri bulan parçacığın bulunduğu konuma doğru hız vektörlerini güncelleyerek hareket devam ederler. Bu işlem sırasında sürünün bireyleri arasındaki mesafelere dikkat edilir, pozisyonların bitişmesine veya çok ayrışmasına izin verilmez. Sürüdeki tüm bireylerin hareketlerinin birbirleriyle uyumlu olmasına ve ortalama hız vektörünün bir yönde olmasına dikkat edilir.

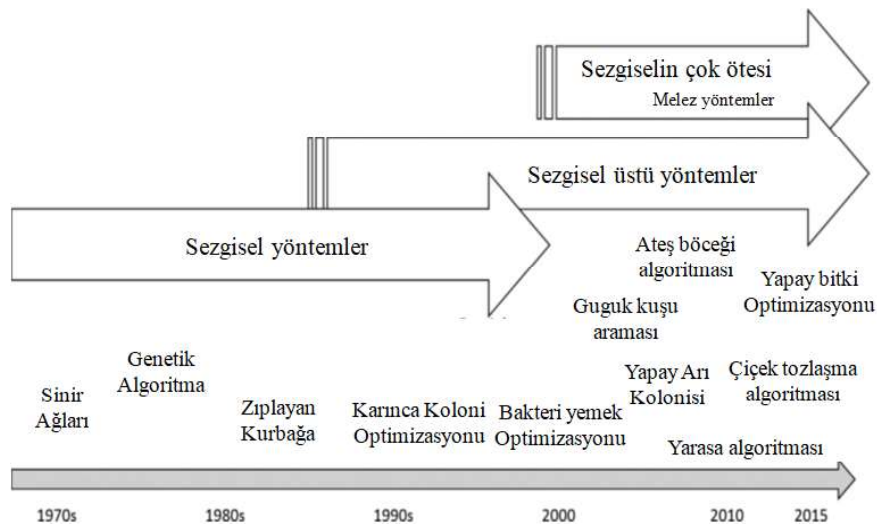
Karınca kolonisi optimizasyonu, karıncaların doğada yiyecek arayıp bulmak için yapmış oldukları davranışları taklit etmektedir. Yiyecek bulan karıncalar, bunu belli etmek için geçtikleri yollara kimyasal madde salgırlar. Bu kimyasal maddenin koku yoğunluğuna bağlı olarak diğer karıncalar bu yol civarında yiyecek kaynaklarının olduğunu anlara ve bu bölgeleri daha çok araştırırlar.

Bulanık sistemlerin optimizasyonu; gerçek hayatta bazı problemler matematiksel ifade yerine sözlü olarak ifade edilebilir. Bu durumlarda değerlendirmeler kesin sınırlara sahip değildir. Optimizasyon işlemleri buna bağlı olarak bulanık şekilde yapılır. Daha sonra keskin sınırları olan setler kullanılarak optimizasyon bölgesi sınırları daraltılır ve doğrusal optimizasyon yöntemleri kolaylıkla minimum değerler bulunur.

Sinir ağı yönteminde, birbirine bağlı sinir hücrelerinden oluşan bir ağ kullanılmaktadır. İlk tabaka giriş parametrelerinin verildiği hücrelerden, son tabaka ise çıktıların olduğu hücrelerden meydana gelir. Bunların arasında sinyalleri taşıyan ara katmanlar bulunur. Her sinir hücresi bir sigmoid fonksiyonu vasıtasıyla gelen sinyali sonraki tabakada bulunan hücrelere iletir. Bu fonksiyonlardaki katsayıların bulunması için önce sinir ağı eğitilir. Daha sonra istenilen bir girdi için hızlı bir şekilde çıktı hesaplanabilir.

Burada sayılanların dışında, yarası optimizasyonu, balina optimizasyonu, istilacı sarmaşık optimizasyonu, ateş böceği optimizasyonu, yasaklı arama vb. birçok yöntem bulunmaktadır.

(Kumar, 2016) sezgisel yöntemlerin tarihsel olarak ortaya çıkışlarını özetlemiştir (Şekil 2.1) [6]. Bu makalede sezgisel yöntemlerin ortaya çıkış tarihleri, kısa tanıtları ve literatürdeki yayınlarda ne kadar yaygın olarak kullanıldıkları anlatılmaktadır. Tarihçesi eski olan sezgisel yöntemlerin literatürde daha fazla geçiyor olması normal olarak görülebilir. Buna göre günümüzde daha çok yeni ve hibrit optimizasyon yöntemleri ortaya çıkmaktadır.

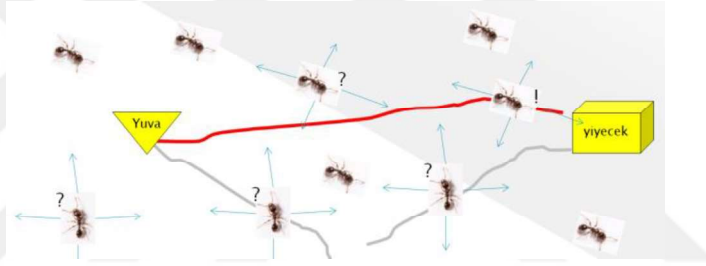


Şekil 2.1. Sezgisel yöntemlerin tarihi gelişimi [6]

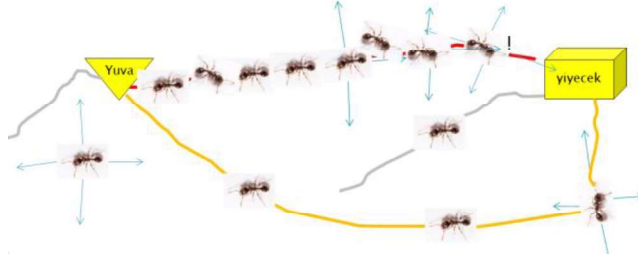
2.1. Karınca Kolonisi Optimizasyonu

(Dorigo & Stützle, 2004) tarafından verilen; Ant-density, Ant-quantity ve Ant-cycle önerileri ile başlamıştır [7]. Ant-cycle adındaki bu günkü Ant-system olarak bilinen yöntem diğer ikisine göre tercih edilen yöntem olmuştur. Bu yöntem karıncaların yiyecek arama davranışları örnek alınarak geliştirilmiştir. Karınca kolonisi kendi içlerinde farklı görevleri olan bireylerden oluşur bunlar; kraliçe karınca, işçi karıncalar, asker karıncalar ve öncü karıncalardır.

Öncü karıncalar yiyecek konumlarını belirlemek için yuva etrafında keşif gezisine çıkarlar. Geçtikleri yerde buldukları yiyecek kalitesine bağlı olarak Feromon adı verilen kimyasal madde salgırlar. Bu kimyasal maddeler yiyecek toplamaya çıkan karıncalar tarafından algılanır ve yiyecekleri kolayca bulmalarını sağlar. Karıncalar yüksek feromonlu yolları tercih ederler ancak diğer yandan rastgele başka yönlere giden yollar da seçebilirler.



Şekil 2.2. Karıncaların yiyecek arama davranışları



Şekil 2.3. Karıncaların yiyecek yolu üzerinde yoğunlaşmaları

Diğer karıncalar da Feromon bırakabildikleri için, yiyecek olan yoldan daha çok karınca geçer ve Feromon seviyesi daha da artar. Böylelikle diğer karıncalar bu yolu ve çevresini daha çok tercih etmeye başlar. Feromon maddesi zamanla bozulmaya uğrar ve etkisi azalır. Bu mekanizma sayesinde, daha iyi bir yiyecek yolu keşfedildiğinde önceki yollardaki feromon seviyesi azalır ve bir süre sonra takip edilmez duruma gelir.

Yollardaki feromon seviyesinin belirlenmesi ile karınca kolonisinin hareketi yönlendirilmiş olur. Bunun için her karınca tarafından bulunan yiyecek kalitesi değerlendirilerek en iyi ve en kötü belirlenir. Farklı değerlendirme yöntemi kullanılarak

birçok karınca kolonisi optimizasyon yöntemi geliştirilmiştir. Bunlar; Ant Systems (AS), Elitist AS, Ant-Q, Ant Colony System, Max-Min AS, Rank Based AS, ANTS, Hyper-cube AS olarak sayılabilir. Karın koloni optimizasyonu yönteminde genel olarak aşağıdaki adımlar izlenir:

- Karınca sayısı ve parametre aralığı (izlenebilecek yollar) belirlenir.
- Feromon miktarları başlangıç haline getirilir.
- İstenilen yakınsama sağlanıncaya kadar aşağıdaki işlemler uygulanır.
- Karıncalar rastgele başlangıç konumlarına gönderilir.
- Gittikleri konumdaki fonksiyon değerleri bulunur.
- Yerel arama yapılır (isteğe bağlı yapılır).
- Sonuçlara göre feromonlar güncellenir.
- İstenilen tolerans sağlanıncaya kadar devam edilir. (Singiresu, 2009)

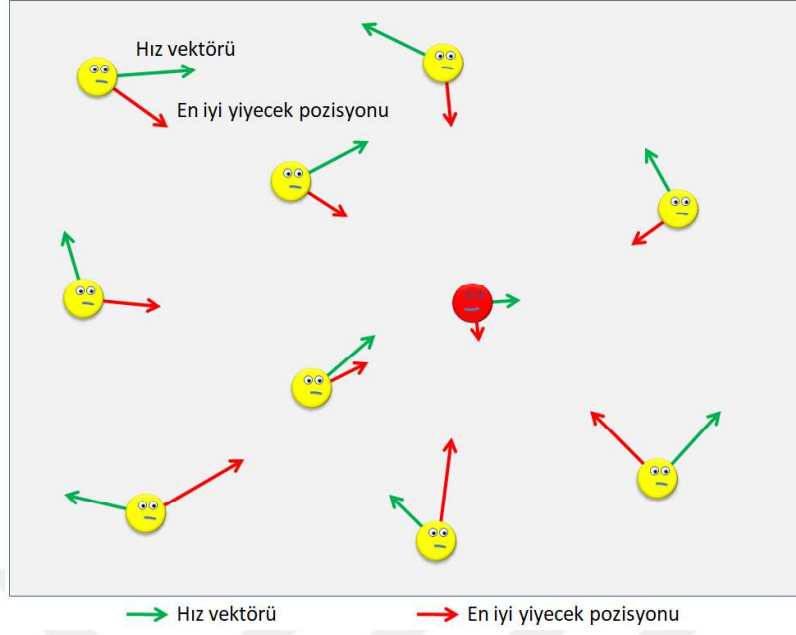
2.2. Parçacık Sürüsü Optimizasyonu

Parçacık sistemleri ilk defa bilgisayar grafikleri ile animasyon yapmak amacıyla kullanılmıştır. (Reeves, 1983) tarafından film efektleri oluşturulması amacıyla parçacıklardan oluşan bir sistem tanıtılmıştır [8]. Reeves parçacıkların taşıdıkları özellikler olarak şunları sıralamıştır;

- İlk pozisyon
- İlk hız vektörü (hız ve yön)
- İlk büyüklük
- İlk renk
- İlk saydamlık
- Şekil
- Hayatta kalma süresi/ömür (Reeves, 1983)

Literatürde parçacık sistemlerinin nasıl programlanabileceğini anlatan faydalı bilgiler verilmiştir [9]. Parçacık sürüsü optimizasyonu yöntemi ilk defa (Eberhart & Kennedy, 1995) tarafından geliştirilmiş ve yayınlanmıştır [10]. Uygulanması kolay ve başarılı bir algoritma olduğu için oldukça çok ilgi görmüştür.

Bu optimizasyon yöntemi; kuşlar ve balıklar gibi sürü halinde hareket ederek yiyecek bulan, çevreye karşı kendilerini koruyan ve varlıklarını başarılı şekilde sürdüren hayvan gruplarının hareketlerini taklit ettiği için Parçacık Sürüsü Optimizasyonu adı verilmiştir.



Şekil 2.4. Parçacıkların pozisyonları, hız vektörleri ve etki eden ivme vektörleri

Uygulama yöntemi açısından stokastik ve popülasyona dayalı bir yöntemdir. Popülasyonu oluşturan sürüdeki bireyler çok boyutlu bir parametre uzayı içerisinde rastgele konumlara yerleştirilir ve rastgele hız vektörleri ile harekete başlatılırlar (Şekil 2.4). Daha sonra kendi hızlarına bağlı olarak hareket ederken buldukları en iyi yiyeceğin konumu hafızalarında tutarlar. Aynı zamanda etkileşim içinde oldukları sosyal grubun hareketlerine göre belirli bir yönde hızlarını değiştirirler. Grup içinde bulunan en iyi değer ve konum bilgisi anında diğer bireyler ile paylaşılır.

$$\vec{V}_{t+1}^i = \vec{V}_t^i + \varphi_1 R_{1t}^i (\vec{p}_t^i - \vec{x}_t^i) + \varphi_2 R_{2t}^i (\vec{g}_t^i - \vec{x}_t^i) \quad (2.1)$$

Ortaya çıkışından bu güne kadar parçacık sürüsü optimizasyonunun birçok iyileştirilmiş çeşitleri yayınlanmıştır. Bu yöntemlerdeki iyileştirmeleri temel olarak; yakınsamayı hızlandırıcı, bölgesel optimum arama kabiliyetini artırıcı ve bölgesel optimuma takılmayı engelleyici müdahaleler şeklinde sınıflandırabiliriz. Aşağıda bu ana sınıflar açıklanmıştır;

- Yakınsama hızlandırıcı parametreler;
 - Eylemsizlik ağırlığı (ω): Parçacığın önceki hızının sonraki iterasyonda hesaplanacak olan hıza katkısını belirleyen katsayıdır. (Shi & Eberhart, 1998) eylemsizlik ağırlığı ifadesini hız terimine eklemiştir [11].

$$\vec{V}_{t+1}^i = \omega \vec{V}_t^i + \varphi_1 R_{1t}^i (\vec{p}_t^i - \vec{x}_t^i) + \varphi_2 R_{2t}^i (\vec{g}_t^i - \vec{x}_t^i) \quad (2.2)$$

φ_1 ; parçacığın kendi bulduğu en iyi yiyecek konumuna yönelmesi için ivmelenme katsayısını, φ_2 , sürünün tamamı içinde bulunan en iyi yiyecek noktasına yönelmesi için göre ivmelenme katsayısını ifade etmektedir. R_1 ve R_2 ; sıfır ve bir arasında rastgele üretilen sayılardır.

- Yapım faktörü (K): Yeni hesaplanan hız vektörünün ölçeklenerek hızın daha fazla artırılmasını veya yavaşlatılmasını sağlamaktadır.

$$\vec{V}_{t+1}^i = K [\vec{V}_t^i + \varphi_1 R_{1t}^i (\vec{p}_t^i - \vec{x}_t^i) + \varphi_2 R_{2t}^i (\vec{g}_t^i - \vec{x}_t^i)] \quad (2.3)$$

- Komşuluk ilişkileri;

- Statik komşuluk: Parçacık sürüsünün daha etkin arama yapabilmesi için kendi içinde gruplara bölünmesi ile komşuluk kavramı oluşmuştur. (Kennedy & Mendes, 2002), parçacık Sürüsü optimizasyonunda fazla sayıda komşuluğun basit problemlerde işe yaradığı, ancak karmaşık problemlerde küçük komşulukların kullanılmasının daha iyi sonuçlar verdiğini sonucuna varmıştır [12]. O güne değin geliştirilmiş olan hepsi birbirine bağlı, piramit, von Neumans, halka ve yıldız mimarîlerini incelemişlerdir. Bunların arasında en iyisinin von Neumans, en kötüsünün de orijinal PSO mimarîsi olan hepsi birbirine bağlı mimarîleri olduğunu belirtmişlerdir.
- Dinamik komşuluk: (Suganthan, 1999) orijinal PSO algoritmasına her parçacığın yarı çapına bağlı olarak yavaşça artan bölgesel dinamik komşuluklar eklemiştir [13]. Buna göre, eğer bir parçacık başka bir parçacığın çevresindeki yarıçap içerisine girerse aralarında bilgi alış verişi olmaktadır. İterasyonlar ilerledikçe, etkileşim yarıçapı büyümekte ve bir noktadan sonra tüm sürü birbiri ile iletişim halinde olmaktadır. Bu şekilde, komşulukların parçacıklar arasındaki mesafeye bağlı olarak belirlendiği duruma konuma bağlı komşuluk denilmektedir. Suganthan (1999) sadece konum değil aynı zamanda hızlanma katsayılarını da yavaş yavaş değiştirerek tasarım uzayındaki arama hızının değiştirilmesini de önermiştir.

- Duraklama faktörü;

- Optimizasyon iterasyonları ilerledikçe, bazı parçacıkların hızı çok azalır ve neredeyse sıfır olur. Çok düşük hızlarda olduğu için bulunduğu pozisyondan uzaklaşamaz ve burada kalır. Buna parçacığın duraklaması

denir. Duraklamalar, global minimum değerlerin bulunmasını zorlaştıran durumlardır. (Van den Bergh & Engelbrecht, 2006) yakınsamanın eylemsizlik ve ilk hıza bağlı olduğunu ve her parçacığın yörüngesinin teorik bir noktaya mutlaka yakınsayacağını belirtmiştir [14]. Parçacıkların kararlı bir noktaya yakınsaması için ilk koşulları belirlemeye yardımcı olan sezgisel bir yaklaşım türetmiştir.

2.3. Levy Uçuşu

Levy uçuşu, kısaca, rastgele bir yönde ve rastgele mesafede hareket ederek arama yöntemidir [15]. Seçilen bir başlangıç noktasından rastgele bir yöne vektör seçilir. Bu yönde rastgele bir mesafe gidilerek yeni bir noktaya ulaşılır. Sonra tekrar rastgele bir yön ve rastgele mesafe kadar gidilir. Bu şekilde hareket sürdürülerek parametre uzayı içerisinde bir arama yapılmış olur. Bölgesel minimumlara takılmadığı için birçok stokastik optimizasyon algoritmasında kullanılmıştır. (Haklı & Uğuz, 2014) Levy uçuşlu Parçacık Sürüsü Optimizasyonu (LFPSO) adını verdikleri bir yöntem geliştirmişlerdir [16]. Parçacıklara, iyileşme olmayan her iterasyonda bir arttırılan bir değişken tanımlamışlar ve hangi parçacık limiti geçerse Levy uçuşu yöntemiyle parametre uzayında onu yeniden yerleştirilmişlerdir. Bu yöntemin normal PSO ya göre daha iyi sonuçlar verdiğini göstermişlerdir.

2.4. ACO ve PSO Melez (Hibrit) Yöntemleri

Parçacık sürüsü optimizasyonu çok popüler bir yöntemdir ve birçok alanda kullanılmıştır. Farklı problemler karşısında yakınsama başarısını arttırmak için algoritmada iyileştirici yaklaşımlar denenmiş ve hatta başka sezgisel yöntemlerle karma şekilde yeni yöntemler ileri sürülmüştür.

Literatürde ACO ve PSO yöntemlerinin birlikte kullanıldığı çalışmalar bulunmaktadır [17,18]. (Shelokar, Siarry, Jayaraman, & Kulkarni, 2007) PSO nun sürü davranışı ile ACO nun yiyecek arama davranışlarını sentezlemişlerdir [17]. Feromon mekanizmasını parçacıkları yönlendirmek için kullanmışlardır. Birinci aşamada PSO olarak ikinci aşamada ACO olarak çalışan bir algoritma geliştirmişlerdir. ACO aşamasında parçacıkların tümünün değerleri kontrol edilerek feromon seviyeleri belirlenmiştir. (Kaveh & Talatahari, 2008) kafes yapıların optimizasyonu için hibrit bir algoritma kullanmışlardır [18]. ACO yerel

olarak aramalar için kullanılmış ve bu bölgedeki parçacıkların PSO parametreleri buna göre yeniden düzenlenmiştir.

(Kıran, Gündüz, & Baykan, 2012) Türkiye'nin enerji ihtiyacını tahmin etmek için PSO ve ACO nun bir arada kullanıldığı yeni bir hibrit optimizasyon yöntemi (HAP – Hybrid Ant Colony and Particle Swarm) geliştirmişlerdir [19]. Hibrit algoritmada parçacıklar ve karıncalar aynı uzayda arama yapmakta ve birbirleri ile bilgi paylaşmaktadırlar. Bu bilgiler daha sonra parçacıkların ve karıncaların algoritmadaki araştırma parametrelerinin iyileştirilmesinde kullanılmıştır.

2.5. Yeni Bir Hibrit Optimizasyon Yöntemi: ACOPSO

Karınca koloni optimizasyonunda, öncelikle tasarım uzayındaki yolların ve bunlar arasında gidiş geliş yollarının belirlenmesi gereklidir. Ayrışık yollara sahip problemlerde bu nispeten daha kolayken sürekli sayısal problemlerde yol sayısı net değildir, yapılan denemelerde bir boyutlu problemler için bu sayının en az yedi-sekiz olması gerektiği görülmüştür. Hassas olarak bir sonuca yakınsama sağlanabilmesi için içlerinden en iyi yolun kendi içinde tekrar bölünerek algoritmanın yeniden çalıştırılması gerekmektedir. Bu tarz bir yaklaşım ile bir ve iki boyutlu problemlerin çözülmesi mümkün iken boyut sayısı arttıkça yöntem karmaşılaşmakta ve algoritmanın başarısı düşmektedir.

Parçacık sürüsü optimizasyonu yönteminde parçacıklar vektör uzayında istenilen yönde kolaylıkla hareket ettikleri için yolların tanımlanmasına gerek yoktur. Bu hareket yöntemiyle uzayın taranması ve sistematik olarak araştırılması daha kolay şekilde sağlanmaktadır. Vektörü oluşturan parametrelerin sayısının artması veya azalması algoritmayı değiştirmemektedir. Dolayısı ile çok parametrelili optimizasyon problemleri için bu yöntem ACO ya göre daha uygun görünmektedir. Bununla birlikte, uygulanan PSO algoritmasının yakınsama ve hassas sonuç elde etme konusunda istenilen seviyede başarılı olmadığı görülmüştür.

Bu gözlemlerden yola çıkılarak Karınca kolonisi optimizasyonu yöntemi ile Parçacık Sürüsü Optimizasyonu yöntemlerinin birlikte kullanıldığı, ACOPSO olarak adlandırılan hibrit bir yöntem geliştirilmiştir. Bu yöntemin algoritması Şekil 2.5'te gösterilmiştir. Bu algoritmada ilk olarak sürüdeki parçacık sayısı ve yiyecek kaynağı sayısı belirlenir. Sonra belirli bir sayıda iterasyon yapılarak elde edilen sonuçlar değerlendirilir. Karınca parametreleri düzenlenerek sürü pozisyonu güncellenmeye devam edilir. Yakınsama olup

olmadığı kontrol edilir. Eğer istenilen yakınsama sağlanamıyorsa sürü pozisyonları en son bilinen en iyi pozisyonlara doğru yörüngeler çizilecek şekilde güncellenir.



Şekil 2.5. ACOPSO algoritması

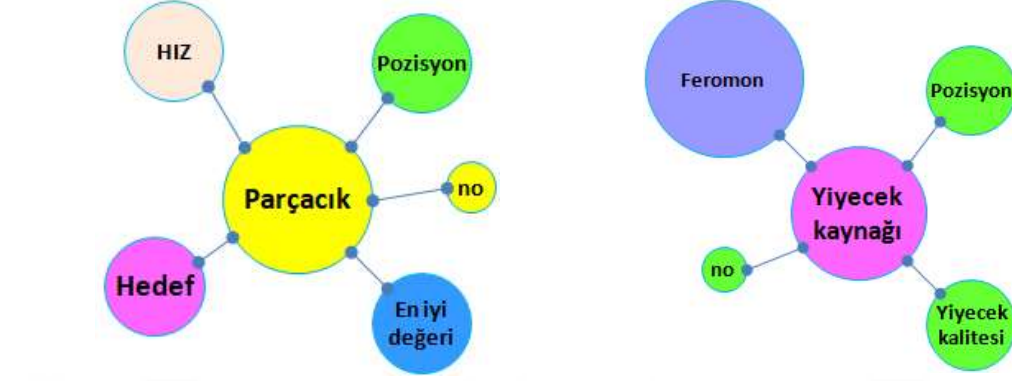
Yöntemde kullanılan parçacıkların ve yiyecek kaynaklarının özellikleri Şekil 2.6’da gösterilmiştir. Parçacıkların hepsi şu bilgilere sahiptir; pozisyon, hız, takip numarası, o ana kadar bulduğu en iyi değer ve hedef pozisyon. Büyük “X”, pozisyonu ifade eden, bağımsız değişkenlerden oluşan bir vektördür;

$$X = [x_1, x_2, x_3, \dots, x_i]^T, x_1 = [0,1], x_2 = [0,1], x_3 = [0,1], \dots, x_i = [0,1], \quad (2.4)$$

i =parçacık sayısı

Vektör içindeki her bir bağımsız parametre, x ile gösterilmiştir ve sıfır ile bir arasında normalize edilmiştir. Toplam parçacık sayısı i ile gösterilmiştir. Hız vektörü parçacığın bir iterasyonda, hangi yönde ne kadar gideceğini gösteren vektördür. Başlangıçta rastgele bir hız vektörü oluşturulur. Daha sonra hedefe bağlı olarak her iterasyonda yeni bir hız vektörü hesaplanır. Maksimum hız değeri fonksiyonun multimodal seviyesine bağlı olarak tasarım uzayının belli bir miktarı kadar seçilmiştir. Hesaplanan yeni parçacık pozisyonunun, sınırlarını aşıp aşmadığı her iterasyonda kontrol edilir ve eğer sınırları aşıyorsa tekrar sınırlar içerisine döndürülür ve hız vektörü tersine çevrilir.

$$V = [v_1, v_2, v_3, \dots, v_i]^T, v_1 = [-v_{\max}, v_{\max}], v_2 = [-v_{\max}, v_{\max}], v_3 = [-v_{\max}, v_{\max}], \dots, v_i = [-v_{\max}, v_{\max}] \quad (2.5)$$



Şekil 2.6. Parçacık ve yiyecek kaynağı nesnelerinin özellikleri

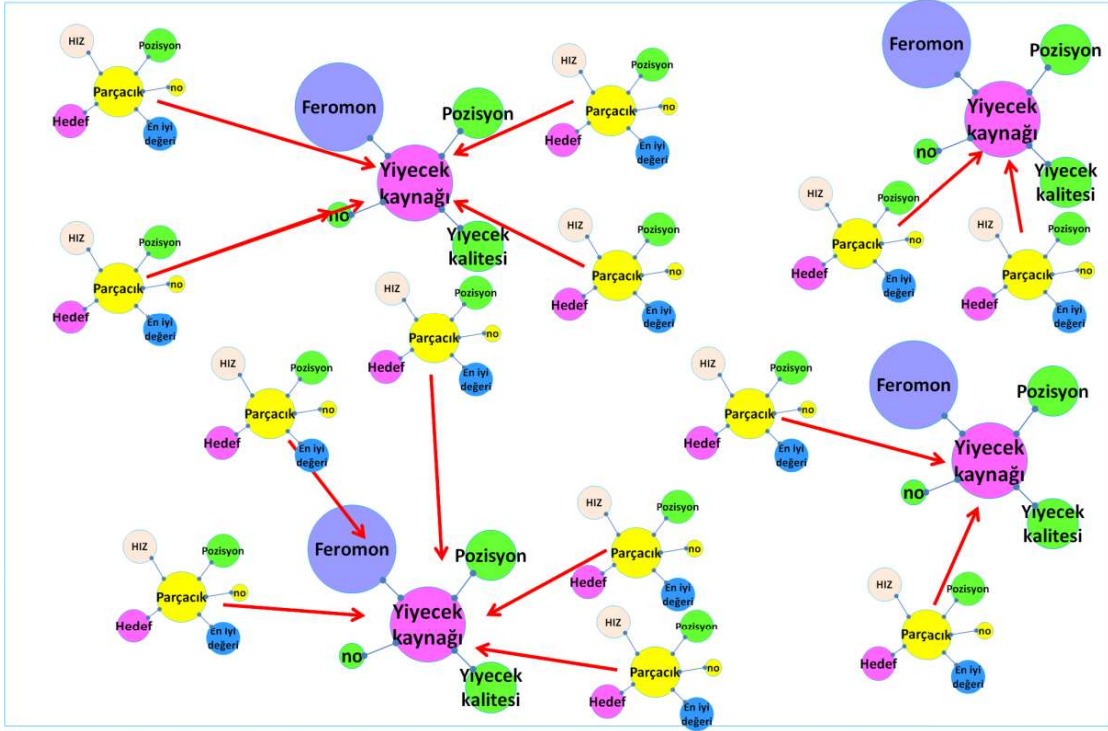
Yiyecek kaynakları; pozisyon, yiyecek kalitesi, feromon seviyesi ve takip numarasına sahiptir. Parçacıkların hedefi yiyecek kaynaklarının pozisyonlarıdır. Yiyecek kalitesi o pozisyondaki fonksiyon değeridir. İlk iterasyonda tüm yiyecek kaynaklarındaki feromon seviyeleri aynıdır. Eğer o hedefe doğru giden bir parçacık daha iyi bir fonksiyon değeri elde ederse yiyecek pozisyonu güncellenerek oraya taşınır.

$$Y = [y_1, y_2, y_3, \dots, y_j]^T, y_1 = [0,1], y_2 = [0,1], y_3 = [0,1], \dots, y_j = [0,1], \quad (2.6)$$

j=yiyecek kaynağı sayısı

Başlangıçta tüm parçacıkların hedefi en yakın oldukları yiyecek kaynağıdır (Şekil 2.7). Belirli bir iterasyondan sonra yiyecek kaynaklarındaki yiyecek kaliteleri değerlendirilerek feromon seviyeleri güncellenir. Karınca parametreleri güncellendiğinde parçacıkların feromon seviyelerine göre rastgele seçim yapmaları sağlanır. Yüksek feromona sahip olan yiyecek yollarının seçilme olasılığı daha fazla olacak şekilde bir düzenleme yapılmıştır.

Yiyecek kaynaklarının feromon seviyesi azaldıkça parçacıkların bunları seçme olasılığı da azalmaktadır. Hiçbir parçacığında takip etmediğin yiyecek kaynaklarına yeni bir konum atanarak sürekli aktif kalmaları sağlanmıştır. Yakınsama sağlanıncaya kadar bu şekilde iterasyonlara devam edilir.



Şekil 2.7. ACOPSO şematik gösterim (14 parçacık 4 yiyecek kaynağı)

3. DOĞRULAMA ÇALIŞMALARI

Tez boyunca öncelikle Karınca Kolonisi Optimizasyonu yöntemi, sonra Parçacık Sürüsü Optimizasyonu yöntemi ve daha sonrada hibrit yöntemin doğru çalışıp çalışmadığının görülmesi amacıyla test problemleri üzerinde optimizasyon uygulamaları yapılmıştır. Ayrıca bu algoritma bazı basit mühendislik problemlerine de uygulanmış ve elde edilen sonuçlar değerlendirilmiştir. Bu çalışmalar sırasıyla aşağıda anlatılmıştır.

3.1. Karınca Kolonisi Optimizasyon Yöntemi Doğrulaması

Çalışmada ilk olarak kullanılan bir boyutlu fonksiyon Eşitlik (3.1) de verilmiştir [5].

$$x \in [0,1] \text{ ve } f(x) = 1 - e^{4 \cdot x} \cdot (\sin(5 \cdot x \cdot \pi))^4 \quad (3.1)$$

Girdi parametre aralığı sekiz adet kutuya ayrılmıştır. Her kutunun bir numarası, olasılık sınırları ve fonksiyon değeri vardır. Başlangıçta tüm kutuların feromon seviyeleri aynıdır ve seçilebilme olasılıkları da eşit olarak ayarlanmıştır. Her karınca için 0 ile 1 arasında rastgele bir sayı üretilmektedir. Bu sayı kutulardan hangisinin sınırları içine düşüyorsa o kutunun fonksiyon değerine ulaşılmaktadır.

Şekil 3.1'deki örnekte gösterilen dört adet karınca, ilk turda 1, 3, 6 ve 7 numaralı kutuları seçmiştir. Üç numaralı karınca en iyi değeri bulmuştur ve bu değerlendirmeye göre feromon seviyeleri ayarlandığında ve kutuların olasılık sınırları yeniden belirlendiğinde, 6 numaralı kutu en geçiş aralığına sahip olmuştur. Kutuların X girdisi sınırları ise aynı kalmıştır.

Algoritmada kutu aralığı ayarlaması dengeli şekilde yapılmıştır. Böylelikle dört numaralı karıncanın daha iyi bir yer bulabilme imkânı olmuştur. Bu aralık gereğinden fazla geniş verilmiş olsaydı çok fazla karınca bu kutuyu seçeceğinden algoritma buraya takılacak ve erken yakınsama oluşacaktı. Erken yakınsamalar çözümün bulunmasında bir engel oluşturmaktadır bu nedenle istenilmemektedir. Uygulana algoritmanın adımları Tablo 3.1'deki akış diyagramı ile açıklanmıştır;

Karınca numarası	Kutu Olasılık sınırları	Kutu Numarası	Kutu x değeri	Fonksiyon Değeri	numarası	sınırları	Numarası	x değeri	Değeri
1	0.000	1	0.0625	0.3863	1	0.000	1	0.0625	
	0.125	2	0.1875		1	0.100	2	0.1875	0.9969
	0.250					0.200	3	0.3125	
2	0.375	3	0.3125	-2.2297	2	0.300	4	0.4375	0.4518
	0.500	4	0.4375			0.400	5	0.5625	
	0.625	5	0.5625			0.500	6	0.6875	-13.4746
3	0.750	6	0.6875	-13.4746	3	0.800	7	0.8125	
	0.875	7	0.8125	0.9626		0.900	8	0.9375	-19.3231
4	1.000	8	0.9375		4	1.000			

Şekil 3.1. Birinci ve ikinci turdaki seçimler

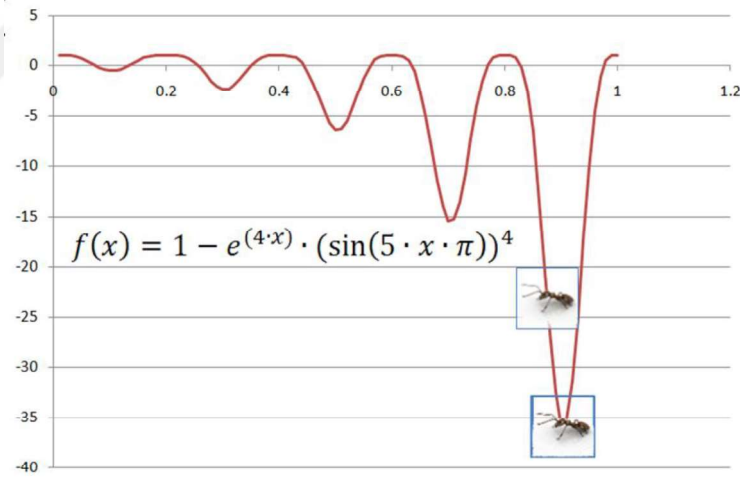
Tablo 3.1. Karınca kolonisi optimizasyon algoritması

Algoritmayı başlat
(a) { Fonksiyon giriş aralığını belirle alttaki döngüye ilet
(b) { For j = 1 To karınca_sayısı
karınca(j) için rastgele sayı üret
üretilen sayıya göre kutu_secimi yap
seçilen kutunun X değerini kullanarak fonksiyon değerini hesapla
maksimum ve minimum değerleri bul
Next
gcc_oran = Abs(min_deger / max_deger)
gcc_orana bakarak yakınsamayı kontrol et
toleranstan büyükse iterasyonlara devam aksi halde döngüden çık
Eğer (gcc_oran > 1) ise
yeni_pheromone_miktari = olcek * basarim * gcc_oran
Değilse
yeni_pheromone_miktari = olcek * basarim / gcc_oran
başarısı düşük kutuların feromon seviyesini bozulma faktörü kadar azalt
başarısı yüksek kutuların feromon seviyesini ölçeklendirme faktörü kadar arttır
toplam feromonu hesapla
kutuların feromon miktarına göre olasılık sınırlarını ayarla
tolerans yeterli değilse (b) den tekrar et
en iyi fonksiyon değerini veren X parametresini gönder
}
Belirlenen miktarda turu tekrarla (b) den devam et
Eğer ($X_{k+1} - X_i > \text{tolerans}$) ise
X parametresi civarındaki bölgeye yaklaşacak şekilde yeni giriş aralığını belirle (a)
dan devam et

Tablo 3.1. *Karınca kolonisi optimizasyon algoritması (devam)*

Değilse
Programı sonlandır
}

Olasılık sınırları geniş olan kutuların seçilme şansı daha fazladır ve olasılık sınırları dar olan kutuların seçilme şansı daha azdır. Ancak daha önceden gidilmemiş bölmelere giden ve daha iyi sonuç elde eden karıncalar olursa buradaki feromon seviyesi hızla artabilir. Böylece fonksiyonun minimum değerinin bulunduğu bölgeye yaklaşılmış olur. Karınca koloni optimizasyonu kısmı, algoritmadaki (b) döngüsüdür. Dıştaki (a) döngüsü bunu kontrol ederek x bölgesinin sınırları daraltmakta böylece daha hassas şekilde sayısal değer bulunması sağlamaktadır. Şekil 3.2’de bir optimizasyon çalıştırmasında elde edilen sonuç gösterilmiştir. Algoritmanın doğası gereği her koşuda farklı sonuç elde edilmektedir. Şekil 3.3’te verilen programın bilgi penceresinde (a) döngüsünün (b) döngüsünü 5 kez çağırdığı görülmektedir.

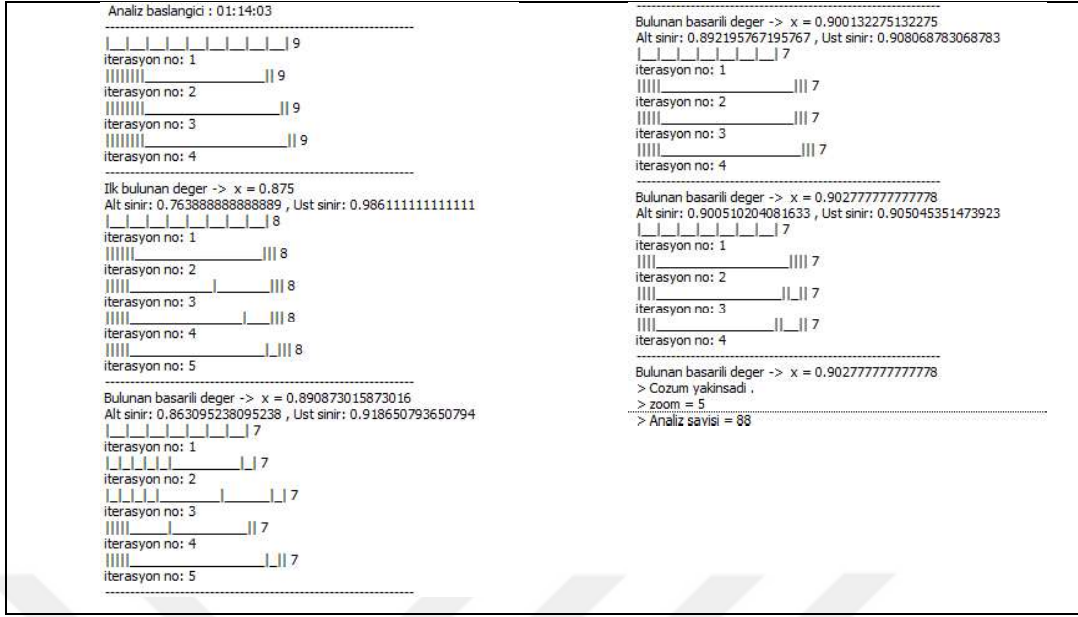


Şekil 3.2. *Optimizasyon sonucunda ulaşılan noktalar*

Tablo 3.2’de, örnek fonksiyonun analitik minimum değeri ile optimizasyonla bulunan minimum değer karşılaştırılmış ve aradaki fark yüzdesel olarak verilmiştir.

Tablo 3.2. *Analitik ve bulunan minimumların karşılaştırması*

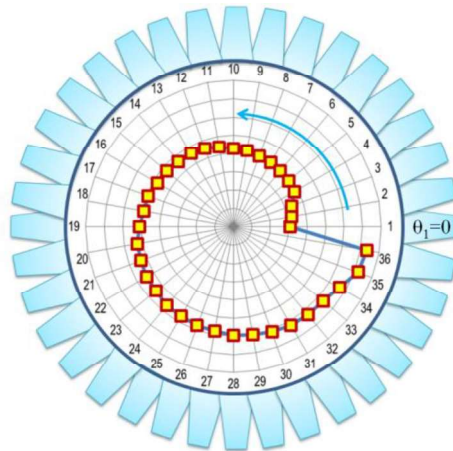
Girdi/Çıktı	Analitik minimum	Bulunan minimum	% Fark
x	0,904047385	0,90388796	0,018
f(x)	-35,8958924	-35,8954278	0,001



Şekil 3.3. İterasyon bilgi penceresi görüntüsü

3.2. Türbin Kanatçıklarını Dizerek Dengeleme Yapılması

Karınca kolonisi optimizasyon algoritması gerçek bir problem üzerinde uygulanmıştır. Bir jet motorunun türbin diski üzerinde, verilen bir kanatçık setinin en iyi balansı verecek şekilde dizilimi istenmektedir. Literatürde bu problem için verilen çeşitli yöntemler bulunmaktadır [20-27]. Bu problemi çözebilmek için mevcut karınca kolonisi optimizasyonu algoritması iki boyutlu ve ayrışık olarak çalışacak şekilde düzenlenmiştir. Şekil 3.4'te bir disk üzerinde ağırlık sıralamasına göre dizilmiş 36 adet kanatçık şematik olarak gösterilmiştir. Küçük karelerin merkezden uzaklıkları kanatçık ağırlığını ifade etmektedir.



$$\theta_i = (i - 1) \cdot \frac{2\pi}{n}$$

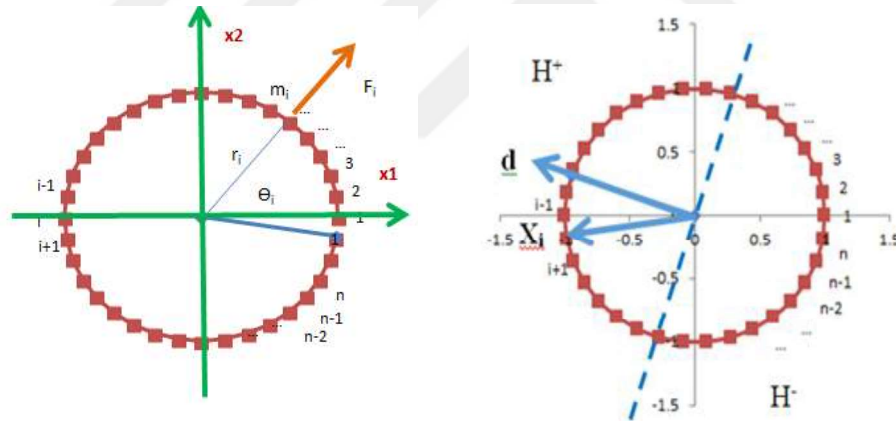
$$\{i = 1, 2, 3, \dots, n\}$$

$$n = 36$$

Şekil 3.4. Disk üzerinde kanatçıkların dizilimi

Kanatçık dizme problemi başlangıç sınır şartına ve uygulanan yöntemle ilgili olarak farklı sonuçlar veren ayrık değerli bir problemdir. Problemin tam olarak çözülmesi için bütün olasılıkların göz önüne alınması gerekmektedir ki bu ise az sayıda kanatçık için bile çok fazla zaman almaktadır. Literatürde bu tarz problemler NP-Hard olarak tanımlanmaktadır.

Algoritma şu şekilde çalışmaktadır; kanatçıklar disk üzerinde belirli yarıçapta ve disk üzerindeki kanalların olduğu belirli açılarda yerleştirilirler. Daha sonra seçilen iki kanatçık yer değiştirildiğinde oluşan balans değeri hesaplanmaktadır. Daha iyi bir balans değeri bulunursa değişim kalır ve sonra tekrar seçim yapılır. Kanatçık seçimlerinin sistematik olması için öncelikle B1 ve B2 setlerine ayrılmıştır. Disk üzerindeki kanatçıkların ağırlık merkezlerinin konumu ile ağırlıklarının çarpımları toplandığında disk dönüş ekseninden olan uzaklığını gösteren vektör “d” elde edilir, uzunluğu da $d_{norm} = ||d||$ dir. Bu vektöre dik olan çizginin H+ tarafında kalan kanatçıklar ağır olan B1 setini, H- tarafında kalan kanatçıklar hafif olan B2 setini oluşturmaktadır.



Şekil 3.5. Kanatçık merkez kaç kuvvetleri ve toplamda hesaplanan “d” vektörü

$$\sum_{i=1}^n \vec{F}_i = \sum_{i=1}^n m_i \cdot \omega^2 \cdot \vec{r}_i \quad (3.2)$$

$$F_{x1} = m_i \cdot \omega^2 \cdot |\vec{r}_i| \cdot \cos(\theta_i) \quad (3.3)$$

$$F_{x2} = m_i \cdot \omega^2 \cdot |\vec{r}_i| \cdot \sin(\theta_i) \quad (3.4)$$

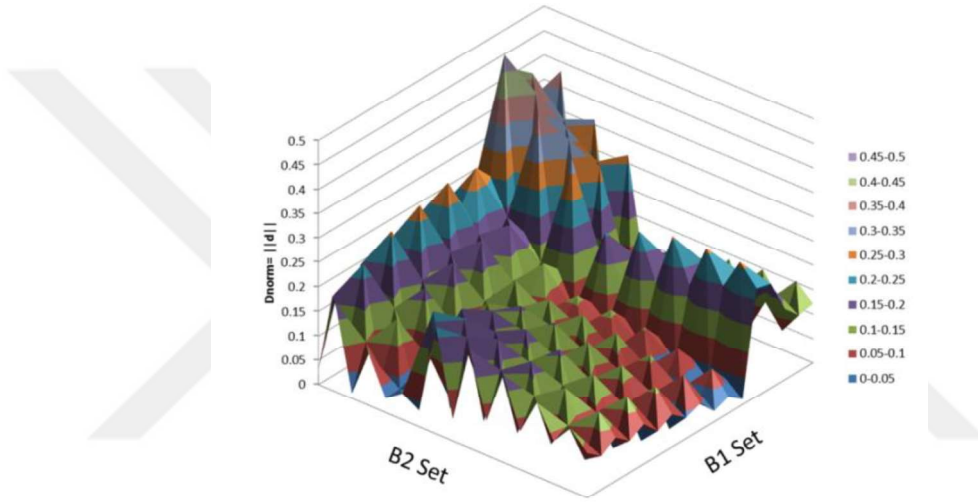
$$\vec{F}_{toplama} = m_{toplama} \cdot \omega^2 \cdot \vec{d} \quad (3.5)$$

$$\vec{d} = \frac{\vec{F}_{toplama}}{m_{toplama} \cdot \omega^2} \quad (3.6)$$

$b1 \in B1$ ve $b2 \in B2$,

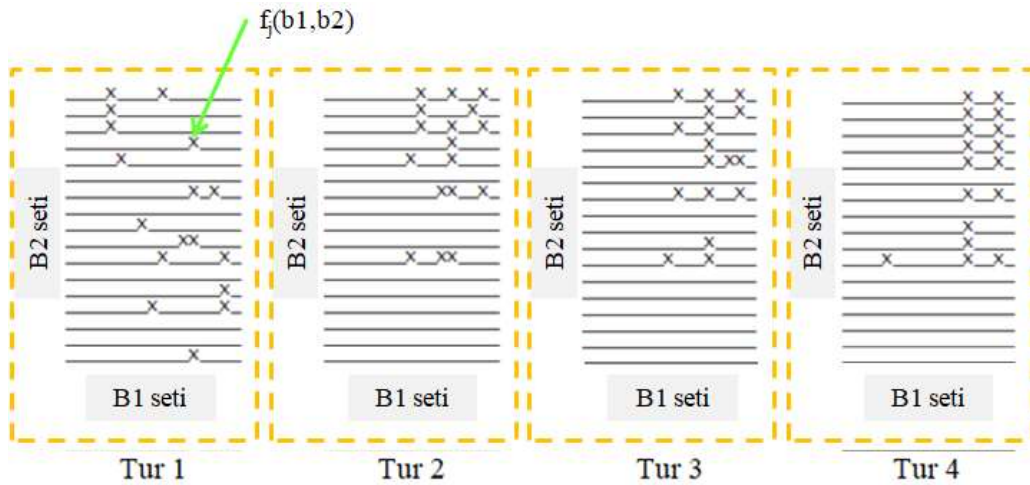
$f_j(b1,b2)=||d||$, j = karınca numarası

B1 ve B2 setlerinden seçilen birer kanatçık yer değiştirilerek oluşacak yeni balans değeri hesaplanır. Setler içerisindeki her $b1$ ve $b2$ çiftinin yer değiştirmesiyle hesaplanan d_{norm} değerleri Şekil 3.6’da grafiksel olarak gösterilmiştir.



Şekil 3.6. $f(b1,b2)=||d||$ grafiksel gösterimi

Şekil 3.7’de bir iterasyondaki dört tur sonunda karıncaların B1 ve B2 seti üzerinde hangi kanatçıkları seçtikleri “x” işaretleri ile gösterilmiştir.

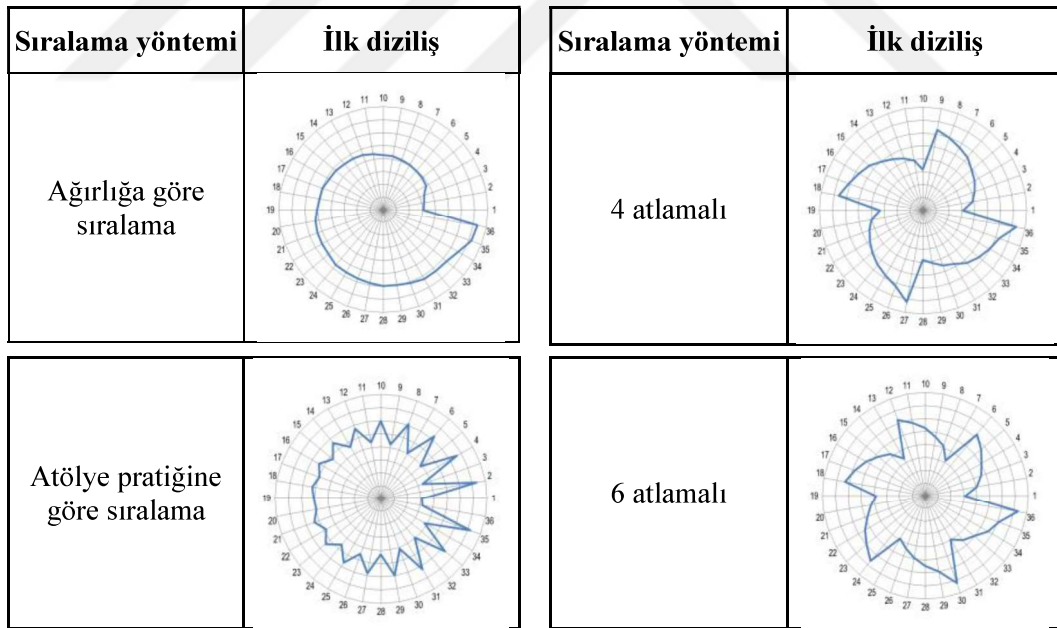


Şekil 3.7. Turlar sırasında karıncaların yerlerinin değişimi

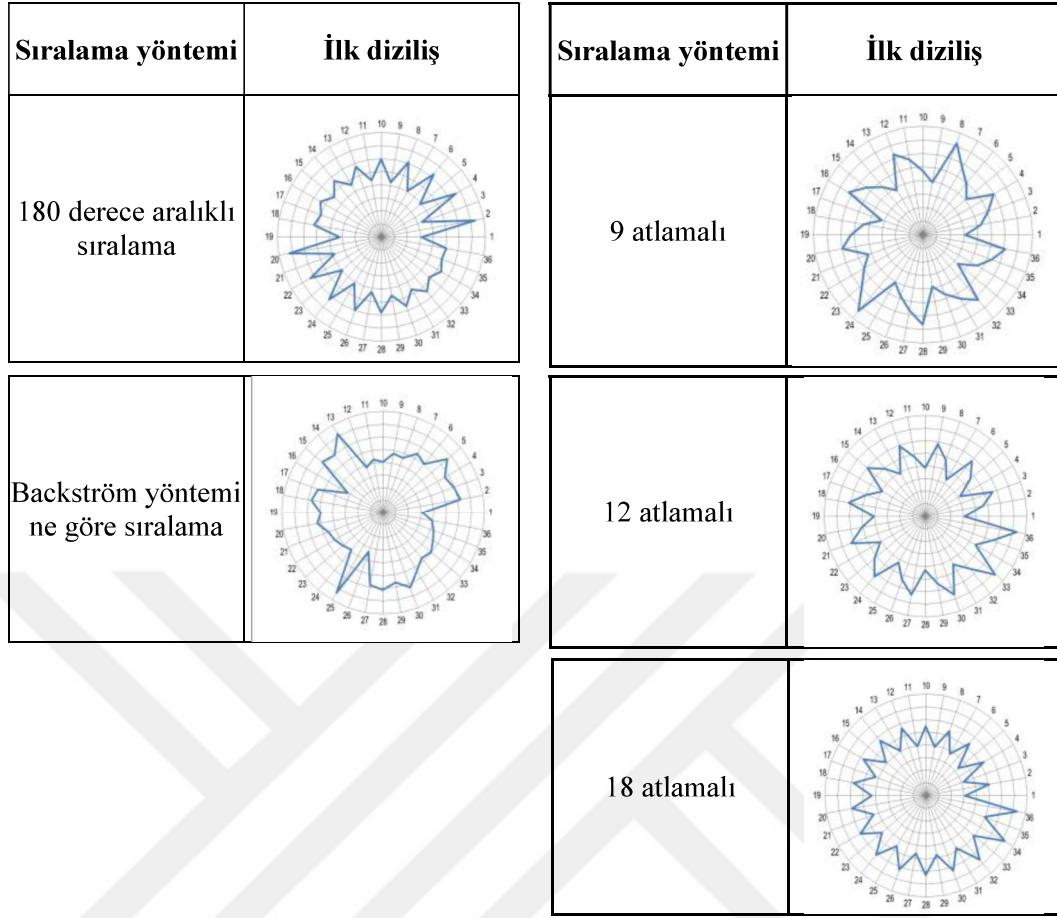
Steepest descent yönteminde bütün değişimler arasında en iyi sonucu veren değişim seçilerek bir sonraki iterasyona geçilir. Bu şekilde bir algoritma uygulandığında aynı başlangıç dizilimi ile her zaman aynı sonuç elde edilmektedir. Karınca koloni optimizasyonu algoritmasında ise her iterasyondaki yakınsamaya bağlı olarak farklı bir kanatçık çifti seçildiğinden dolayı bir sonraki dizilim farklı olmaktadır. Dolayısı ile ACO algoritması ile aynı başlangıç dizilimi ile her çalıştırmada farklı bir sonuç elde edilmektedir.

Kanatçık dizilim probleminin diğer bir yönü ise başlangıçta uygulanan kanatçık diziliminin sonuçta elde edilen çözümü etkiliyor olmasıdır. Bu nedenle Şekil 3.8’de gösterilen farklı başlangıç dizilimleri ile üç farklı algoritma kullanılarak (Exchange Heuristics – EH, Steepest Descent – SD, Karınca kolonisi optimizasyonu – ACO) çözümleme yapılmış ve sonuçları karşılaştırılmıştır. Şekildeki mavi çizginin daire merkezine olan uzaklığı kanatçık ağırlığını göstermektedir.

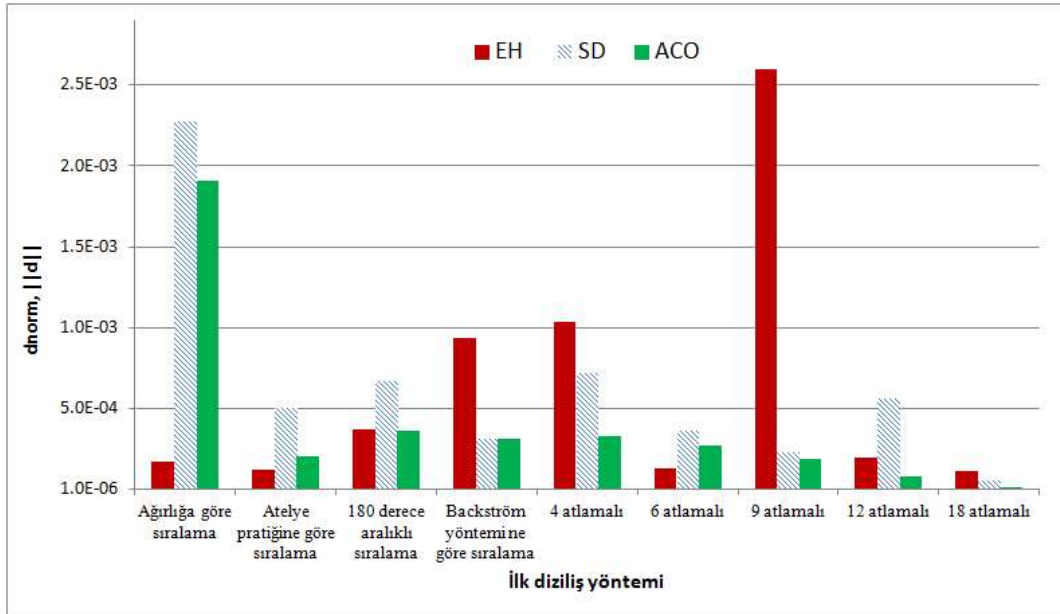
Şekil 3.9’da optimizasyon algoritmalarından elde edilen sonuçlar karşılaştırılmıştır. Buna göre Karınca Koloni Algoritması ile çok iyi sonuçlar elde edilebildiği görülmüştür.



Şekil 3.8. Kanatçıkların ilk dizilişleri



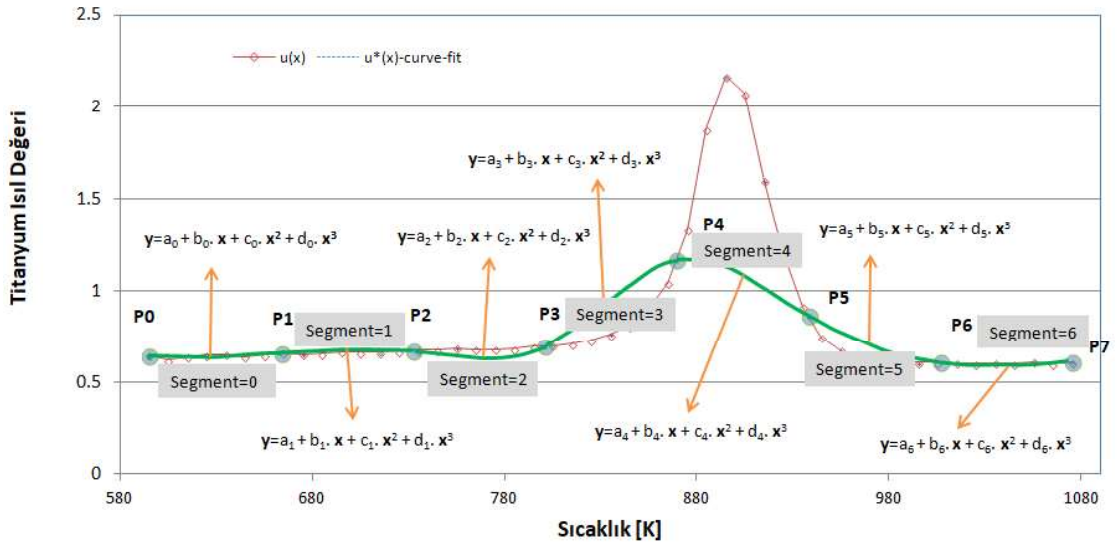
Şekil 3.8. Kanatçıların ilk dizilişleri (devam)



Şekil 3.9. Optimizasyon algoritmaları ile elde edilen sonuçların karşılaştırılması

3.3. Regresyon Yöntemi İle Eğri Modellemesi

Deneysel veri noktalarından oluşan eğrilerin matematiksel olarak modellenmesine regresyon veya eğri uydurma adı verilmektedir. En küçük kareler yöntemi ile istenilen dereceden polinom eğrisi uydurmak mümkündür. Polinomun derecesi artırılarak verilere daha iyi uyması sağlanabiliyor olsa da bu işlem istenilen hassasiyeti sağlamakta yetersiz kalabilmektedir. Daha iyi bir modelleme elde etmek için polinom derecesini arttırmak yerine spline eğrileri kullanılmaktadır. Bunlardan, doğal kübik spline eğriler, segment adı verilen birçok eğri parçasından oluşan birleşik eğrilerdir [28-33]. Her bir segment 3ncü derece bir denklem ile ifade edilmektedir. Eğri parçalarının birbirine eklendikleri yerde pozisyon sürekliliği ve eğim sürekliliği göz önüne alındığı için sanki pürüzsüz bir tek eğri varmış gibi görünmektedir (Şekil 3.10). Bölgesel olarak kontrol edilebildiklerinden istenilen hassasiyetin yakalanması çok daha kolay olmaktadır.



Şekil 3.10. Spline eğrisini oluşturan segmentlerin gösterimi

(Blair, 2021) kodundan faydalanılarak spline eğrilerin matematiksel olarak modellemesinde kullanılan bir program oluşturulmuştur [34]. Bu programda Tablo 3.3'te verilen adımlar izlenerek Tablo 3.4'teki spline katsayı matrisi bulunmaktadır.

Tablo 3.3. Spline matrisi hesaplama adımları

Adım	Açıklamalar
0	Verilerin düzenlenmesi ve sıralanması
1	h(i) hesaplanması
2	Alfa(i) hesaplanması
3	l(0), mu(0) ve Z(0) sınır şartları atanması
4	l(i), mu(i) ve Z(i) ara değerler hesaplanması

Tablo 3.3. *Spline matrisi hesaplama adımları (devam)*

5	$l(np)$, $mu(np)$ ve $Z(np)$ sınır şartları atanması
6	Spline katsayılarının hesaplanması ($y=a + b. x + c. x^2 + d. x^3$)
7	Spline çizilmesi

Tablo 3.4. *Spline katsayı matrisi*

Segment	1	x	x^2	x^3
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2
3	a3	b3	c3	d3
4	a4	b4	c4	d4
5	a5	b5	c5	d5
6	a6	b6	c6	d6

Spline eğrileri oluşturmak için düğüm noktalarına ihtiyaç vardır. Düğüm noktalarının belirlenmesi için ilk yöntemler de (De Boor & Rice, 1968) tarafından sunulmuştur [30,31].

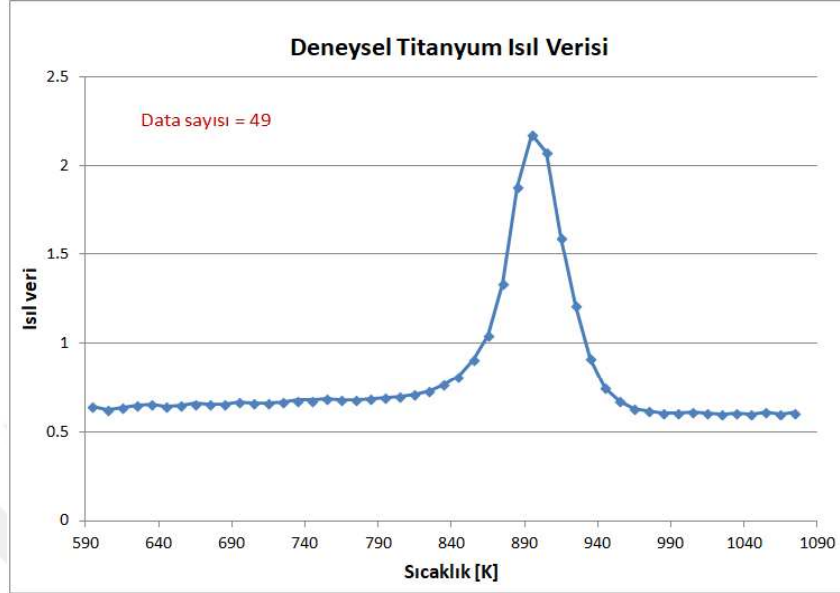
De Boor ve Rice tarafından tarif edilen yöntemler şunlardır;

- Sistematiik olmayan düğüm noktası optimizasyonu: Bu düğüm noktalarından bazılarının yerlerinin iyi olduğu ancak aradıklarının iyileştirilerek daha iyi bir sonuç elde edilebileceğini ifade etmektedir. Bütün düğüm noktalarının döngüsel olarak birkaç kez ziyaret edilip yerlerin ayarlanması gerekir.
- Sistematiik olarak düğüm noktaların eklenmesi: maksimum hatanın olduğu bölgenin bulunarak yeni düğüm noktasının buraya eklenmesi hatayı hızlı şekilde azaltacaktır (L_{∞}). Alternatif olarak bu bölgedeki başka bir yere de ekleme yapılabilir (L_2). Hata istenilen seviyeye düşünceye kadar düğüm noktası eklenerek devam edilebilir.
- Düğüm noktalarının sistematiik olarak optimizasyon ile eklenmesi: bu yöntemde bazı ana noktalar ilk başta eklenirken diğerleri eklendikte optimizasyon yapılması ve gerekli görüldükçe yenilerinin eklenmesi şeklinde ilerlemektedir.(De Boor & Rice, 1968)

Düğüm noktalarının seçilmesi ve yerlerinin optimize edilmesi ile ilgili literatürde çalışmalar bulunmaktadır. (Uyar & Ulker, 2017) tarafından yayılcı yabancı ot optimizasyon yöntemi ile B-spline eğri uydurulması [35], (Iglesias, Galvez, & Collantes, 2015) tarafından Yarasa algoritması ile bezier eğrilerinin parametrize edilmesi [36] ve (Galvez & Iglesias, 2011; Galvez & Iglesias, 2013) tarafından parçacık sürü optimizasyonu ve ateşböceği algoritması ile B-spline eğri uydurulması çalışmaları yayınlanmıştır [37,38].

Bu tez çalışmasında Karınca kolonisi optimizasyonu ve Goldsection yöntemleri kullanılarak kübik spline eğri uydurma çalışması yapılmıştır. Spline eğri uydurma ve optimizasyon algoritmasının başarısını ölçebilmek için (De Boor & Rice, 1968)

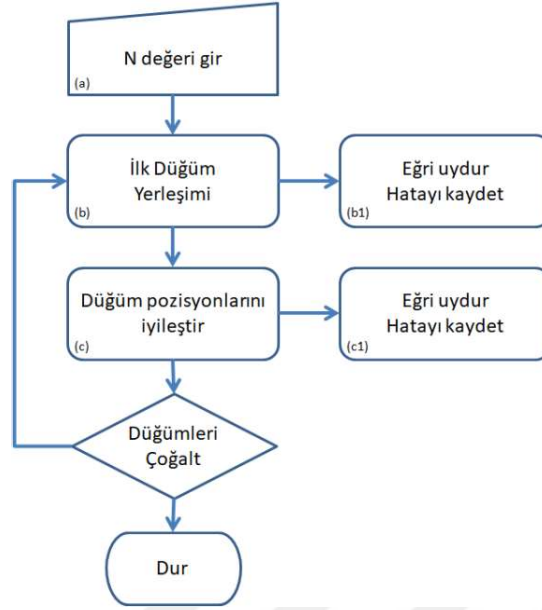
çalışmalarından [30,31] faydalanılmıştır. De Boor (1968) ilk defa eğri uydurma amacıyla Titanyum malzemesine ait ısı verisini kullanmıştır (Şekil 3.11).



Şekil 3.11. Deneysel titanyum ısı verisi

Optimizasyondaki hedef veriler ile modellenen eğri arasındaki hatanın en az olacak şekilde düğüm noktalarının yerlerinin belirlenmesidir. Düğüm sayısı arttıkça hatanın azaltılacağı açıktır ancak buradaki amaç mümkün olan en az sayıda düğüm noktası kullanarak en az hatayı veren eğriyi elde etmektir. Bunun için beş adet düğüm sayısından başlayıp onaltı adet düğüm noktasına kadar optimizasyon uygulanarak en uygun düğüm konumları hesaplanmıştır.

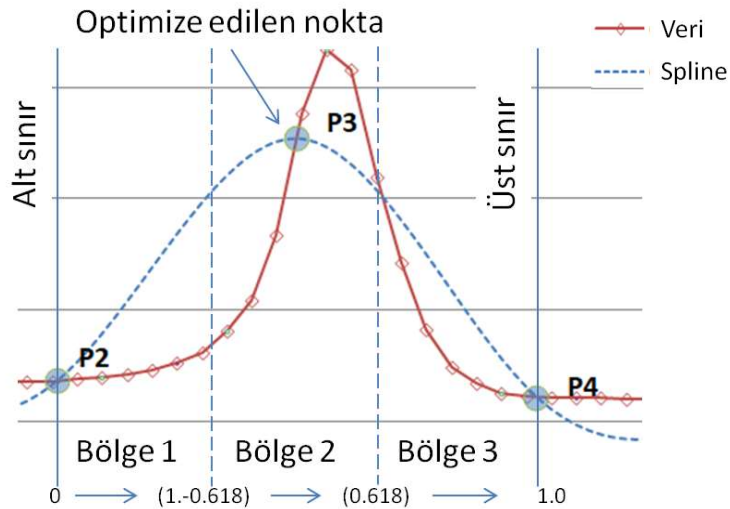
De Boors tarafından önerilen [30] “sistemik olmayan” yöntemle göre yapılan dizilişler D1 ile (Dierckx, 1996) tarafından önerilen “sistemik” yöntemle [28] göre yapılan dizilişler D2 ile adlandırılmıştır. Optimizasyonda uygulanan adımlar şu şekildedir; seçilen sistematige göre (D1 veya D2) N adet düğüm noktasının yerleşimi yapılır. Bu düğüm noktaları kullanılarak spline katsayıları hesaplanır. Gerçek veriler ile spline eğrisi üzerindeki değerlerin farklarının karelerinin toplanması ile toplam hata hesaplanır. Daha sonra seçilen optimizasyon yöntemi (ACO veya Goldsection) uygulanarak düğüm noktalarının yerleri iyileştirilir ve toplam hata yeniden hesaplanır. Düğüm noktalarının yer değiştirmesi ile hatanın daha fazla düşürülemediği noktada iterasyonlar sonlandırılarak bir sonraki düğüm sayısına geçilir. Bu adımları gösteren akış diyagramı Şekil 3.12’de verilmiştir.



Şekil 3.12. Spline eğri uydurma adımları

3.3.1. Goldsection optimizasyonu yöntemi ile eğri uydurma

Goldsection optimizasyon yöntemi [5], “O1” olarak adlandırılmıştır. D1 veya D2 ye göre yerleştirilmiş olan düğüm noktaları içinden 1 ile N-1 arasında olanlar önce soldan sağa doğru ve sonra sağdan sola ziyaret edilerek hata azaltılmaya çalışılır. Örneğin Şekil 3.13’te gösterilen spline bölgesinde P2 ve P4 noktaları goldsection yönteminde alt ve üst sınırlar olarak seçilmiştir. Aradaki alan iç bölgeye ayrılır ve bölge sınırındaki noktalar için spline hatası hesaplanır bulunan değere göre bölgelerden birisi veya ikisi elenerek yeni alt ve üst sınır seçilir. Alt ve üst sınırlar hızlı şekilde daralarak P3 noktasının en uygun yeri bulunur.



Şekil 3.13. Goldsection optimizasyon yöntemi

“O1” Optimizasyon problemi Eşitlik (3.7) ve (3.8) ile tarif edilmiştir.

$$\text{Minimize } \|mse\| = \frac{1}{m} \cdot \sum_{i=0}^m (f(x_i) - s(x_i))^2 \quad (3.7)$$

$y_j = s(x_j)$ spline fonksiyonu ve $P_j = \{x_j, y_j, j=1, N\}$ düğüm noktaları vektörü

$$\text{ve } x_j \in (x_{j-1}, x_{j+1}) \quad (3.8)$$

X ve Y koordinatlarından oluşan m adet veri bulunmaktadır.

$$\mathbf{X}_i \in \mathbb{R}^2, S = \{\mathbf{X}_i: i=0 \text{ to } m\} \quad (3.9)$$

$$\mathbf{X}_i = \{x_i, y_i\}$$

N adet spline düğüm noktası seçilir. K spline düğüm vektörü diye adlandırılır.

$$\begin{aligned} P_j &= \{x_j, y_j\}, K = \{P_j: j=0 \text{ to } N\} \\ P_0 &< P_1 < P_2 < \dots < P_{N-1} < P_N, \end{aligned} \quad (3.10)$$

3.3.2. Karınca kolonisi optimizasyonu ile eğri uydurma

Karınca kolonisi optimizasyonu yöntemi “O2” olarak adlandırılmıştır. Bu yön-temde D1 veya D2 ye göre yerleştirilmiş olan düğüm noktalarına spline eğrisi uydurularak, en yüksek sapmaya sahip segment bulunur. Bu segment ile sağındaki ve solundaki segmentleri tanımlayan düğüm noktaları seçilir ve bunların yerleri iyileştirilir (Şekil 3.14). Daha sonra spline üzerinde başka bir segment seçilir. Bu işlem birkaç tekrar edilerek spline hata miktarı azaltılır.

“O2” Optimizasyon problemi aşağıdaki şekilde tarif edilmiştir.

$$\text{Minimize } \|mse\| = \frac{1}{m} \cdot \sum_{i=0}^m (f(x_i) - s(x_i))^2 \quad (3.11)$$

$$y_j = s(x_j), P_j = \{x_j, y_j, j=k-2, k+1\}, \quad (3.12)$$

$$\text{ve } x_j \in (x_{j-1}, x_{j+1}) \text{ } k=\text{en kötü segment numarası}$$

X ve Y koordinatlarından oluşan m adet veri bulunmaktadır.

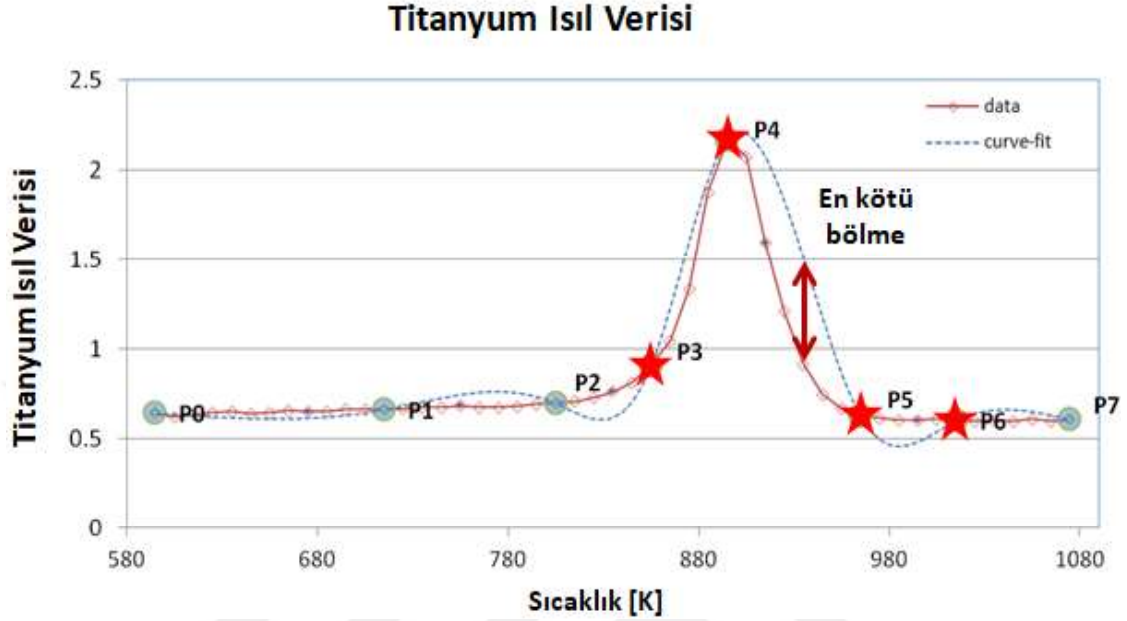
$$\mathbf{X}_i \in \mathbb{R}^2, S = \{\mathbf{X}_i: i=0 \text{ dan } m\text{'ye}\} \quad (3.13)$$

$$\mathbf{X}_i = \{x_i, y_i\}$$

N adet spline düğüm noktası seçilir. K spline düğüm vektörü diye adlandırılır.

$$P_j = \{x_j, y_j\}, K = \{P_j: j=0 \text{ dan } N' \text{ ye}\} \quad (3.14)$$

$$P_0 < P_1 < P_2 < \dots < P_{N-1} < P_N,$$

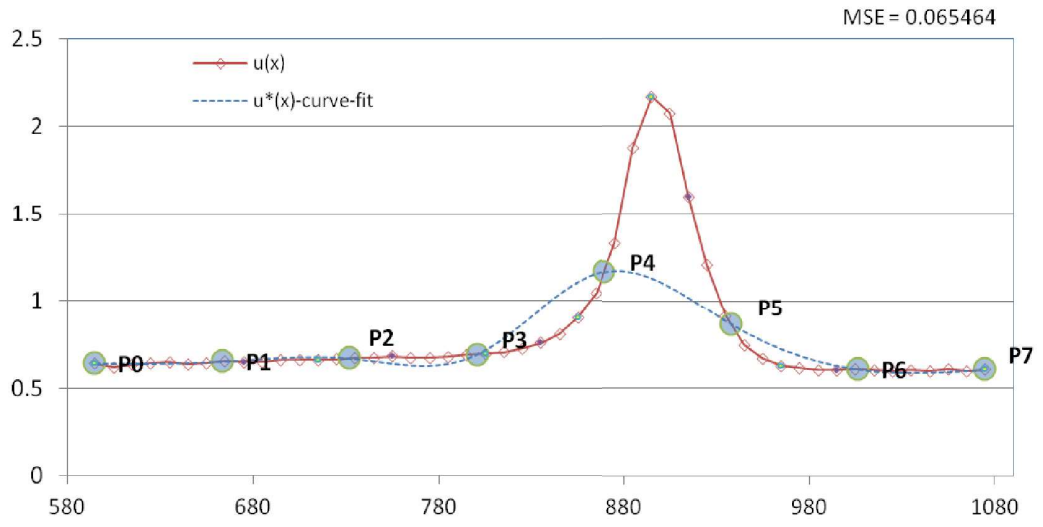


Şekil 3.14. Karınca kolonisi optimizasyon yönteminin uygulanması

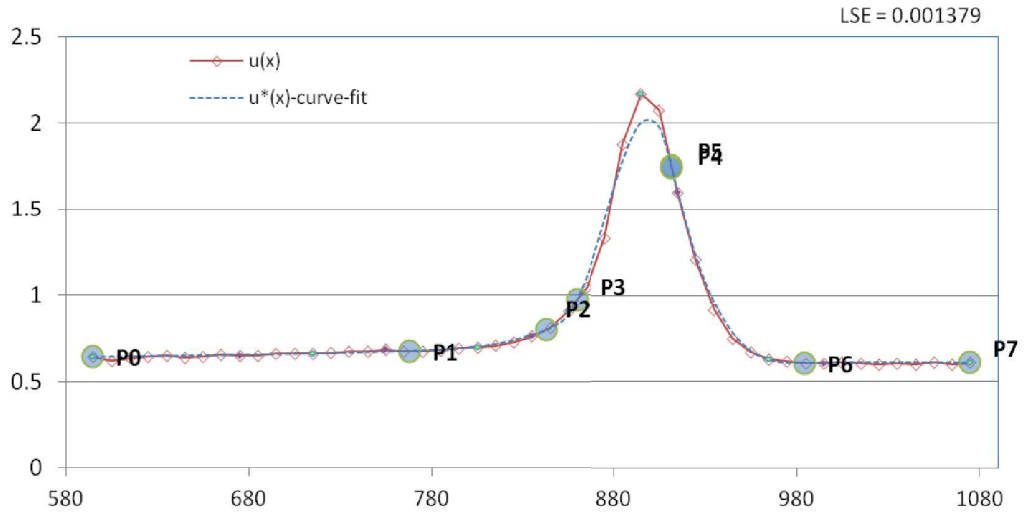
3.3.3. Goldsection ve ACO sonuçlarının karşılaştırılması

Örnek olarak, NS=7 segment için; D1 dizilişi ardından O1 ve O2 yöntemleri ile iyileştirilen spline eğrileri Şekil 3.17, Şekil 3.16 ve Şekil 3.17’de, D2 dizilişi ardından O1 ve O2 yöntemleri ile iyileştirilen spline eğrileri Şekil 3.18, Şekil 3.19 ve Şekil 3.20’de gösterilmiştir. Şekiller, kolaylık olması açısından, ilk diziliş ve ardından uygulanan optimizasyon yöntemi verilerek adlandırılmıştır. Örneğin “D1-O1” ifadesinde, D1 dizilişi ve O1 optimizasyon yöntemini ifade etmektedir.

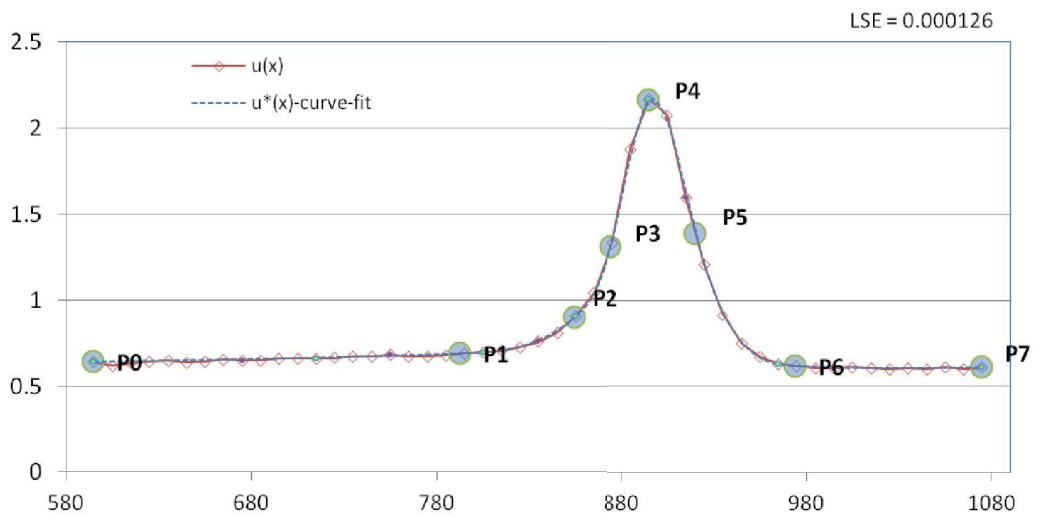
Düğüm noktası sayısı 5 ten 16 kadar değiştirilerek (NS=4 ten 15 e kadar), D1 ve D2 ilk diziliş yönteminden gelen “MSE” ve “R” değerleri ve diziliş yöntemi ardından uygulanan O1 ve O2 optimizasyon yöntemleri ile ne kadar iyileştirilebildiği Tablo 3.5 ve Tablo 3.6’da özetlenmiştir.



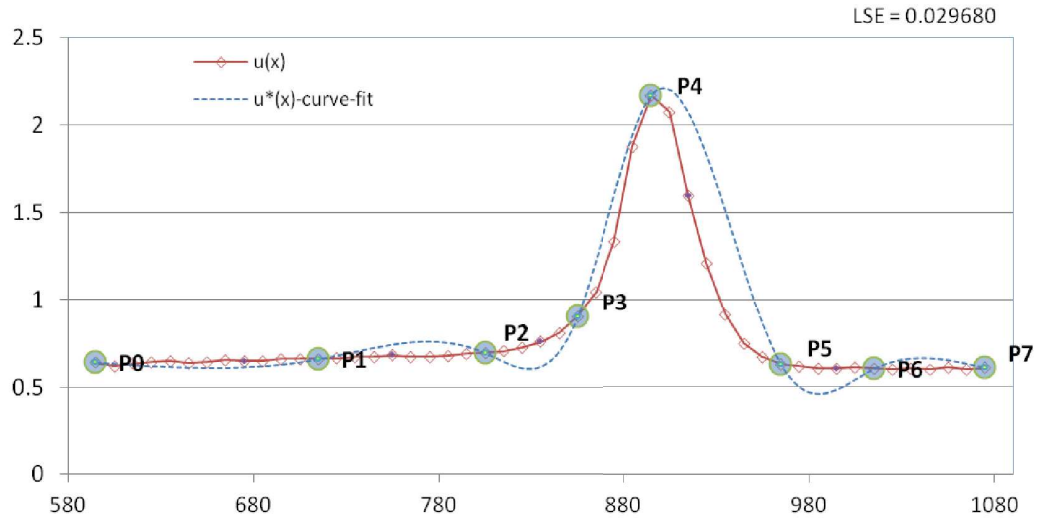
Şekil 3.15. D1 ilk diziliş (NS=7)



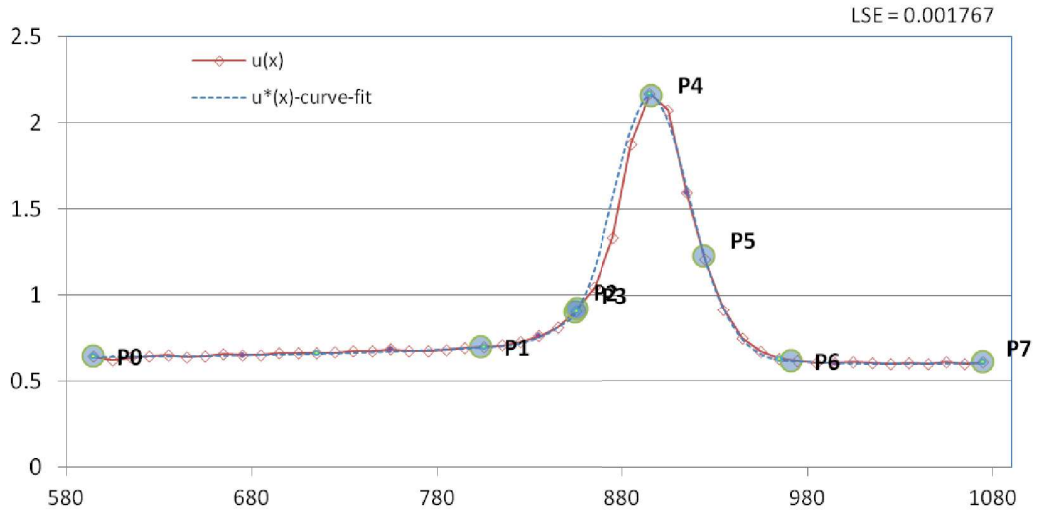
Şekil 3.16. D1-O1 optimizasyon sonucu (NS=7)



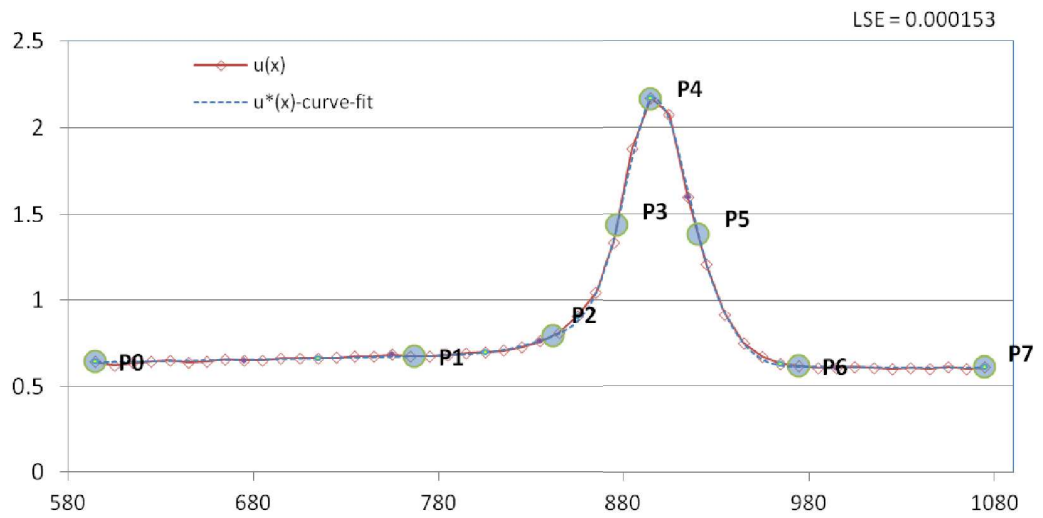
Şekil 3.17. D1-O2 optimizasyon sonucu (NS=7)



Şekil 3.18. D2 ilk dizilişi ($NS=7$)



Şekil 3.19. D2-O1 optimizasyon sonucu ($NS=7$)



Şekil 3.20. D2-O2 optimizasyon sonucu ($NS=7$)

Tablo 3.5. MSE özet tablosu

NS	D1	D2	D1-O1	D2-O1	D1-O2	D2-O2
4	0,13806	0,26583	0,05861	0,05861	0,04712	0,05196
5	0,06725	0,10685	0,02212	0,03615	0,01160	0,00950
6	0,04765	0,07314	0,03737	0,00404	0,01186	0,00818
7	0,06546	0,02968	0,00138	0,00177	0,00013	0,00015
8	0,02893	0,00727	0,00093	0,00183	0,00018	0,00014
9	0,02782	0,00141	0,00004	0,00015	0,00011	0,00022
10	0,02024	0,00017	0,00022	0,00013	0,00012	0,00015
11	0,00532	0,00044	0,00004	0,00012	0,00011	0,00014
12	0,01608	0,00023	0,00003	0,00013	0,00013	0,00016
13	0,00547	0,00016	0,00036	0,00011	0,00012	0,00010
14	0,00284	0,00015	0,00023	0,00002	0,00011	0,00003
15	0,00646	0,00013	0,00034	0,00002	0,00014	0,00002

Tablo 3.6. R özet tablosu

NS	D1	D2	D1-O1	D2-O1	D1-O2	D2-O2
4	0,48458	0,78146	0,77380	0,77363	0,82606	0,82575
5	0,80079	0,84735	0,91635	0,86309	0,95963	0,96532
6	0,81233	0,88775	0,86509	0,99059	0,96261	0,97201
7	0,80219	0,94261	0,99623	0,99450	0,99961	0,99950
8	0,94633	0,97957	0,99729	0,99510	0,99935	0,99954
9	0,90137	0,99571	0,99984	0,99949	0,99959	0,99940
10	0,92418	0,99958	0,99924	0,99959	0,99956	0,99965
11	0,98485	0,99879	0,99984	0,99961	0,99963	0,99966
12	0,95064	0,99932	0,99988	0,99960	0,99952	0,99962
13	0,98013	0,99962	0,99902	0,99964	0,99955	0,99967
14	0,98968	0,99955	0,99935	0,99992	0,99961	0,99990
15	0,97731	0,99954	0,99881	0,99994	0,99952	0,99993

3.4. Parçacık Sürüşü Optimizasyonu Yöntem Doğrulaması

3.4.1. Küresel konveks fonksiyon

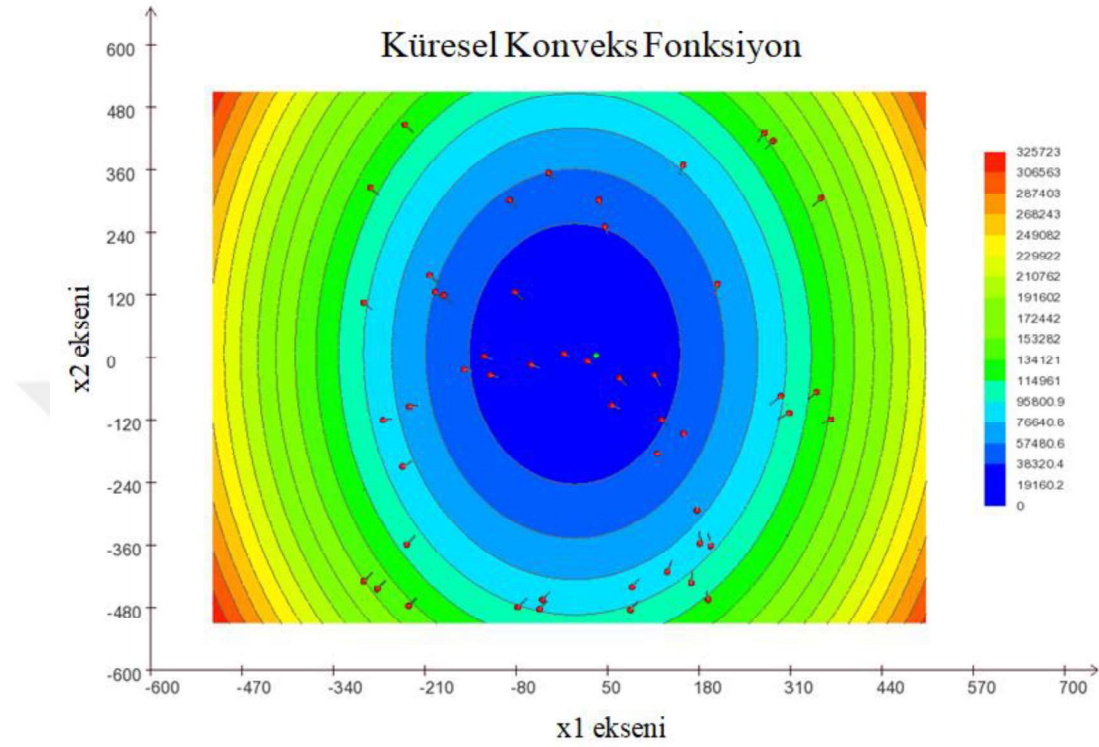
İlk olarak Eşitlik (3.15)'teki basit konveks küresel fonksiyon seçilmiştir. Fonksiyonun bu aralıkta aldığı minimum değer sıfırdır ve $P[0, 0]$ noktasındadır. Programda seçilen parametrelere bağlı olarak istenilen hassasiyet ile sonuca yakınsayabilmektedir. PSO ile elde edilen sonuçlar Tablo 3.7'de verilmiştir.

$$\text{minimize } f(x) = \left(\frac{(x_1 \cdot x_1)}{110} + \frac{(x_2 \cdot x_2)}{110} \right) \cdot 100$$

$$x_i \in [-512, 512], i = 1, 2 \quad (3.15)$$

Tablo 3.7. Yakınsamış sonuçlar

x1	x2	F(x1,x2)
7,68225E-06	9,82609E-05	3,27313E-09



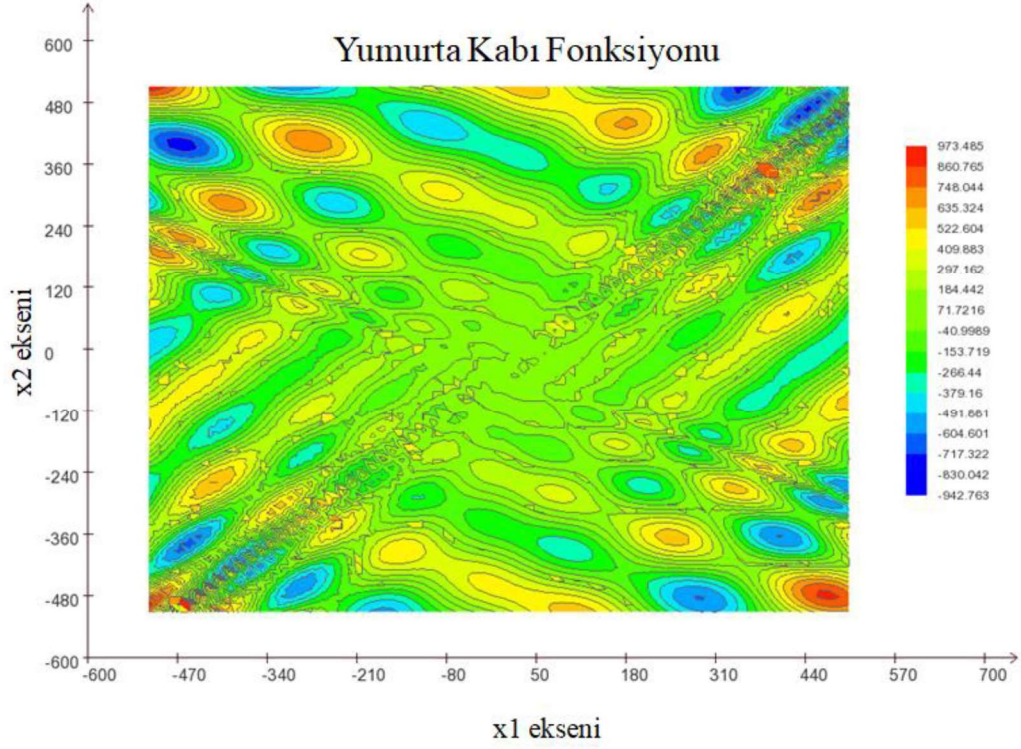
Şekil 3.21. Konveks fonksiyon grafiği

3.4.2. Yumurta Kabı (Egg-holder) Fonksiyonu

Yumurta kabı fonksiyonu iki boyutlu bir fonksiyon olup Eşitlik (3.16) de verilmiştir [4]. Fonksiyonun bu aralıkta aldığı minimum değer sıfırdır ve P[512,0; 404,2319] noktasındadır. Fonksiyonda birçok yerel minimum ve maksimum noktaları olduğundan birçok optimizasyon algoritması için zor bir problem olarak değerlendirilmektedir.

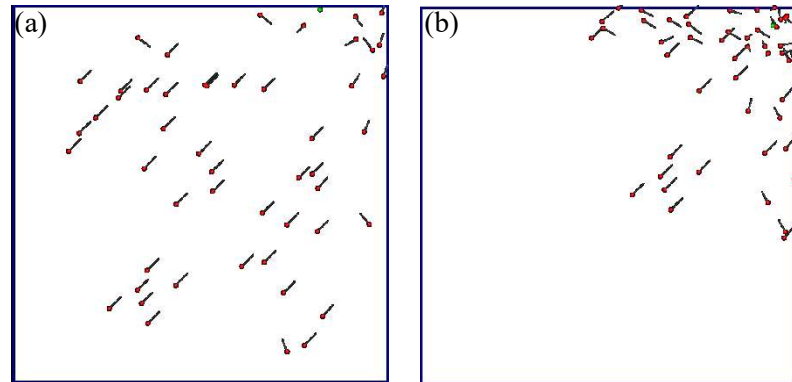
$$f(x) = -(x_2 + 47) \cdot \sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \cdot \sin(\sqrt{|x_1 - (x_2 + 47)|}) \quad (3.16)$$

$$x_i \in [-512, 512], i = 1, 2$$

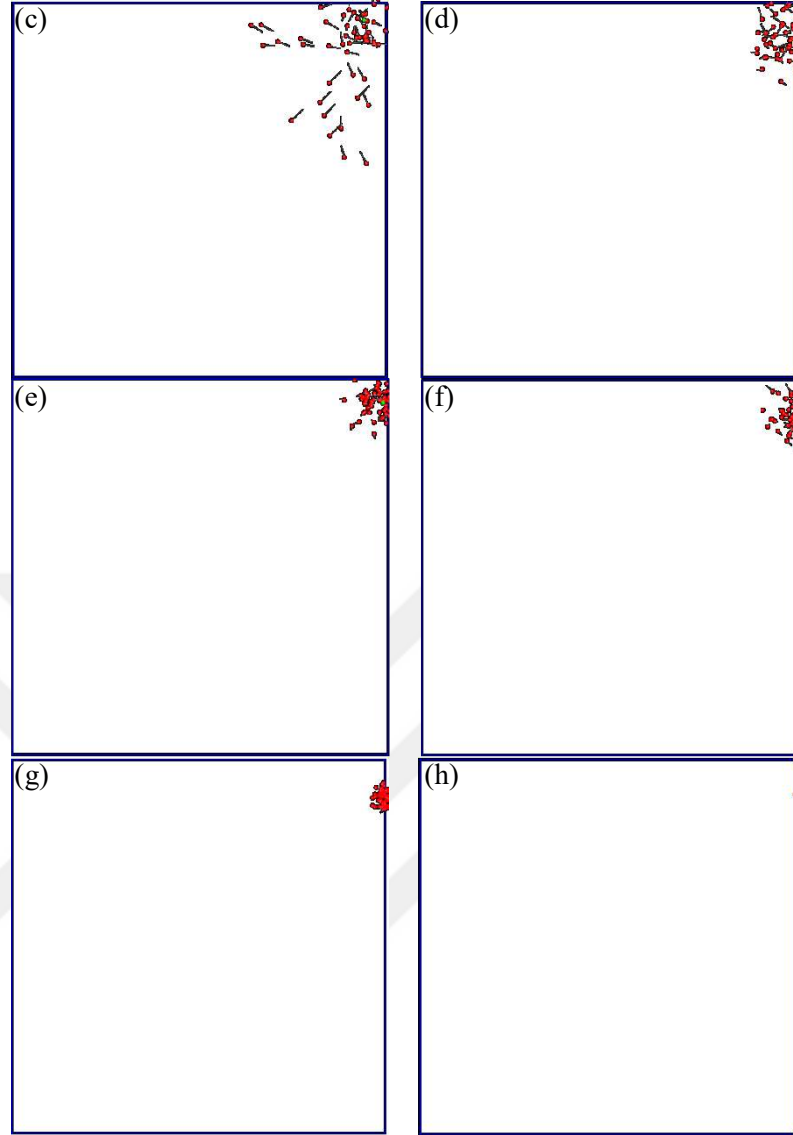


Şekil 3.22. Yumurta kabı fonksiyonu renklendirilmiş grafiği

Şekil 3.23’da parçacıkların ilk konumları ve ara iterasyonlardaki hareketleri gösterilmiştir. Sürünün global hedefi tüm parçacıklar için daha ağırlıklı olduğundan diğer bölgeler iyice araştırılmadan hızlı bir şekilde bir noktaya yakınsamaktadır. Bu nedenle de bölgesel minimum değere takılma ihtimali vardır. Yakınsama sağlanabilmesi için optimizasyon parametrelerinin ayarlanması gerekmektedir. Uygulanan algoritmanın adımları Tablo 3.8’de verilmiştir.



Şekil 3.23. Parçacıkların hareketleri ve yakınsamasını gösteren ekran görüntüleri



Şekil 3.23. Parçacıkların hareketleri ve yakınsamasını gösteren ekran görüntüleri (devam)

Tablo 3.8. Uygulanan PSO algoritması

- Parametrelerin sınırları ve fonksiyon tanımlanır.
- Sürü büyüklüğü belirlenir.
- Tüm parçacıkların ilk konumları ve ilk hızları rastgele olarak belirlenir.
- Tüm parçacıkların pozisyon vektörleri için fonksiyon değerlendirme yapılır.
- Parçacık en iyi ve sürü en iyi pozisyonlar belirlenir.
- Sürü için en iyi pozisyon tüm parçacıklar için yeni hedef olarak verilir.
- Parçacıkların birbirine sosyal mesafesi belirli bir değerden düşük olup olmadığı kontrol edilir.
- Parçacık çarpışmaları ve sınır aşmaları olup olmadığı kontrol edilir.
- Yeni hızlar belirlenir.
- Belirli bir iterasyon sayısında iyileşme olmadığı gözlenirse;
 - Sosyal mesafe yarıya indirilir.

Tablo 3.8. Uygulanan PSO algoritması (devam)

○ Sosyal mesafe yarıya indirilir. ○ Son iki bulunan noktayı birleştiren doğrultuda konik bir saçılım yapılır. ○ Böylece bu bölgede bulunan yerel minimum değerine yakınması sağlanır.
• İterasyon sayısı sınır aştığında veya yakınsama sağlandığında program durdurulur.

Genel olarak, uygulanan PSO algoritması, konveks fonksiyonda makul seviyede başarılı olurken birden fazla yerel minimuma sahip yumurta kabı fonksiyonunda bazen başarılı olmuştur.

3.5. ACOPSO Doğrulama Çalışmaları

Tez sırasında geliştirilen ve nihaî olarak elde edilen hibrit ACOPSO kodu ile aşağıdaki test fonksiyonları çözülmüştür ve bulunan sonuçlar literatür değerleri ile karşılaştırılmıştır.

3.5.1. Yumurta kabı fonksiyonu

Eşitlik (3.16)'de verilen fonksiyon ACOPSO ile tekrar çözdürülmüştür. Aynı zamanda literatürde bulunan sonuçlar ile karşılaştırılmıştır.

Tablo 3.9. Sonuçlar ve karşılaştırma tablosu

Parametre	Ref [4]	RPSO [40]	ACOPSO
x_1	512,0	512,0	512,0
x_2	404,2319	404,2319	404,230498
$f_{\min}$	-959,640	-959,640663	-959,640118

3.5.2. Rosenbrock-Watson regresyon problemi

(Mishra, 2020) birçok fonksiyonunun test edilebildiği bir Fortran kodu paylaşmıştır [39]. Bu kod içerisinde Repulsive Swarm Particle Optimization (RSPO) yöntemi kullanılmaktadır. (Mishra, 2020) başka bir çalışmasında [40] optimizasyon test problemleri ve elde ettiği sonuçları paylaşmıştır. Rosenbrock-Watson Regresyon verileri Tablo 3.10'da verilmiştir. Tanımlanan “Rate” çıktısını en iyi şekilde tanımlayan beta $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ değişkenleri araştırılmıştır.

$$rate = \frac{\beta_1 x_2 - \frac{x_3}{\beta_5}}{1 + \beta_2 x_1 + \beta_3 x_2 + \beta_4 x_3} \quad (3.17)$$

Tablo 3.10. *Rosenbrock-Watson veri tablosu*

x1	x2	x3	Rate
470,0	300,0	10,0	8,55
285,0	80,0	10,0	3,79
470,0	300,0	120,0	4,82
470,0	80,0	120,0	0,02
470,0	80,0	10,0	2,75
100,0	190,0	10,0	14,39
100,0	80,0	65,0	2,54
470,0	190,0	65,0	4,35
100,0	300,0	54,0	13,00
100,0	300,0	120,0	8,50
100,0	80,0	120,0	0,05
285,0	300,0	10,0	11,32
285,0	190,0	120,0	3,13

Tablo 3.11. *Rosenbrock-Watson regresyon problemi karşılaştırma tablosu*

	Rosenbrock-Quasi-Newton	RPSO [39]	ACOPSO
β_1	1,253031	1,557520	1,2540640
β_2	0,062798	0,078101	0,0628496
β_3	0,040063	0,050867	0,0401024
β_4	0,112453	0,138796	0,1125292
β_5	1,190943	0,955739	1,1899800
S2	0,298901	0,301934	0,298901

3.5.3. Bukin fonksiyonu no “f6”

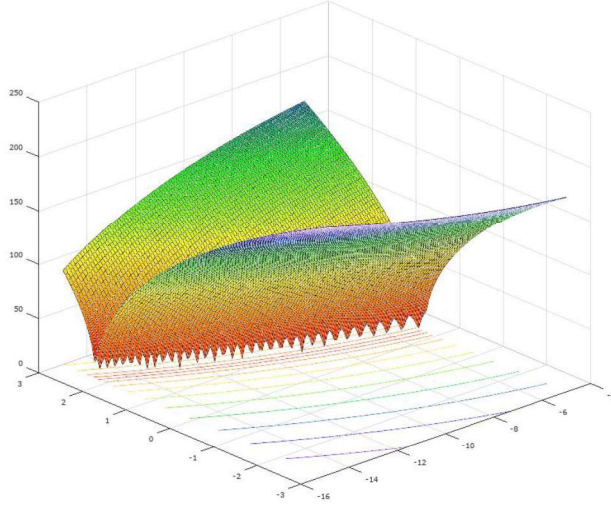
Bukin f6 [12] birçok yerel minimum değerine sahip oldukça zor bir optimizasyon problemidir. Fonksiyon denklemi Eşitlik (3.18) de, grafiksel olarak gösterimi Şekil 3.24’te verilmiştir. Tam çözümü ve ACOPSO ile hesaplanan değerler Tablo 3.12’de verilmiştir.

$$\text{Minimize } f(x) = 100 \cdot \sqrt{|x_2 - 0,01 \cdot x_1^2|} + 0,01 \cdot |x_1 + 10,0|$$

$$x_1 \in [-15, -5], x_2 \in [-3, 3]$$
(3.18)

Tablo 3.12. *Global minimum ve ACOPSO ile bulunan en iyi değer*

Parametre	Global minimum	ACOPSO ile bulunan
x1	-10,0	-10,04027292
x2	1,0	1,008070798
f(x)	0,0	0,007528366



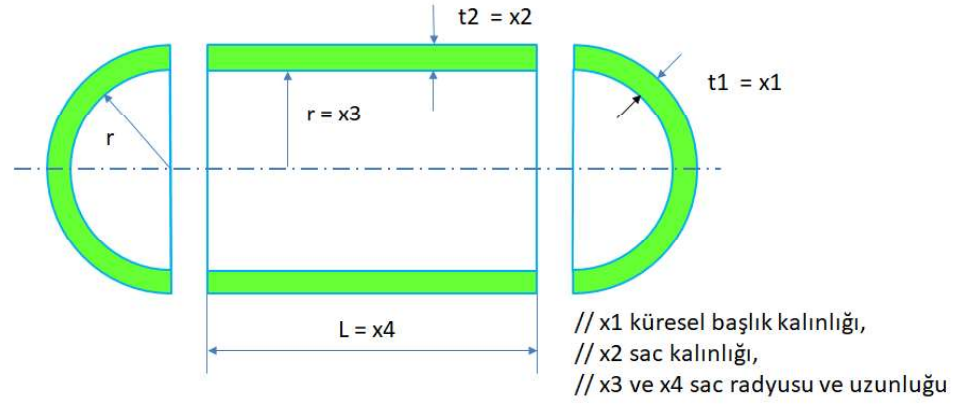
Şekil 3.24. Bukin fonksiyonu no “f6”

3.5.4. Basınçlı kap tasarım problemi

Basınçlı kap tasarım problemi; yeterli yapısal dayanıklılığa ve verilen asgarî bir hacme sahip, basınçlı bir kabın en düşük ağırlıkla imal edilebilmesi olarak tarif edilmektedir. Üzerinde birçok çalışma yapılmıştır. Önemli çalışmalardan birisi, (Ke, Zhang, Li, & Du, 2016) tarafından yapılan meyve sineği optimizasyon Algoritması (FOA) dır [41]. Diğer önemli bir çalışma (Yang, Huyck, Karamanoglu, & Khan, 2013) tarafından yapılan detaylı bir matematiksel bir çözümdür ve buna göre global optimum ağırlığın 6059,714335048436 olduğunu belirtmişlerdir [42]. Aynı çalışmada Lagrange Multiplier (LM) yöntemi ile hesapladıkları ağırlığın doğru olduğunu göstermişlerdir. Basınçlı kap probleminde hedef fonksiyonu $f(x)$ sistemin toplam ağırlığını veren bir fonksiyondur. Problemin tanımı şu şekildedir;

Tablo 3.13. Optimizasyon tanımlama denklemleri

Hedef fonksiyonu	:	minimize $f(x) = 0,6224 \cdot t_1 r L + 1,7781 \cdot t_2 r^2 + 3,1661 \cdot t_1^2 L + 19,84 \cdot t_1^2 r$
Girdi vektörü	:	$X = [x_1, x_2, x_3, x_4]^T$, $x_1 = t_1$, $x_2 = t_2$, $x_3 = r$, $x_4 = L$
Parametre aralıkları	:	$x_1 = [0,0625; 6,1875]$, $x_2 = [0,0625; 6,1875]$, $x_3 = [25; 200]$, $x_4 = [25; 200]$
Kısıt fonksiyonları	:	$g_1(x) = -x_1 + 0,0193 \cdot r \leq 0$ $g_2(x) = -x_2 + 0,00954 \cdot r \leq 0$ $g_3(x) = -\pi x_3^2 L - \frac{4}{3} \pi x_3^3 + 1296000 \leq 0$ $g_4(x) = x_4 - 200 \leq 0$



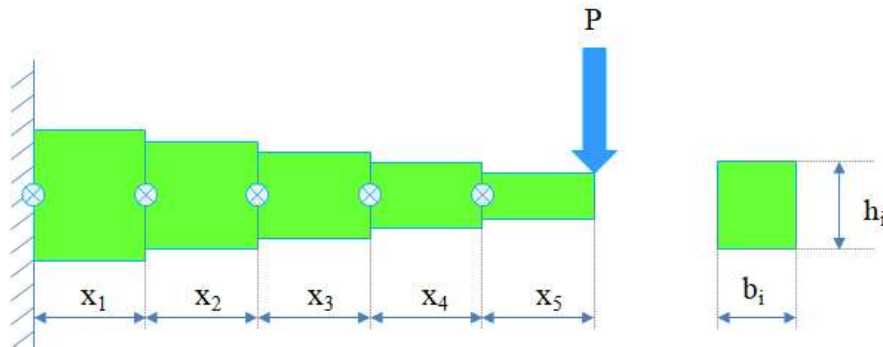
Şekil 3.25. Basınçlı kap şematik gösterimi

Tablo 3.14. Basınçlı kap optimizasyonu sonuçları karşılaştırması

Parametre/ Fonksiyon	FOA [41]	LM [42]	ACOPSO
x1	0,780956	0,8125	0,778230
x2	0,386318	0,4375	0,384681
x3	40,43396	42,09845	40,322180
x4	198,5049	176,6366	200,000000
g1	-0,000580649	0,0	-0,00001217
g2	-0,00057806	-0,035880829	-0,00000760
g3	-464,5613353	3,95812E-09	-182,09796811
g4	-41,495111	-63,36340416	-40,00000000
f(x)	5896,94890	6059,714335	5886,292113

3.5.5. Ankastre kiriş tasarım problemi

Ankastre kiriş problemi ilk olarak ayrıışık kirişler problemi olarak sunulmuştur [43]. (Gandomi, Yang, & Alavi, 2011) Ateşböceği optimizasyonu algoritmasını uygulayarak bu problemi çözmüşlerdir [44]. (Yang, Huyek, Karamanoglu, & Khan, 2013) bunu sürekli bir optimizasyon problemine dönüştürerek çözmüşlerdir [42].



Şekil 3.26. Ankastre kiriş şematik gösterimi

Hedef fonksiyonu, girdi vektörü, parametre aralıkları ve kısıt fonksiyonu Tablo 3.15'te verilmiştir. Kirişin toplam ağırlığı hedef fonksiyonu olarak verilmiştir.

Tablo 3.15. Optimizasyon tanımlama denklemleri

Hedef fonksiyonu	:	$minimize f(x) = 0,6224 \cdot (x_1 + x_2 + x_3 + x_4 + x_5)$
Girdi vektörü	:	$X = [x_1, x_2, x_3, x_4, x_5]^T$,
Parametre aralıkları	:	$x_1 = [0,01;100]$, $x_2 = [0,01;100]$, $x_3 = [0,01;100]$, $x_4 = [0,01;100]$, $x_5 = [0,01;100]$
Kısıt fonksiyonu	:	$g_1(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$

Elde edilen sonuçlar ile literatürde verilen değerler Tablo 3.16'da karşılaştırılmıştır. Buradan görüldüğü üzere ACOPSO ile literatürdeki sonuçlara yakın ve hatta daha iyi sonuçlar elde edilmiştir. Böylece uçak itki sistemleri optimizasyonu ve tasarımında kullanılmasının uygun ve yeterli olacağı gösterilmiştir.

Tablo 3.16. Ankastre kiriş optimizasyon sonuçları

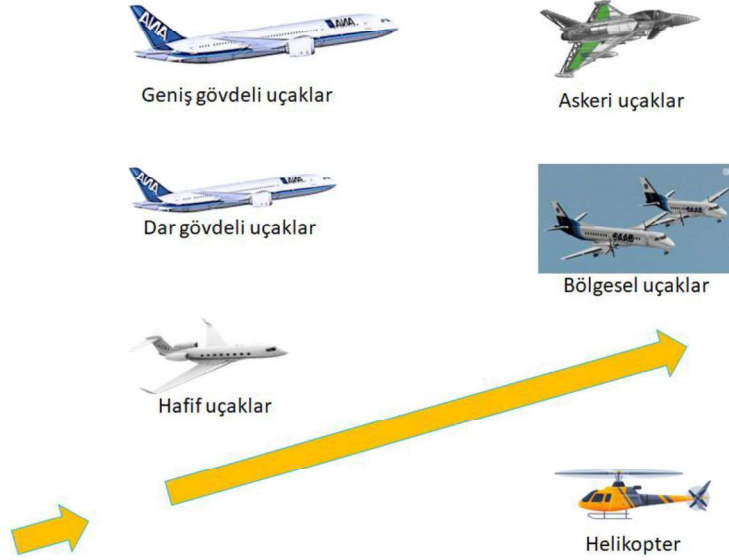
Parametre/ Fonksiyon	LM [42]	ACOPSO
x1	6,0089	6,023783896
x2	5,3049	5,306014041
x3	4,5023	4,492747018
x4	3,5077	3,501325232
x5	2,1504	2,149821887
g1	-0,0000644864	-0,0000000014
f(x)	1,339956367	1,339958385

4. UÇAK MOTORU İTKİ SİSTEMLERİ

Uçak motoru genellikle kullanılacağı uçağın görev profiline ve amacına göre tasarlanmaktadır. Bu nedenle itki sistemlerinin optimizasyonu da kullanılacak plâtfoma göre yapılmalıdır. Günümüzde kullanılan uçaklara ve üzerlerindeki motor tiplerine bakılarak, turbofan motorların yaygın şekilde kullanıldığı görülmektedir.

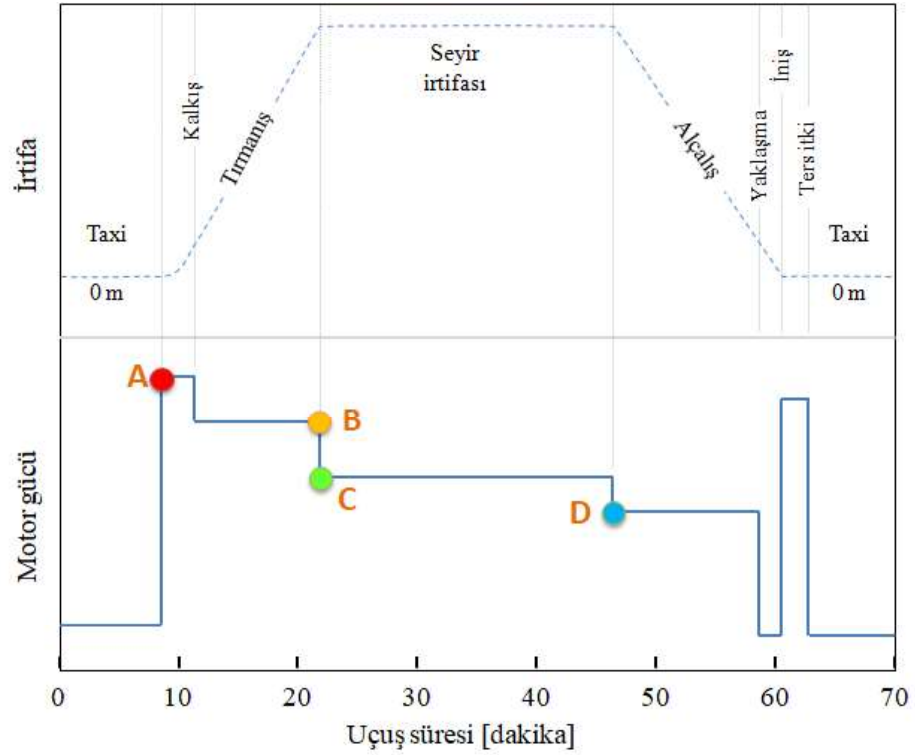
Tablo 4.1. Uçaklar ve kullanılan motor tipleri

Plâtfom tipi	Motor tipi
Geniş gövdeli uçaklar	Yüksek by-pass Turbofan
Dar gövdeli uçaklar	Orta/Yüksek by-pass Turbofan
Bölgesel uçaklar	Orta bypass Turbofan, Turboprop
Hafif/küçük uçaklar	Orta bypass Turbofan, Turboprop
Askeri savaş uçakları	Art yanmalı düşük by-pass turbofan
Askeri nakliye uçakları	Orta/Yüksek by-pass Turbofan Turboprop
Helikopterler	Turboşaft



Şekil 4.1. Havacılıkta yaygın olarak kullanılan uçak tipleri

Geniş ve dar gövdeli uzun menzilli yolcu uçaklarında yüksek by-pass oranlı, yüksek basınç oranlı ve yüksek türbin giriş sıcaklığına sahip motorlar tercih edilmektedir. Tipik olarak sivil bir yolcu uçağının uçuş profili ve motorun güç seviyesinin görev boyunca değişimi Şekil 4.2’de gösterilmiştir.



Şekil 4.2. Tipik bir ticarî turbofan uçuş profili [45]

Tablo 4.2. Motor seçiminde/tasarımında dikkate alınan A, B, C ve D noktaları

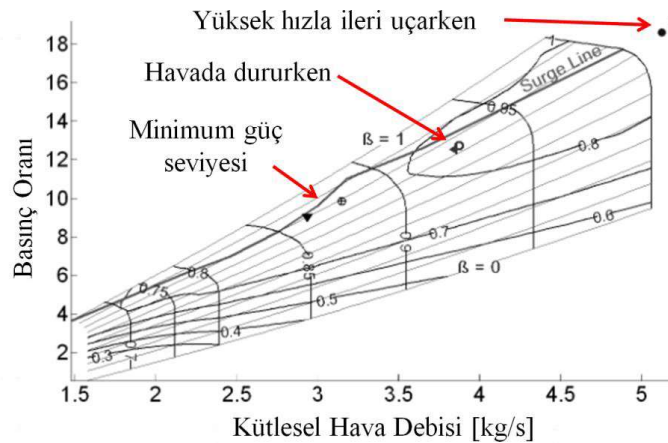
Nokta	Motor gücü	Açıklama
A	Maksimum	Uçağın en kısa mesafeden kalkış yapabilmesi için motor itkisinin maksimum seviyede olması gereklidir
B	İrtifada maksimum	İrtifa artışı nedeniyle motor itkisi azalır. Tırmanışı tamamlayabilmek için motorlar bu irtifada gerekli itkiyi üretebilmelidir.
C	Normal uçuş	Seyahatin en uzun kısmında yakıt sarfiyatının da en düşük olması istenir. İstenilen uçuş hızında gerekli itki sağlanmalıdır
D	Uçuştaki en düşük	Uçağın havada yavaşlayabilmesi için gerekli minimum motor gücüdür. Motorun bu şartlarda çalışmasını zorlaştıran etkenler bulunabilir.

Bu noktaların uçağın kullanılacağı tüm hava meydanları ve ortam sıcaklıkları için gerekli itki seviyelerinin sağlanabiliyor olması gerekmektedir. “A” noktası genellikle büyük yolcu uçakların kalkış yapabilecek pistlerin uzunluklarının yeterli olması açısından kritik olmaktadır. “B” noktası seyir irtifasına tırmanırken gerekli olan maksimum itkiyi, dolayısı ile motorun büyüklüğünü etkileyen noktadır. Seyahat süresi çok uzun olan, yüksek itkili bir motor için “C” noktası kritik bir noktadır. “D” noktası motor gücünün azalması buna paralel olarak yakıt miktarının düşmesi nedeniyle yanma açısından önemli olmaktadır.

Ayrıca uçağın yavaşlaması sırasında buzlanma ortamından geçmesi nedeniyle yeterli buz çözme havası enerjisi olması gerekmektedir.

Bölgesel ve hafif uçaklarda kullanılan turbofan motorlarda orta by-pass oranlı ve orta basınç oranlı motorlar kullanılmaktadır. “A” noktası ve “B” noktası bu motorların tasarımında öne çıkmaktadır. Seyahat süresi uzun olmadığı durumlarda “C” noktasındaki yakıt sarfiyatının önemi azalmaktadır. Bunun yerine daha hafif ve ekonomik motor tasarımları tercih edilmektedir. Turboprop motorlar genellikle pervane dişli kutusundan dolayı daha ağır olmaktadır. Bununla birlikte ayarlanabilir pervane açıları sayesinde kalkış ve uçuş sırasında istenilen itki seviyesini kolay şekilde sağlayabilmektedirler. “C” noktasındaki uçuş hızları 0.7-0.9 Mach seviyelerindedir.

Helikopter motorlarının uçuş profilleri turbofan motorlardan farklıdır, helikopterlerin uçaklardan en büyük farkı havada asılı durabilmeleridir. “A” ve “B” noktalarında yüksek shaft gücü vermeleri istenir. “C” noktasında ihtiyaç duyulan güç ise bundan %50 aşağıda olabilir. Helikopterlerin hızları uçaklara göre daha düşüktür ve menzilleri doğrudan “C” noktasındaki yakıt sarfiyatına bağlıdır. Şekil 4.3’te üç farklı helikopter çalışma noktasının kompresör haritası üzerinde yerleri gösterilmiştir. Helikopter ileri doğru düşük hızda uçarken en düşük güç seviyesine ihtiyaç duyarken, ileri doğru yüksek hızda uçarken yaklaşık olarak dört kat fazla güce ihtiyaç duymaktadır [46]. Bu tasarımda orta-yüksek güç kullanımında kompresörün verimli olduğu bölgenin hedeflenmesi gerektiği söylenebilir.



Şekil 4.3. Helikopter motoru çalışma noktaları [46]

4.1. Motor Tasarım Süreçleri

(Walsh & Fletcher, 1998), motor bileşenlerinin tasarımının çok detaylı ve karmaşık olduğunu, detaylı tasarım bilgisinin şirketlerin özel bilgisi olduğu ve şirketlerin bunları

tasarlayabilmek için kendilerine has tasarım kuralları ve araçları geliştirdiklerini söylemiştir [47]. Literatürde mevcut diğer kitaplardan farklı olarak sadece formülleri vermediklerini ve motor bileşenlerinin temel tasarımı için örnek yaklaşımları açıkladıklarını belirtmişlerdir.

(Mattingly, Heiser, & Pratt, 2002) uçak motoru tasarımının, birbiriyle etkileşimde bulunan birçok disiplini ilgilendiren iteratif bir süreç olduğunu söylemişlerdir [48]. Başlıca bu disiplinler; termodinamik, aerodinamik, ısı transferi, yanma, yapısal, malzeme, imalat süreçleri, enstrümantasyon ve kontrol olarak sayılabilir. Tasarım da tek bir çözüm yoktur, tasarımcı birçok faktörü göz önüne alarak tasarım isterlerini sağlayan en uygun tasarımı bulmaya çalışır.

Havacılıkta kullanılan motorların kendi kendilerine çalışan otonom sistemlerden oluşması ve kullanım amacına göre en uygun yakıt sarfiyatını sağlayacak şekilde tasarlanması istenilmektedir. Motorun sağlıklı bir şekilde çalışabilmesi için üzerinde birçok alt sisteme ihtiyaç duyulmaktadır. Bunlar; yağlama, yakıt, elektrik, ikincil hava akışları, kontrol ve sağlık izleme gibi fonksiyonların yerine getirilmesini sağlamaktadır. Kullanım amacına ve bölgesine bağlı olarak çalışma ortamı içerisindeki her yerde bu sistemlerin beklenildiği şekilde çalışması için tasarımın buna uygun şekilde yapılması gerekmektedir.

Hava aracının istenilen uçuş manevralarını yapabilmesi için ihtiyaç duyulan güç, itki, menzil gibi gereklilikler ve ağırlık gibi sınırlayıcı faktörler en uygun termodinamik çevrimin araştırılmasını yönlendirir. Bunun sonucunda bir tasarım uzayı elde edilir. Termodinamik çevrimler motorun hava debisi, enerji ihtiyacı ve bileşenlerin maruz kalacağı tahmini sıcaklıklar gibi temel bilgileri ortaya çıkartır. Kütleli debiler, basınç oranı ve sıcaklıklar motor bileşenlerinin seçilmesinde ve boyutlandırılmasında kullanılır. Bileşenlerin tasarımında birden fazla uygun seçenek olabilir.

Kullanım amacına bağlı olarak motor tasarımları farklılıklar göstermektedir. Örneğin askeri uçaklarda ileri doğru uçuş hızı yüksek olduğundan kompresör girişindeki toplam hava basıncı genellikle yüksektir ve bu nedenle kompresör basınç oranının çok yüksek olması gerekli değildir. Askeri motorlardan beklenen birincil ister yüksek itki ağırlık oranıdır. Uçuş profili ve itki seviyesi o andaki göreve bağlı olarak farklılıklar göstermektedir. Sivil ulaşım amacıyla kullanılan yolcu ve kargo uçaklarında ise kullanım genellikle standart bir uçuş profilidir ve düşük işletim maliyeti en öncelikli isterdir. Bu durumda yakıt sarfiyatını düşürmek için yüksek basınç oranlı kompresörler ve yüksek verimli türbinler tasarlanması gerekmektedir.

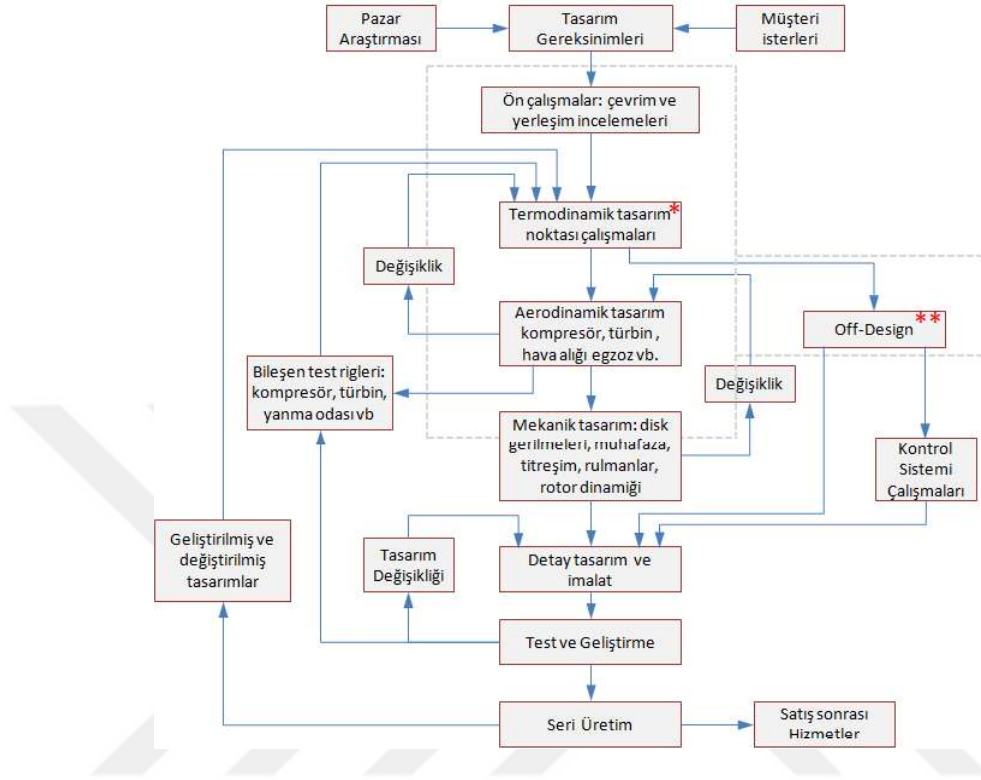
Helikopterlerde ise kalkış sırasında ve acil durumlarda yüksek şaft gücüne ihtiyaç duyulurken normal uçuş sırasında neredeyse tam gücün yarısı kadar bir güce ihtiyaç duyulmaktadır. Bu nedenle bu tarz motorların geniş bir aralıkta verimli şekilde çalışması beklenmektedir. Genel olarak bakıldığında bir uçak motorun tasarımında aşağıdaki noktalar öne çıkmaktadır;

- Kullanım zarfındaki tüm sıcaklık ve irtifalarda kalkış yapabilecek yeterli gücü sağlaması.
- Tırmanma sonundaki itki seviyesinin yeterli olması.
- En çok kullanılan bölgedeki yakıt sarfiyatının en düşük olması, böylece menzilin arttırılması veya yakıt giderlerinin azaltılması.
- Motor ağırlığının en az olması.
- Ekonomik işleme sahip olması.
- Parça ömürlerinin uzun olması.
- Düşük bakım/yenileme maliyeti.

Her üretici firmanın kendi tecrübesine bağlı olarak izlediği motor tasarım adımları farklı olmakla birlikte tipik olarak izlenen ana tasarım adımları Şekil 4.4'te gösterilmiştir. (Mattingly, Heiser, & Pratt, 2002) burada nelere dikkat edilmesi gerektiğini açıklamışlardır [48]. Sıfırdan bir motor tasarımı yapılması zahmetli ve pahalı bir yöntem olarak görüldüğünden, günümüzde motor üretici firmalar mevcut motorlarda tasarım değişiklikleri yaparak, çekirdek adı verilen, motorun güç üreten doğrulanmış özünü kullanarak, daha güçlü, daha verimli ve farklı uygulamalara yönelik motorlar geliştirmektedirler. Bu sayede birçok alanda karşılaşılabilecek olan belirsizliklerden kaçınılmaktadırlar. Bazı durumlarda motor bileşenleri üzerinde yeterli tecrübe olduğundan test edilip yeni seçenekler aranmasının bile gerekli olmadığı değerlendirilmektedir.

Motor ve bileşenlerinin tasarımı ile ilgili literatürde oldukça faydalı bilgiler bulunmaktadır [47-54]. (Cohen, Rogers, & Saravanamuttoo, 1972) tarafından verilen tasarım adımlarına göre tasarım sürecinin başlangıcı uçak üreticisinden ve pazar araştırmalarından gelen isteklerdir [49]. Tasarlanan motorun ekonomik olarak üretilmesi ve üründen maksimum seviyede kar sağlanabilmesi için kullanılacağı uçakların yeteri kadar fazla olması gerekmektedir. Bu istekler teknik olarak tasarlanacak motorun tanımlanmasında kullanılır. Termodinamik çevrimler üzerinde ön çalışmalar yapılır ve

yerleşimsel olarak kurgular yapılır. Daha sonra en uygun görülen tasarım bölgesinde tasarımın yapılması için ON-DESIGN adı verilen tasarım noktası hesaplamaları yapılır.

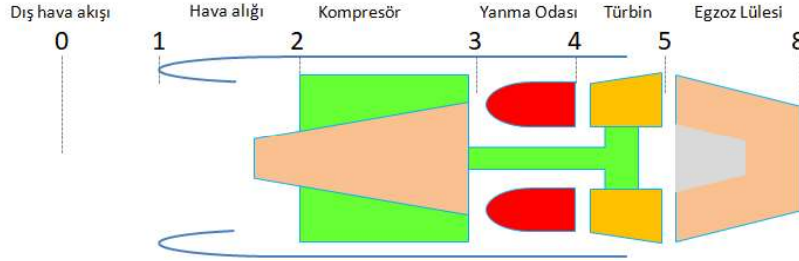


Şekil 4.4. Tipik gaz türbini tasarım adımları [48]

Hesaplamalar motor tasarımında kullanılacak bileşenler teknoloji seviyeleri ve yapılabirliklerine göre ön kabuller içermektedir. Performans hesaplamaları (Şekil 4.4'te * ve ** ile işaretlenmiştir) motor bileşenlerinin tasarlanması için tasarım girdilerini oluşturur. Bütün adımlar önemlidir ancak Şekil 4.4'te kesikli çizgiler içinde kalan bölge en kritik çalışmaları içeren alandır. Tasarım ve geliştirme süreçleri çok disiplinin dahil olduğu iteratif çalışmalardır. Motorun ana bileşenleri üzerinde yapılan detay çalışmalar ile performans analizlerinde kullanılabilecek yeni girdiler elde edilir. Böylelikle motorun tasarım noktası dışındaki davranışı daha iyi bir şekilde incelenebilmektedir.

Termodinamik çevrim hesapları ve ön performans hesapları motor tasarımının başında oldukça fazla yer tutan ve aynı zamanda en önemli faaliyetlerdir. Parametrik çevrim analizleri yapılan bu aşamada "Lâstik Motor" adı verilen boyutları henüz belirlenmemiş bir motor bulunmaktadır [55]. Motor bileşenlerinin detaylandırılması ve motorun boyutlandırılması aşamaları daha sonra gelir. Motor çevrim ve performans hesaplamalarında motor üzerindeki diğer sistemlerin etkileri ilk aşamada göz ardı edilebilir

veya hesaplamalara kabaca dâhil edilebilir. Performans hesaplamalarında basitten başlanarak karmaşıklığın arttığı yönde bir yol izlenir. İlk adımlarda temel bileşenler olan kompresör, yanma odası ve türbin kullanılarak en uygun termodinamik çevrim araştırılır.



Şekil 4.5. Klâsik bir turbojet motorun bileşenleri

Daha sonra ana hava akış yolu üzerindeki parçalar olan hava alığı (intake/inlet), fan, kompresör, yanma odası, yüksek basınç türbini, düşük basınç türbini, ara akış kanalları ve egzoz dahil edilir. Motor üzerinde güç çeken jeneratör ve pompalar, ana hava akışından hava çıkışları, türbin soğutma havası ve bunun gibi elde edilen itkiye ve dolayısı ile yakıt sarfiyatına etkisi olan tüm faktörler göz önüne alınır.

4.2. Motor Performans Optimizasyonu (Literatür incelemesi)

Motor performans optimizasyonu, belirlenmiş çalışma koşulları altındaki en uygun termodinamik çevrimin belirlenmesi amacıyla yapılır. Bu optimizasyon; kompresör basınç oranı, fan basınç oranı, by-pass oranı, türbin giriş sıcaklığı gibi temel parametreleri içerir.

(Cesar, Vishal, Riti, & Pericles, 2015) çevre dostu uçak motoru çevrimlerinin bulunması amacıyla bir çalışma yapmışlardır [56]. Üç farklı tipteki turbofan motorunun termodinamik çevrimi üzerinde girdi olarak Tablo 4.3'te verilen parametreleri kullanarak, uçuş şartlarına bağlı en düşük özgül yakıt tüketimini, en düşük yakıt sarfiyatını ve en düşük nitrojen oksit salınımını bulmak için, çok disiplinli tasarım kodlarını kullanarak, optimizasyon gerçekleştirmişlerdir.

Tablo 4.3. Tasarım parametreleri

Parametre ismi	Birim	Alt sınır	Üst sınır
FPR	-	1,1	1,9
IPCR	-	2,0	5,0(15,0*)
HPCPR	-	5,0 (2,0*)	20,0(15,0*)
BPR	-	2,0	15,0
TET	K	1,200	1,800
TET TO	K	1,200	1,900

(Atashkari, Nariman-Zadeh, Pilechi, Jamali, & Yao, 2005), ideal turbojet çevrim parametrelerini bulmak için çok hedefli genetik algoritmalar kullanarak pareto front grafiklerini oluşturmuşlardır [57]. (Boulkeraa & Ghenaiet, 2010), NSGA-II (Non-dominated Sorting Genetic Algorithm) stokastik optimizasyon algoritmasını kullanarak turboprop motor çevrimini iyileştirmişlerdir [58]. Girdi vektörü olarak $X=\{CPR, TIT \text{ ve } TTR\}$ (basınç oranı, türbin giriş sıcaklığı ve türbin sıcaklık oranı) ve hedef fonksiyonu olarak $F=\{SFC, W_{sp-1}\}$ (özellik yakıt tüketimi ve özellik güç) kullanmışlardır. Tasarım kısıtlamaları ise penaltı fonksiyonları olarak ele alınmıştır. Elde edilen sonuçlara göre, ulaşılan en iyi (optimum) değerlerin tasarım kısıtlarına son derece hassas şekilde bağlı olduğunu belirtmişlerdir.

Tablo 4.4. Girdi parametreleri [58]

Parametre	Alt sınır	Üst sınır
OPR	7	25
TTR	0,5	0,8
TIT [K]	1100	1205

Tablo 4.5. Tasarım kısıtları [58]

Tasarım parametresi	Kısıtlama
T3 [K]	≤ 750
T4.5 [K]	≤ 1050
Kalkış Gücü [hp]	≥ 4700
Seyir Gücü [hp]	≥ 2000

(Choi, Chung, & Sung, 2015), askeri uçakta kullanılan bir turbofan motorunun performansını parçacık sürüşü yöntemi ile optimize etmişlerdir [59]. Girdi olarak bypass oranı, fan basınç oranı, kompresör basınç oranı ve yanma odası çıkış sıcaklığını kullanmışlardır. Hedef olarak maksimum motor itkisi seçildiğinde Tablo 4.6'daki değerler, hedef olarak özellik yakıt tüketimi seçildiğinde Tablo 4.7'deki değerleri elde etmişlerdir. Hedef fonksiyonu bileşenlerinin farklı ağırlıklandırılması ile elde edilen optimum parametre değerleri bunlar arasında değişmiştir.

Tablo 4.6. Maksimum net itki için optimum parametreler [59]

Parametre ismi	Değeri
By-pass oranı	0,201
HP Basınç oranı	4,183
Fan Basınç oranı	2,614

Tablo 4.6. Maksimum net itki için optimum parametreler [59](devam)

Parametre ismi	Değeri
Yanma odası Çıkış sıcaklığı	1734
Net itki	58,52

Tablo 4.7. Minimum özgül yakıt tüketimi için optimum parametreler [59]

Parametre ismi	Değeri
By-pass oranı	0,594
HP Basınç oranı	7,745
Fan Basınç oranı	5,725
Yanma odası Çıkış sıcaklığı	1516
Net itki	13,34

(Homaifar, Lai, & McCormic, 1994), bir turbofan motorunun özgül itkisini ve toplam verimini genetik algoritma yöntemiyle optimize etmişlerdir [60]. Özgül itkiyi ve toplam verimi hem ayrı hedef fonksiyonları hem de birlikte bir tek hedef fonksiyonu içinde tanımlamışlar ve sonuçları kıyaslamışlardır. Girdi olarak Tablo 4.8'deki parametreleri kullanmışlardır.

Tablo 4.8. Optimizasyon parametreleri ve sınırları [60]

Tasarım Parametresi	Alt Sınır	Üst Sınır
Mach	0,3	0,9
CPR	5	25
FPR	1	3
BPR	0,1	6

Yanma odası çıkış sıcaklığı olarak $T_{04}=1555$ K için buldukları en yüksek toplam verimi 0,22 olarak belirtmişlerdir.

(Rifai, 2005), yüksek lisans tezinde, motor ağırlığı ve yakıt tüketimini azaltma hedefiyle iki tane genetik algoritma yöntemi ile bir tane klasik optimizasyon yöntemini karşılaştırmıştır [61]. Motorun ağırlığının azaltılması daha fazla yakıt tüketmesine neden olacağından iki hedefinde en uygun olduğu yeri pareto grafikleri ile araştırmıştır.

(Reffold, 1995), bir turbojet motorunun tasarım noktasındaki ve tasarım noktası üzerinde olmayan çalışma noktalarında performans hesaplarının yapılması için basit şekilde uygulanabilecek bir yöntemi anlatmaktadır [62]. Bu yöntemle hazırlanan (ilk hali Lotus olan) bir Excel dosyasının, öğrencilere verilen eğitimde kullanılmasını amaçlamıştır.

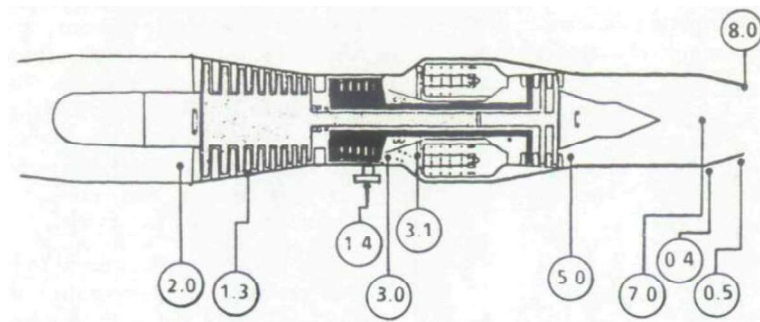
(Kurzke, 1998), gaz türbinlerinde kısıtlamaları sağlayacak şekilde en optimum çevrim noktasının bulunmasında optimizasyonun kullanıldığından bahsedilmiştir [50]. Gradyan yönteminin çözüm noktasını hızlı bir şekilde bulabildiği ancak kısıtlamalar nedeniyle en uygun çözümü bulmakta yetersiz kalabildiğini belirtmiştir. Alternatif olarak rastgele arama yöntemi ile optimizasyon yapılabileceğini söylemiştir. İki şaftlı bir turbofan motorun itki artırımı için en uygun çevrim noktasının optimizasyon yöntemi ile nasıl bulunduğunu anlatılmıştır.

(Kurzke, 2005) gaz türbin üreticilerinin tasarım sırasında çok detaylı bilgiler içeren performans modelleri kullandıklarını ve bu bilgilerin sadece bazılarının kullanıcılara verildiği belirtilmiştir [51]. Verilen bilgilerin genellikle kalkış noktasındaki kompresör basınç oranı, egzoz gaz sıcaklığı gibi bilgiler olduğunu ve bunlar kullanılarak ancak itki, yakıt sarfıyatı, gaz sıcaklığı gibi temel seviyede bilgilerin hesaplanabileceğini belirtmiştir.

Uçak motorlarının performans hesaplamalarını yapabilmek için ticarî programlar bulunmaktadır. Kurzke (2005), bu programları kullanarak performans hesaplaması yapılması için bir yöntem anlatmıştır. Yöntem dört adımdan oluşmaktadır;

- Uygun bir referans çevrim noktası seçilir. Bu noktaya uygun şekilde veriler hazırlanır.
- Mimarî olarak benzer kompresör ve türbin haritaları seçilir ve ölçeklenir.
- Hesaplamalar yapılarak mevcut performans verileri grafiksel olarak karşılaştırılır. İteratif olarak adaptasyon yapılır.
- Tahmin edilen haritalar üzerinde kompresör hızları ayarlanır. (Kurzke, 2005)

Kurzke (2005) buna örnek olarak iki şaftlı bir turbojet motor üzerinde yöntemin uygulaması göstermiştir. Buradaki motor J57-19W model, 50,7 kN itki üreten iki şaftlı bir turbojet motorudur ve test verileri (Ashwood, 1990) tarafından verilmiştir [63]. Testler birçok farklı motor bremzesinde tekrar edilmiş ve sonuçlar karşılaştırılmıştır. Ölçüm alınan istasyonlar Şekil 4.6’da gösterilmiştir.



Şekil 4.6. İki şaftlı turbojet motoru şematik gösterimi ve enstrümantasyon noktaları [63]

Bremzeler şartlandırmasız ve irtifa şartlandırılmalı olmak üzere iki çeşide ayrılmaktadır. İrtifa şartlandırılmalı bremzelerde motora giren havanın toplam basıncı, toplam sıcaklığı ve ram basınç oranı bağımsız olarak kontrol edilebilmektedir. Ashwood (1990) tarafından verilen testlerdeki çevresel parametreler aşağıda gruplandırılmış olarak verilmiştir.

- 82,7 kPa giriş basıncı ve 1 ram sıkıştırması ile dört farklı giriş sıcaklığı (253, 268, 288 ve 308 K)
- 288 K giriş sıcaklığı ve 1,3 ram sıkıştırma oranı ile dört farklı giriş basıncı (82,7; 51,7; 34,7 ve 20,7 kPa)
- 288 K giriş sıcaklığı ve 82,7 kPa giriş basıncı ile dört farklı ram sıkıştırma oranı (1,0; 1,06; 1,30 ve 1,70)

(Bakalis & Stamatis, 2010), küçük bir turbojet motorunun test verilerini kullanarak kalibrasyon yapılmasını ele almıştır [64]. Benzer kompresör ve türbin haritaları kullanılarak komponent verimlerini tahmin eden bir simülasyon kodu kullanmışlardır. Basit bir optimizasyon çalışması yaparak deneyleri ve hesaplamaları karşılaştırmışlardır.

(Duponchel, Loisy, & Carrillo, 1992) tarafından, motor sağlık durumu izleme yöntemlerinde, ölçülmeyen motor parametrelerinin hızlı ve güvenilir şekilde tahmin edilmesi için yeni yöntemlere ihtiyaç duyduğu belirtilmiştir [52]. Bunlar sayesinde sıcak kısımlardaki motor parçalarının ömür takipleri daha doğru şekilde yapılabilmektedir. Örnek olarak yanma odası çıkışındaki yönlendirme kanatçıklarındaki metal sıcaklığının tahmin edilmesi açıklanmıştır.

(Choi, Lee, & Yang, 2011), düşük bypass oranlı bir turbofan motorunun performans modellemesini yapmışlardır [65]. Burada motor bileşenlerine ait bilinmeyen değerlerin bulunması için uyum sağlayıcı rastgele arama (Adaptive Random Search) yöntemini kullanmışlardır.

(Choi, Chung, & Sung, 2015), F404 turbofan motorunda çok hedefli (multi-objective) Parçacık Sürüsü Optimizasyonu (Particle Swarm Optimization) yöntemi ile ana tasarım parametrelerini optimize etmişlerdir [59]. Girdi parametreleri olarak bypass oranı, fan basınç oranı, kompresör basınç oranı ve yanma odası çıkış sıcaklığı alınmıştır. Hedef fonksiyonu olarak net itki ve özgül yakıt tüketiminin ağırlıklı bir fonksiyonu alınmıştır.

(Mirza Baig & Saravanamuttoo, 1992), turbofan motorlarında tasarım noktası dışındaki (off-design) performans parametrelerinin hesaplanmasında kompresör ve türbin

haritaları kullanılmadan bazı basit kabuller ile oldukça yakın sonuçlar elde edilebileceğini göstermişlerdir [53]. Özellikle seyir şartlarındaki motor itkisi ve yakıt akışı parametrelerini iyi bir hassasiyetle hesaplamışlardır. Hesaplamalarını literatürde açık olarak verilen Garret TFE-731 ve RB211 motorlarının deniz seviyesi ve irtifadaki itki ve yakıt akışı değerleri ile karşılaştırmışlardır.

(Bourget, Kelmer, & Leonard, 2006) türbinli bir motorun çevriminin genetik algoritma ile optimize etmişlerdir [66]. (Turan, 2007) turbofan motorların elitizimli genetik algoritma ile optimizasyonunu ele almıştır [67]. Bu çalışmalarda uçuştaki özgül yatkı akışı ve kalkış sırasındaki özgül itki en iyi duruma getiren üç adet girdi parametresi (kompresör basınç oranı, fan basınç oranı ve by-pass oranı) araştırılmıştır.

$$MFP = \frac{\dot{m} \cdot \sqrt{T_t}}{P \cdot A} = M \cdot \sqrt{\frac{\gamma \cdot g_c}{R} \cdot \left(1 + \frac{\gamma - 1}{2} M^2\right)} \quad (4.1)$$

(Mavris & Denney, 2009) jet motorunun test dataları ile doğrulanmasında Filtered Monte Carlo ve Single Value Decomposition yöntemlerini araştırmışlardır [68]. Problemin eksik tanımlanmış olmasından ve tam belirlenmemiş doğrusal olmayan yapısından dolayı çözülmesinin zor olduğunu belirtmişlerdir. Durum performans modeli (Status deck) çıkarılması işleminin motorun termodinamik çevrim modelinin test ölçümlerine göre ayarlanması olduğu ifade edilmiştir. Klâsik olarak uygulanan işlem üç aşamadan oluşmaktadır;

- Testten alınan ölçümlere bakılarak incelenecek etken parametreler belirlenir.
- Etken parametrelerin test edilen noktalardaki değişimleri başka değişkenlere göre makul şekilde değişen eğriler halinde ifade edilir. Örneğin kompresör verimi parametresi düzeltilmiş motor devrine, hareketli kanatçıkların pozisyonuna, kompresör giriş Reynolds sayısına, kompresör kanatçık ucu boşluğuna bağlı olarak çizdirilir.
- Daha sonra bu eğriler motor performans modeline uygulanarak test ölçümleri ile hesaplamaların uyumluluğu kontrol edilir. (Mavris & Denney, 2009)

Bu yöntemin doğru şekilde uygulanabilmesi önceki tecrübelerle bağlı olup elle yürütüldüğü için oldukça fazla zaman almaktadır. Mavris ve Denney (2009) eğri uydurma yöntemi ile durum modeli doğrulanmasının olumsuzlukları ele almışlardır. Tam ve doğru tanımlanmış problemlerde tek bir çözümün olması söz konusu iken eksik tanımlanmış (ill-posed) problemlerde bir çözüm olmayabileceği gibi, yaklaşık ve birden fazla çözüm olması söz konusudur. Çözümün araştırılması sırasında bir çözüme ulaşamaz ise bunun nedeni

yeterince araştırma yapılmaması da olabilir. Ayrıca çözümlerin kararlı olmaması gibi bir problem de oluşabilir. Özellikle testte ölçülen değerlerdeki sapmalar çözüm üzerinde çok fazla değişkenliklere neden olabilir. Mavris ve Denney (2009), bu problemlerin çözümünde optimizasyon algoritmalarının kullanılmasının uygun olacağını değerlendirmişlerdir. Doğrusal ve türev esaslı optimizasyon algoritmalarının global minimumu bulamaması nedeniyle uygun olmadığı, sezgisel ve stokastik algoritmaların daha uygun olmakla birlikte her zaman doğru çözüme ulaşamadıkları ve uzun çözüm süresi gerektirdiğini belirtmişlerdir.

Motor performans hesaplamaları sırasında birçok işlem yapılması gerektiğinden motor davranışını ifade eden yaklaşık modeller veya yapa zekâ gibi öğrenen fonksiyonlar geliştirilmesi yönünde çalışmalar yapılmıştır. Örneğin yapay sinir ağları ile doğrusal olmayan fonksiyonlar bile modellenebilmektedir. Bunların da doğru şekilde eğitim sağlanmasının çok zor olmasından dolayı uygun olmayacağı belirtilmiştir. Mavris ve Denney (2009), NPSS motor performans programında Pratt & Whitney PW2037 turbofan motorunun modeli oluşturarak sanal ölçümler elde etmişlerdir. Daha sonra ölçülen test verileri üzerinde FMC (Filtered Monte Carlo) ile önemli parametreleri belirlemişler ve ardından SVD (Singular Value Decomposition) uygulaması ile Tablo 4.9’da verilen onbeş adet çarpanı belirlemişlerdir. Performans modelinde bu çarpanların kullanılması ile testte ölçülen değerler arasında çok iyi bir uyum olduğunu belirtmişlerdir.

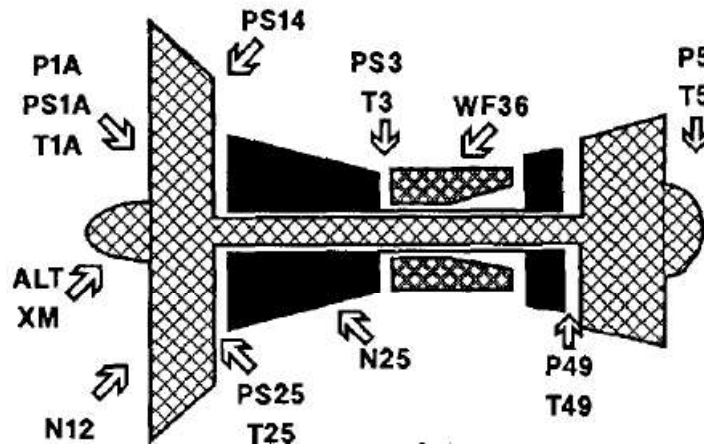
Tablo 4.9. Aşınmış motora ait parametre çarpanları

Parametre çarpanı	Değeri
Fan verim çarpanı	0,986
Fan Akış Düzeltme çarpanı	0,996
HPC verim çarpanı	0,976
HPC Akış Düzeltme çarpanı	0,986
Yanma odası basınç düşümü çarpanı	0,996
LPT verim çarpanı	0,986
LPT Akış Düzeltme çarpanı	0,976
HPT verim çarpanı	0,976
HPT Akış Düzeltme çarpanı	0,976
İtke verim çarpanı	0,981
HPC giriş kanalı basınç düşümü çarpanı	0,996
HPC çıkış kanalı basınç düşümü çarpanı	0,976
LPT çıkış kanalı basınç düşümü çarpanı	0,991

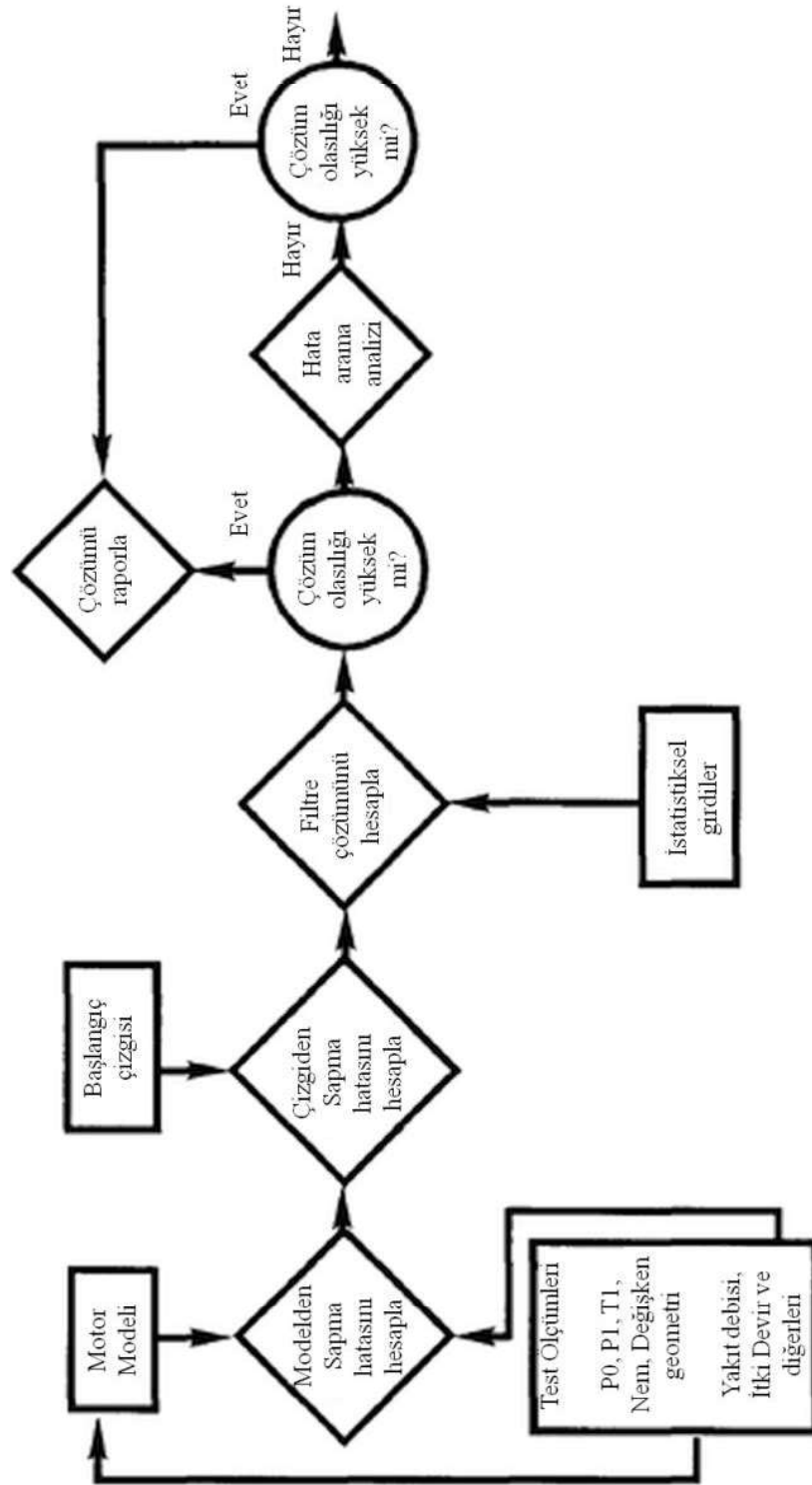
Tablo 4.9. Aşınmış motora ait parametre çarpanları (devam)

Parametre çarpanı	Değeri
Egzoz giriş basınç düşümü çarpanı	0,981
Fan hava kanalı basınç düşümü çarpanı	0,986

Uçak motorları imalât sonrasında bir teste tabi tutularak, ölçülen akış yolu parametrelerinden (Şekil 4.7) kullanıma verilip verilemeyeceğinin kontrol edilmesi gerekmektedir. (L.Doel, 1992), TEMPER adı verilen bir performans izleme ve hata bulma yazılımı ile bu kontrol işleminin nasıl yapılacağını anlatmıştır (Şekil 4.8) [69]. TEMPER, performansta problem var ise bunun kaynağının bulunarak giderilmesi için yol gösteren bir programdır. Aynı zamanda ölçüm hatalarının da değerlendirilmesini yapabilmektedir. TEMPER, uçak üzerine takılan motorlarında performanslarının takip edilmesi ve revizyon tamir ihtiyaçlarının önceden anlaşılacak en kısa şekilde giderilmesine yönelik olarak da kullanılmıştır. Doel (1992), bu program sayesinde erken tespit edilen arızalar olduğu gibi arıza olmadığı halde müdahale edilen durumlarda olduğunu belirtmiştir. TEMPER in kullanımının başarılı olabilmesi için teknisyenlerin tecrübeli olması gerektiği buna rağmen bazı durumların ancak ileri seviyede uzman kişilerce yapılabileceği belirtilmiştir. Bulanık mantık ve yapay sinir ağları gibi konuların gelişmesiyle daha ileri seviyelere gelinebileceğini belirtmiştir.



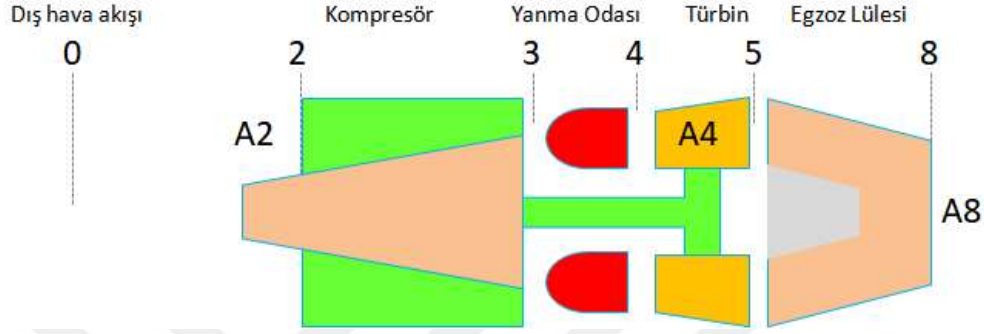
Şekil 4.7. Gaz akış yolu analizinde kullanılan tipik sensörler [69]



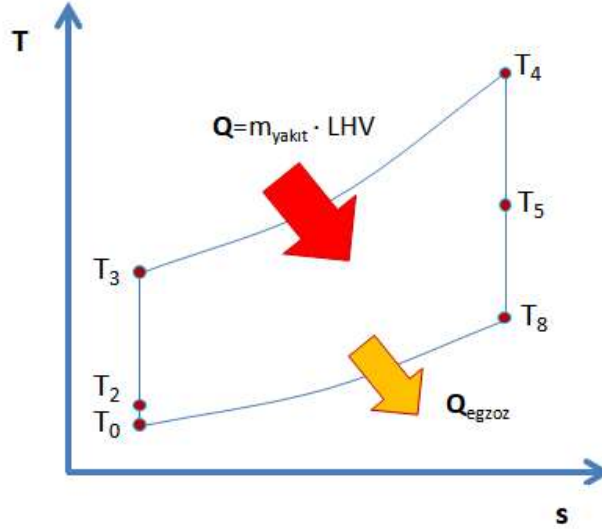
Şekil 4.8. TEMPER programı akış diyagramı [69]

5. TERMODİNAMİK MOTOR PERFORMANS MODELİ

Gaz türbinli motorlarda, hava üzerinde sıkıştırma, yönlendirme, ısıtma ve genişleme işlemleri meydana gelmektedir. Hesaplamalarda, gaz üzerinde uygulanan bütün bu işlemler termodinamik olarak ifade edilmektedir. Şekil 5.1’de bir turbojet motorunun istasyon numaralandırması, Şekil 5.2’de karşılık gelen T-s diyagramı gösterilmiştir.



Şekil 5.1. Bir turbojet motorunun istasyon numaraları



Şekil 5.2. Turbojet motoru T-s diyagramı

Uçağın ileri doğru hızı nedeniyle 0 noktasındaki havanın basıncı ve sıcaklığı artar. Kompresör üzerine uygulanan döndürme işi ile 2 noktasındaki havayı sıkıştırır basıncını artırır 3 noktasına getirir, böylece havadan daha fazla iş elde edilebilmesi için gerekli ön şartı oluşturur. Yanma odasında 3-4 noktaları arasında yakıt püskürtülüp yakılarak hızlı bir şekilde havanın iç enerjisi artırılır. 4 noktasındaki sıcak gazlar türbinde genişletilerek iş çekilir böylece sıcaklığı ve basıncı 5 noktasına düşürülür. Bu çekilen iş kompresörün dönmesi için harcanır. Türbinden çıkan 5 noktasındaki gazların genişleyerek iş yapması için

hala yeterli enerjisi ve basıncı vardır. 8 noktasında gazlar hızlandırılarak egzozdan dışarı atılır ve böylelikle jet itkisi elde edilir.

İdeal turbojet çevriminde, 0-2 ve 2-3 noktaları arasında basınç artışı izentropik, 3-4 noktaları arasındaki yanma odasında yanma işlemi izobarik, 4-5 noktaları arasındaki türbinde gazın genleşmesi izentropik ve 5-8 noktaları arasında egzozdaki genleşme izentropik olmaktadır. 8 noktasında egzoz boğulmuş ise T_8 sıcaklığı T_0 dan farklı bir basınç eğrisi üzerinde olacaktır.

Termodinamik hesaplamalarda kullanılan denklemler aşağıda verilmiştir;

$$P_1 = P_0 \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M_0^2\right)^{\frac{\gamma}{\gamma - 1}} \quad (5.1)$$

$$T_1 = T_0 \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M_0^2\right) \quad (5.2)$$

$$P_2 = P_1 \cdot (1 - \eta_{difüzör}) \quad (5.3)$$

$$T_2 = T_1 \quad (5.4)$$

Aşağıdaki denklemler tek bir kompresör olması durumuna göre yazılmıştır. Birden fazla kompresör var ise önce ilk kompresörün basınç oranı ve verimi kullanılarak çıkışındaki basınç ve sıcaklık değerleri, sonrada benzer şekilde ikinci kompresörün çıkışındaki basınç ve sıcaklık değerleri hesaplanabilir.

$$P_3 = P_2 \cdot PR_{HPC} \quad (5.5)$$

$$T_3 = T_2 \cdot \left[1 + \frac{\left(\frac{P_3}{P_2}\right)^{\frac{\gamma-1}{\gamma}} - 1}{\eta_{HPC}}\right] \quad (5.6)$$

$$\eta_{HPC} = \frac{(T_{3ideal} - T_2)}{(T_{3gerçek} - T_2)} = \frac{\Delta T_{32ideal}}{\Delta T_{32gerçek}} \quad (5.7)$$

Enerjinin korunumu gereği, yanma odasına giren havanın enerjisi ve yakıtın yanma enerjisinin toplamı yanma odasından çıkan gazların enerjisine eşittir;

$$Q_{yakıt} + Q_{giren_hava} = Q_{çıkan_hava} \quad (5.8)$$

$$\begin{aligned} \eta_{Yanma} \cdot LHV \cdot w_{yakıt} + (w_2 - w_b) \cdot Cp_{hava} \cdot T_3 \\ = (w_2 + w_{yakıt} - w_b) \cdot Cp_{gaz} \cdot T_4 \end{aligned} \quad (5.9)$$

Genellikle yanma odası çıkış sıcaklığı kontrol edilebilen bir parametre olduğu için girdi olarak verilmektedir. Buna bağlı olarak yakıt debisi hesaplanabilmektedir. T_4 sıcaklığının verilmesi durumunda yanma sonra gazların özgül ısılarının hesaplanması daha kolay olmaktadır. Girdi olarak yakıt debisi verilirse, çıktı olarak T_4 hesaplanabilir, ancak bu durumda hava debisine bağlı olarak T_4 ün sınır sıcaklık değerlerinin limiti aşması mümkün olabilir. Basitleştirme amacıyla çoğu yerde yakıt debisi ve bleed havası miktarları birbirine yakın kabul edilerek sadeleştirme yapılmaktadır.

$$w_{yakıt} = \frac{(w_2 - w_b) \cdot (Cp_{gaz} \cdot T_4 - Cp_{hava} \cdot T_3)}{(\eta_{Yanma} \cdot LHV - Cp_{gaz} \cdot T_4)} \quad (5.10)$$

$$f = \frac{(1 - b) \cdot (Cp_{gaz} \cdot T_4 - Cp_{hava} \cdot T_3)}{(\eta_{yanma} \cdot LHV - Cp_{gaz} \cdot T_4)} \quad (5.11)$$

Alternatif olarak,

$$f = \frac{w_{yakıt}}{w_4} \quad (5.12)$$

$$T_4 = \frac{\eta_{yanma} \cdot LHV \cdot f + (1 - b) \cdot Cp_{hava} \cdot T_3}{(1 + f - b) \cdot Cp_{gaz}} \quad (5.13)$$

Yanma odasındaki basınç kaybı P_3 'ten çıkarılarak, türbin girişindeki toplam basınç bulunur.

$$P_4 = P_3 \cdot (1 - \Delta P_{yo}) \quad (5.14)$$

Türbin tarafından çekilen iş, kompresör tarafından yapılan sıkıştırma işini sağlamalıdır. Burada diğer harcanan işler ihmal edilmiştir ancak bazı durumlarda diğer işlerin de dahil edilmesi gerekebilir. Yine aşağıdaki denklem tek türbin ve tek kompresör için yazılmıştır. İki şaftlı motorlarda şaft ile birbirine bağlı olan kompresör ve türbinlerin işleri birbirine eşit olacak şekilde denklemler düzenlenmelidir.

$$W_{HPC} = \eta_{mek} \cdot W_{HPT} \quad (5.15)$$

$$(w_2 - w_b) \cdot Cp_{hava} \cdot (T_3 - T_2) = \eta_{mek} \cdot (w_2 + w_{yakıt} - w_b) \cdot Cp_{gaz} \cdot (T_4 - T_5) \quad (5.16)$$

$$T_5 = T_4 - \frac{(w_2 - w_b) \cdot Cp_{hava} \cdot (T_3 - T_2)}{\eta_{mek} \cdot (w_2 + w_{yakıt} - w_b) \cdot Cp_{gaz}} \quad (5.17)$$

$$(T_{5ideal} - T_4) = \frac{(T_5 - T_4)}{\eta_{HPT}} \quad (5.18)$$

$$P_5 = P_4 \cdot \left(\frac{T_{5ideal}}{T_4} \right)^{\frac{\gamma}{\gamma-1}} \quad (5.19)$$

$$T_8 = T_5 \quad (5.20)$$

$$P_8 = P_5 \quad (5.21)$$

Egzoz'un boğulmuş olup olmadığı ($M_8=1$) kontrol edilir;

$$P_8 = P_{8s} \cdot \left(1 + \frac{\gamma-1}{2} \cdot M_8^2 \right)^{\frac{\gamma}{\gamma-1}} \quad (5.22)$$

$$\frac{P_8}{P_{8s}} = \left(1 + \frac{1,33-1}{2} \right)^{\frac{1,33}{1,33-1}} \quad (5.23)$$

$$\frac{P_8}{P_{8s}} = 1,850604 \quad (5.24)$$

Buna göre aşağıdaki eşitsizlik sağlanıyorsa egzoz boğulmuş demektir. Boğulmuş egzozda gazların genleşmesi boğazda sınırlanmaktadır. Bu nedenle sıcak gazların atmosferik şartlara kadar genleşmeden motordan çıkması söz konusudur.

$$\frac{P_8}{P_0} \geq 1,850604 \quad (5.25)$$

ise boğulmuş egzoz için çıkış gaz sıcaklığı;

$$T_8 = T_{8s} \cdot \left(1 + \frac{\gamma-1}{2} \cdot 1 \right) \quad (5.26)$$

$$T_{8s} = \frac{T_8}{1,165} \quad (5.27)$$

M8=1 olduğu için egzoz jet hızı direk ses hızı olarak hesaplanabilir

$$V_8 = \sqrt{\gamma \cdot R_{gaz} \cdot T_{8s}} \quad (5.28)$$

Boğulmamış egzoz için;

$$\frac{T_8}{T_{8s}} = \left(\frac{P_8}{P_0}\right)^{\frac{\gamma-1}{\gamma}} \quad (5.29)$$

$$V_8 = \sqrt{2 \cdot c_p \cdot \eta_{egzoz} \cdot (T_8 - T_{8s})} \quad (5.30)$$

veya

$$V_8 = \sqrt{2 \cdot c_p \cdot \eta_{egzoz} \cdot T_8 \cdot \left(1 - \left(\frac{P_0}{P_8}\right)^{\frac{\gamma-1}{\gamma}}\right)} \quad (5.31)$$

Böylece motor performansının hesaplanması için gerekli tüm termodinamik değerler bulunmuş olur. Egzoz çıkış hızı motorun ürettiği itkiyi hesaplamak için kullanılır. Yakıt debisi ve itki oranı özgül yakıt tüketimini vermektedir. Motorun termal, itkisel ve toplam verimi bu şekilde hesaplanır.

$$F = w_2 \cdot ((1 + f) \cdot V_8 - V_0) \quad (5.32)$$

$$TSFC = 3600 \cdot \frac{w_{yakıt}}{F} \quad (5.33)$$

$$\eta_{termal} = \frac{\frac{1}{2} \cdot (V_8^2 - V_0^2)}{f \cdot LHV} \quad (5.34)$$

$$\eta_{itkisel} = \frac{F \cdot V_0}{\frac{1}{2} \cdot (V_8^2 - V_0^2)} \quad (5.35)$$

$$\eta_{toplam} = \eta_{itkisel} \cdot \eta_{termal} \quad (5.36)$$

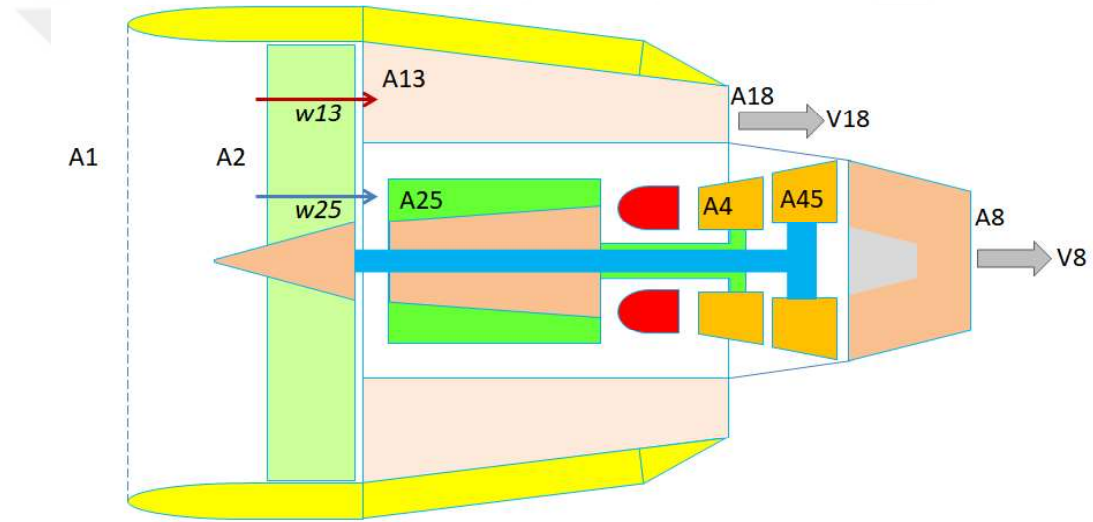
Diğer tüm gaz türbinli motorlarda kullanılan bileşenler, turbojet motorunda kullanılan bileşenlerle benzer şekilde ifade edilebilmektedir. Turbofan motorlarda fan basınç oranı ve by-pass oranı bilindiğinde, fan sonrası basınç, sıcaklık ve geçen hava debisi bulunmaktadır.

$$P_{13} = P_{25} = FPR \cdot P_2 \quad (5.37)$$

$$T_{25_{ideal}} = T_2 \cdot FPR^{\left(\frac{\gamma-1}{\gamma}\right)} \quad (5.38)$$

$$T_{25_{gerçek}} = T_2 + \frac{(T_{25_{ideal}} - T_2)}{\eta_{hpc}} \quad (5.39)$$

$$w_{13} = BPR \cdot w_{25} \quad (5.40)$$



Şekil 5.3. Ayrık akışlı turbofan motor şematik gösterimi

Fan ile LPT şaft birbirine bağlanmaktadır ve çekirdek motor şaftı ile aralarında mekanik bir bağlantı veya güç aktarımı yoktur. Aerodinamik olarak LPT tarafından çekilen iş fan bileşenine aktarılmaktadır.

$$W_{FAN} = \eta_{mek} \cdot W_{LPT} \quad (5.41)$$

$$(1 + BPR) \cdot C_{p_{air}} \cdot (T_{13} - T_2) = \eta_{mek} \cdot (1 + f - b) \cdot C_{p_{gaz}} \cdot (T_5 - T_{45}) \quad (5.42)$$

Bleed olarak çekilen hava düşük basınç türbininden önce akışa karışıyorsa $b=0$ olarak alınabilir. Turbofan motorlarda özgül itki, termal verim ve itkisel verim aşağıdaki denklemler ile ifade edilmiştir.

$$\frac{F}{w_{25}} = ((1 + f) \cdot V_8 - V_0) + BPR \cdot (V_{18} - V_0) \quad (5.43)$$

$$(V_{eşdeğer} - V_0) = ((1 + f) \cdot V_8 - V_0) + BPR \cdot (V_{18} - V_0) \quad (5.44)$$

$$(V_{eşdeğer} - V_0) = (1 + f) \cdot V_8 + BPR \cdot V_{18} - (1 + BPR) \cdot V_0 \quad (5.45)$$

$$V_{eşdeğer} = (1 + f) \cdot V_8 + BPR \cdot (V_{18} - V_0) \quad (5.46)$$

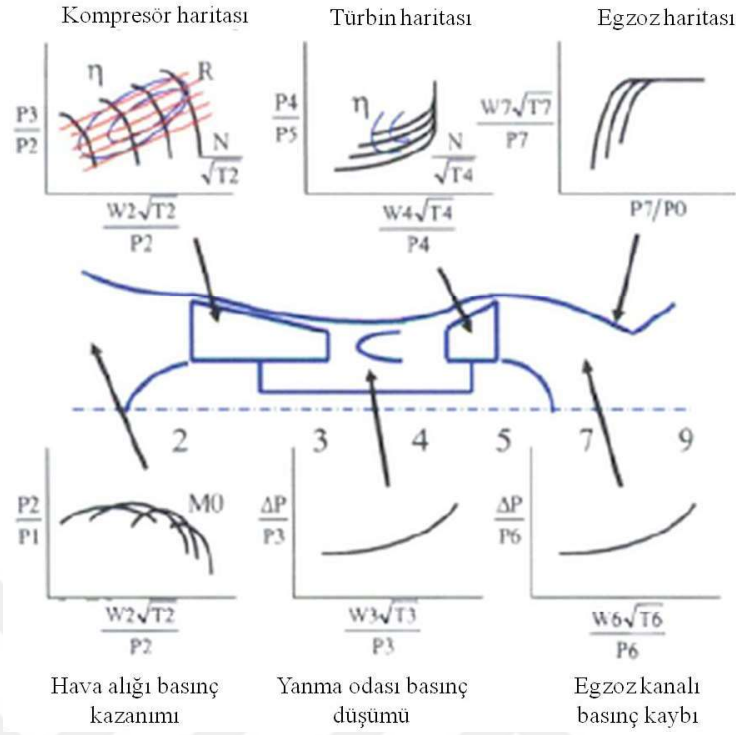
$$\eta_{termal} = \frac{\frac{1}{2} \cdot (V_8^2 - V_0^2) + BPR \cdot \frac{1}{2} \cdot (V_{18}^2 - V_0^2)}{f \cdot LHV} \quad (5.47)$$

$$\eta_{itkisel} = \frac{F \cdot V_{eşdeğer}}{\frac{1}{2} \cdot (V_8^2 - V_0^2) + BPR \cdot \frac{1}{2} \cdot (V_{18}^2 - V_0^2)} \quad (5.48)$$

Düşük basınç türbinine fan bağlanmayıp şaft gücü çekilirse turboprop veya turboşaft motora dönüştürülmüş olur. Bu tip motorlarda egzozda itki elde edilmesi yerine şaft gücü alınması tercih edildiğinden, serbest türbin basınç oranı yüksek tutulmaya çalışılır. Böylece çıkıştaki gaz basıncı atmosferik basınca neredeyse eşit olmaktadır.

Termodinamik hesaplamalarda kullanılan denklemler motorun tüm çalışma noktalarında geçerlidir. Ancak bunlar sıfır boyutludur, yani hiçbir geometrik bilgi içermemektedir. Motordaki bileşenlerin verimleri, basınç oranları, basınç kayıpları; geometriye, dönüş hızına ve motordan geçen hava debisine bağlı olarak değişmektedir (Şekil 5.4). Bu değişimleri gösteren tablo ve grafiklere performans haritası denilmektedir.

Performans haritaları genellikle deneysel olarak çıkarılmaktadır. Yeni bir motor tasarlanırken benzerlikten faydalanılarak önceden bilinen bileşenlerin ölçeklenmiş haritaları da kullanılmaktadır. Başka bir imkân olarak, hesaplamalı akışkanlar dinamiği analizleri ile performans haritalarının çıkarılması da mümkün olmaktadır. Hesaplama yöntemi, bazı sınırları olmakla birlikte, daha ucuz ve çabuk sonuç veren yöntemdir. Motor çalışma noktası değiştiğinde, motorun devri, geçen hava debisi ve bunlara bağlı olarak birçok parametre değişir. Bileşenlerin performanslarını değişmesi ile motor tekrar dengede çalışabileceği en yakın noktaya gelecektir.

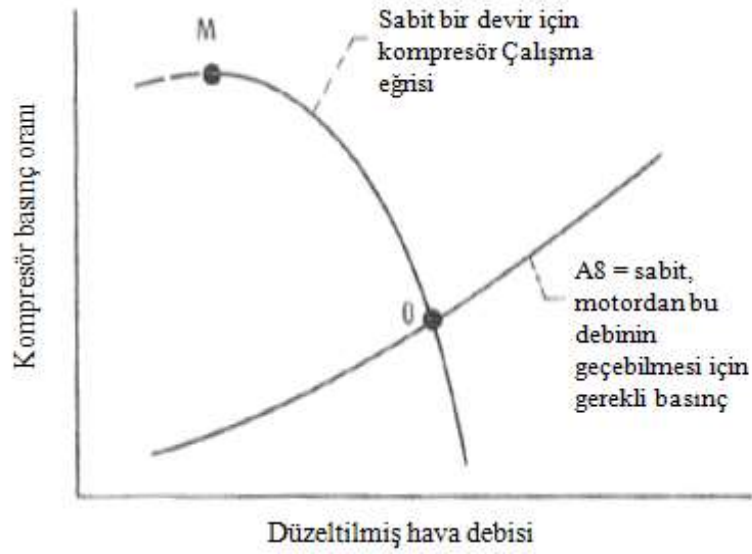


Şekil 5.4. Turbojet motoru bileşenlerinin performans haritaları ve eğrileri [69]

Motor bileşenlerinin farklı güç ve ortamlarda birlikte nasıl çalıştığının bulunması için “Bileşen eşleştirme” (component matching) işlemi yapılır. Bu işlemin grafiksel olarak tek saftlı bir turbojet motorunda uygulaması (G.Evans, 1970) tarafından anlatılmıştır [70].

Kısaca bileşenlerin farklı çalışma noktalarında birbirlerini nasıl etkilediğini açıklamak amacıyla temel elemanlar olan kompresör, türbin ve egzoz lülesi ele alınabilir. Sabit bir devir için kompresör çalışma eğrisi Şekil 5.5’te gösterilmiştir.

Normal çalışma eğrisi üzerinde, kompresör arkasındaki bileşenlerin akışa karşı gösterdikleri direnci yenebilmek için yeterli basınç O noktasında sağlanmaktadır. Türbin ve egzoz çıkış alanı azaltılacak olursa motordan geçebilecek olan hava debisi azalacaktır, çalışma noktası eğri üzerinde sol tarafa doğru gidecektir ve en sonunda kompresör M noktasına geldiğinde stall olacaktır. Tersine egzoz çıkış alanı arttırılacak olursa kompresörün hava debisi artacak ancak basınç oranı azalacaktır, çalışma noktası eğri üzerinde O noktasından sağ tarafa doğru gidecektir. Kompresör boğulacağından hava debisi neredeyse sabitlenecektir.



Şekil 5.5. Kompresör ve sistem çalışma basınç eğrileri [70]

Her kompresör devri için farklı bir M noktası vardır. Bu noktaların birleşimi kompresör stall eğrisini oluşturur. Bir kompresörün kararlı çalışması ve motorun hasarlanmaması için stall eğrisinden yeterince uzak olunması gereklidir.

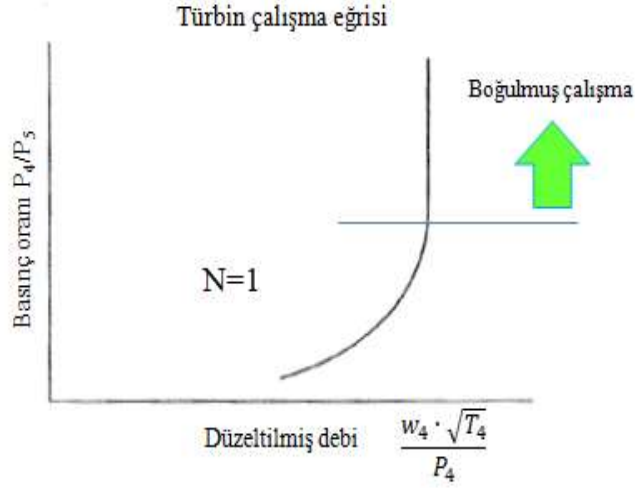
Kompresörün hangi noktada çalıştığına bağlı olarak yanma odasına gelen hava debisi ve basıncı değişecektir. Yanma odası çıkışında istenilen gaz sıcaklığının sağlanabilmesi için ihtiyaç duyulan yakıt miktarı değişecektir.

Şekil 5.6'da kompresör ile aynı devirde dönen bir türbin için çalışma eğrisi gösterilmiştir. Türbinden geçen basınç oranı arttıkça debi artmaktadır. Belirli bir debiye ulaştıktan sonra türbin boğulmakta ve düzeltilmiş gaz debisi artık değişmemektedir. Boğulma noktasından sonra türbin basınç oranı artarsa ürettiği iş artmaya devam edecektir. Ancak bir yerden sonra artık türbin iş üretme kapasitesi aşıldığı için basınç oranı da değişmeyecektir.

Dengede çalışma şartı olarak; kompresörden gelen hava debisi, sıcaklığı ve basıncı eğri üzerindeki düşey konumu belirlemektedir. Türbin eğrisi üzerinde bu debiye karşılık gelen basınç oranı bulunmaktadır. Türbin basınç oranı, giriş sıcaklığı ve verimlilik türbin tarafından yapılabilecek işi belirlemektedir. Motorun kararlı çalışması için; türbinin ürettiği işin, kompresör iş gücünü karşılaması gereklidir.

Türbin sonrasında egzoz lülesi bulunmaktadır. Türbine benzer şekilde egzoz içinde bir basınç oranı-düzeltilmiş debi karakteristik eğrisi vardır. Türbinden gelen gazların atmosfere atılabilmesi için gerekli basınç oranı ve düzeltilmiş debisi sağlanmalıdır. Egzozdan geçebilecek debinin aşılması fiziksel olarak mümkün değildir. Kompresör debisi,

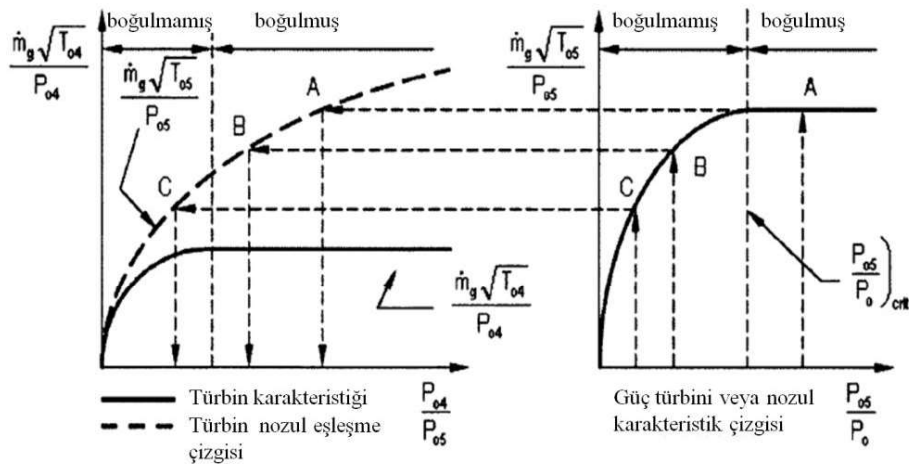
türbinden ve egzozdan geçebilecek debiden fazla ise iterasyonda tekrar başa dönüp yeni bir kompresör çalışma noktası seçilmesi gerekmektedir.



Şekil 5.6. Sabit bir N devrindeki türbin çalışma eğrisi

(Suraweera, 2011) Şekil 5.7’de egzoz ile türbin bileşenlerinin eşleştirilmesi sırasında olası A, B ve C noktaları gösterilmektedir [71]. A noktasında egzoz ve türbin boğulmuş durumdadır. B noktasında türbin boğulmuş ancak egzoz boğulmamıştır. C noktasında egzoz ve türbin her ikisi de boğulmamıştır.

Motordan azamî itkinin sağlanabilmesi için egzozun boğulmuş olması istenmektedir. Egzozun boğulması ile türbin çalışma noktasının değişkenliği azaltılmış olmaktadır. Türbin boğulma noktasına yakın yerlerde en verimli şekilde çalışmaktadır.



Şekil 5.7. Turbojet motorda türbin ve egzoz eşleşmesi [71]

Bileşenlerin eşleştirilmesi hesaplarında ilk olarak bir kompresör çalışma noktası seçilip daha sonra diğer bileşenlerin dengeye gelip gelmedikleri, istenilen hava debisini geçirip geçiremedikleri kontrol edilir. Sistem dengeye gelmediyse yeni bir kompresör çalışma noktası seçilir. Denge sağlandığı görülene kadar iteratif bir arama yapılması gerekmektedir. Bu şekilde hesaplama iç-içe döngüsel hesaplama yöntemi denilmektedir [47]. Beşin üzerindeki döngü sayılarında çözüm işlemi çok verimsiz olduğundan döngüsel yöntem tercih edilmemektedir.

Bunun dışında matris çözüm yöntemi uygulanmaktadır. Matris yönteminde her parametrenin sonuçlar üzerindeki etkisi hesaplanarak bir gradyan matrisi oluşturulur. Bileşenlerde hesaplanan ve olması gereken debiler arasındaki farklardan bir hata matrisi oluşturulur. Hata matrisinin sıfırlanabilmesi için iteratif olarak Newton Raphson çözümleri yapılır (Eşitlik (5.49),(5.50) ve (5.51)). (Kurzke & Halliwell, 2018), Gasturb programında Newton Raphson yöntemi ile hata hesaplaması yapıldığı ve Adaptive Random Search yöntemi kullanıldığını belirtmişlerdir [72].

(Koenig & Fishbach, 1972) Nasa tarafından geliştirilen performans hesaplama programı GENENG de benzer şekilde bileşenlerde oluşan hatayı toplayıp bunu minimize ederek problemi çözmektedir. Programının akış diyagramı oldukça karmaşıktır (Şekil 5.8).

$$\begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_n \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_n \end{pmatrix}_0 + \begin{pmatrix} \frac{\partial E_1}{\partial V_1} & \dots & \frac{\partial E_1}{\partial V_n} \\ \frac{\partial E_2}{\partial V_1} & \dots & \frac{\partial E_2}{\partial V_n} \\ \vdots & & \vdots \\ \frac{\partial E_n}{\partial V_1} & \dots & \frac{\partial E_n}{\partial V_n} \end{pmatrix} \cdot \begin{pmatrix} \partial V_1 \\ \partial V_2 \\ \partial V_3 \\ \vdots \\ \partial V_n \end{pmatrix} \quad (5.49)$$

$$E_i = E_{0i} + \sum_{j=1}^n \left(\frac{\partial E_i}{\partial V_j} \right) \cdot \partial V_j \quad (5.50)$$

$$E_1 - E_n = 0 \quad (5.51)$$

GENENG programında bileşenlerin eşleştirilmesi için kullanılan değişkenler ve hesaplanan hatalar Tablo 5.1’de gösterilmiştir [73]. Tabloda iki şaftlı turbofan, karışık akışlı turbofan, iki şaftlı ve tek şaftlı turbojet motorlar yer almaktadır.

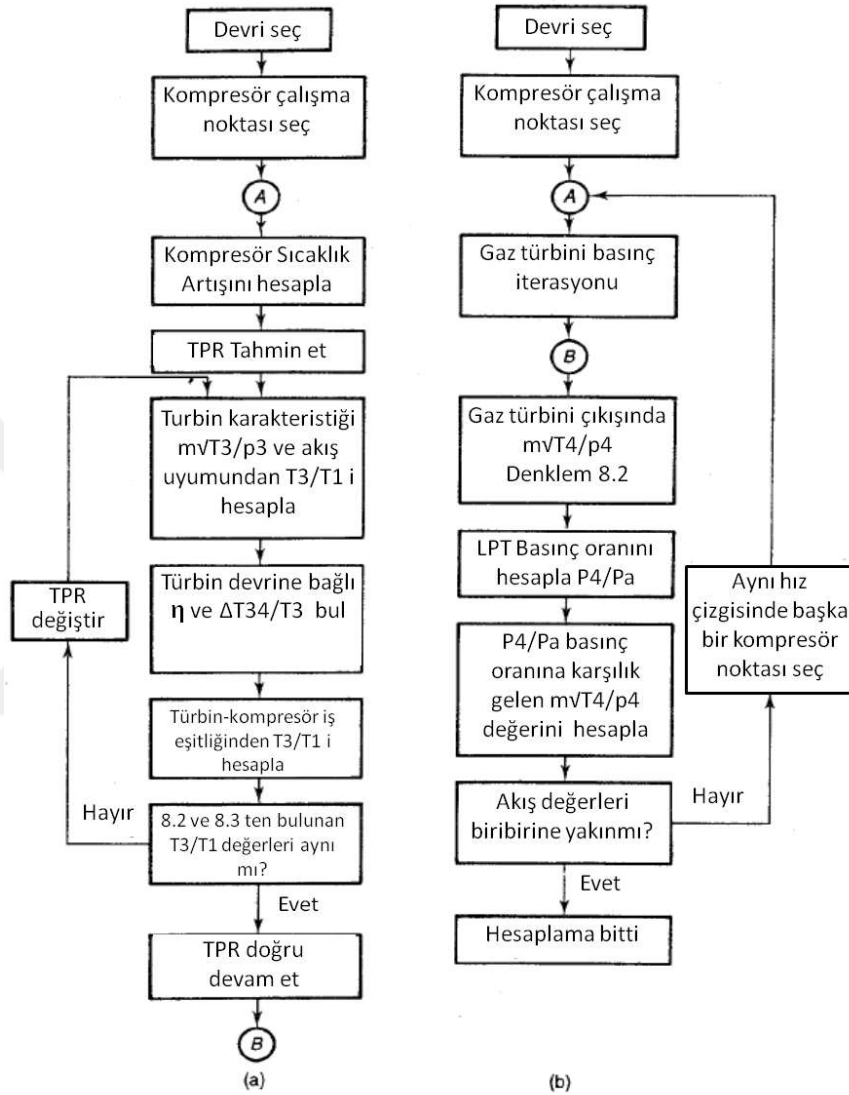
Tablo 5.1. *Newton Raphson hesaplama matrisi parametreleri [73]*

Değişken	İki şaftlı turbobfan	Karışık akışlı turbobfan	İki şaftlı turbojet	Tek şaftlı turbojet
Parametre 1	ZF	ZF	ZF	ZF
Parametre 2	PCNF	PCNF	PCNF	PCNF
Parametre 3	ZC	ZC	ZC	ZC
Parametre 4	PCNC	PCNC	PCNC	----
Parametre 5	TFFHP	TFFHP	TFFHP	----
Parametre 6	TFFLP	TFFLP	TFFLP	----
Hata 1	$\frac{TFHCAL - TFFHP}{TFHCAL}$	$\frac{TFHCAL - TFFHP}{TFHCAL}$	$\frac{TFHCAL - TFFHP}{TFHCAL}$	$\frac{TFLCAL - TFFLP}{TFLCAL}$
Hata 2	$\frac{DHTCC - DHTCHP}{DHTCC}$	$\frac{DHTCC - DHTCHP}{DHTCC}$	$\frac{DHTCC - DHTCHP}{DHTCC}$	$\frac{DHTCF - DHTCLP}{DHTCF}$
Hata 3	$\frac{TFLCAL - TFFLP}{TFLCAL}$	$\frac{TFLCAL - TFFLP}{TFLCAL}$	$\frac{TFLCAL - TFFLP}{TFLCAL}$	$\frac{P7R - P7}{P7R}$
Hata 4	$\frac{DHTCF - DHTCLP}{DHTCF}$	$\frac{DHTCF - DHTCLP}{DHTCF}$	$\frac{DHTCF - DHTCLP}{DHTCF}$	----
Hata 5	$\frac{P25R - P25}{P25R}$	$\frac{PS25 - PS55}{PS25}$	$\frac{WAF - WAC - BLF}{WAC}$	----
Hata 6	$\frac{P7R - P7}{P7R}$	$\frac{P7R - P7}{P7R}$	$\frac{P7R - P7}{P7R}$	----
Matris boyutu	6x6	6x6	6x6	3x3

(Cohen, Rogers, & Saravanamuttoo, 1972), tek şaftlı bir turboşaft motor için, turbojet motor için ve serbest türbinli bir motor için bileşenlerin eşleştirilmesi metotlarını açıklamıştır (Şekil 5.9). Burada hava giriş kanalları ve egzozda basınç kayıpları olmadığı kabul edilmiştir. Turboşaft ve turboprop motorlarda türbinden çıkan gazların tamamen genişleyeceği ve kompresör basınç oranı ile türbin basınç oranlarının aynı olacağı yaklaşımı yapılmıştır. Ayrıca türbin çalışma eğrisinin bir tek eğri üzerinde olduğu kabulünün yapılabileceği belirtilmiştir. Turbojet motorda ise türbindeki sıcaklık farkları eşitlenerek basınç oranlarına geçilmiştir. Eğer daha sonra bir serbest türbin varsa burada basınç düşümü ve akış fonksiyonları karşılaştırılmakta ve arada uyumsuzluk varsa yeni bir kompresör çalışma noktası seçilmektedir.

Motordaki bileşenler arasında uyumsuzluk olması durumunda, istenilen verim değeri elde edilemeyebilir, motor beklenenden daha sıcak çalışabilir veya istenilen itkiyi üretmeyebilir. Motorda kullanılacak olan kontrol sistemi, istenilen devire sağlayacak şekilde yakıt akışını hesaplayacaktır. Ancak bileşenler burada dengede olmadıkları için başka bir noktada dengeye gelmeye çalışacaktır.

Optimizasyon uygulanarak bileşenlerin birbiriyle uyumlu çalışabilmesi için boyutlarının nasıl seçilmesi gerektiği araştırılabilir. Böylece motor tasarım iterasyonlarının sayısı ve dolayısı ile de tasarım için harcanan süre azaltılabilir.

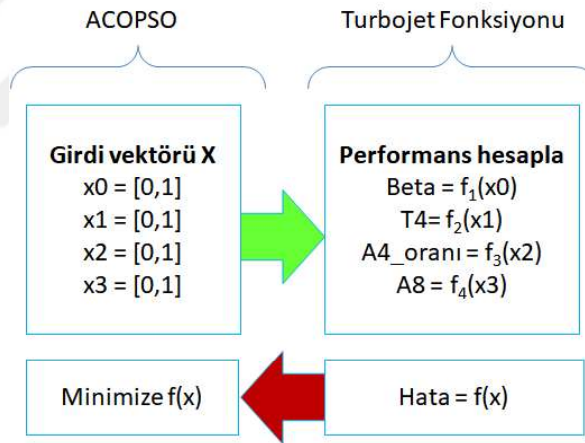


Şekil 5.9. Gaz üretici (a) ve serbest türbin (b) eşleştirme akış diyagramı [49]

6. İTKİ SİSTEMİ PERFORMANS OPTİMİZASYONU

Optimizasyon kodunun turbojet fonksiyonu ile etkileşimi Şekil 6.1’de gösterilmiştir. Girdi vektörü içerisindeki parametre sayısı, turbojet fonksiyonu içinde kullanılacak parametre sayısına göre değiştirilmektedir. Girdi vektörü içerisindeki her bir parametre sıfır ile bir arasında değişmektedir ve turbojet fonksiyonu içerisindeki ilgili girdinin belirlenen sınırlar içerisinde bir değere dönüştürülmesi için kullanılmaktadır.

Hedef fonksiyonu turbojet fonksiyonu içerisinde oluşturulmaktadır. Turbojet fonksiyonu yerine turboprop, turboşaft, turbofan veya başka bir hedef fonksiyonu kullanılabilir. Bu fonksiyon; özgül itki, özgül yakıt tüketimi, kompresör verimi, toplam verim, itkisel verim, bileşenlerdeki akış parametre hata yüzdesi gibi minimize/maksimize edilmek istenilen herhangi birkaç değeri içerebilir. Değerler arasındaki merteye farkları nedeniyle bazılarının baskın olması ve buna bağlı olarak sonucun etkilenmesi söz konusu olduğundan hedef fonksiyon içerisinde ağırlık katsayıları kullanılmaktadır.



Şekil 6.1. Optimizasyon programı ile turbojet fonksiyonu arasındaki etkileşim

Ağırlık katsayılarının bulunması için analiz ilk önce deneme amaçlı çalıştırılarak elde edilen değerler incelenir ve tahmini hedef değer bulunur. Buna bağlı olarak, hedef fonksiyonu değeri 1,0-10,0 arasında kalacak şekilde bir ağırlık belirlenir. Alternatif olarak her hedef tek başına kullanılarak elde edilen sonuçlar kaydedilir ve çoklu hedeflerin bir arada verdikleri sonuçlar ile kıyaslanır. Bu şekilde hedef fonksiyonu içerisine dahil edilen terimlerin ne kadar etkin oldukları izlenir.

6.1. Tasarım Noktası Çevrim Optimizasyonu

İki şaftlı ayrıklı akışlı ticarî bir turbofan motorun termodinamik modeli hazırlanarak belirlenen optimizasyon parametreleri ve farklı hedef fonksiyonları ile elde edilen sonuçlar incelenmiştir. Tasarım noktasında sağlanması gereken itki değeri, aşılmaması gereken yanma odası sıcaklık değeri gibi sınırlayıcı etkenler penaltı fonksiyonu olarak hedef fonksiyona eklenmiştir.

Tablo 6.1. Optimizasyon tanımlama denklemleri

Hedef fonksiyonu	:	minimize $f(\mathbf{x}) = hedef0 \cdot c_0 + hedef1 \cdot c_1 + hedef2 \cdot c_2 + hedef3 \cdot c_3 + hedef4 \cdot c_4$ $c_0 = \{0,1\}, c_1 = \{0,1\}, c_2 = \{0,1\}, c_3 = \{0,1\}, c_4 = \{0,1\}, c_5 = \{0,1\}$
$hedef0$	=	$TSFC \cdot 100,0$
$hedef1$	=	$100,0 / (BPR + \epsilon)$
$hedef2$	=	$100,0 / toplam_verim$
$hedef3$	=	$100,0 / özgül_itki$
$hedef4$	=	$1000,0 \cdot yakıt_debisi$
$hedef5$	=	$W2 \cdot 10,0$
Girdi vektörü	:	$X = \{x_0, x_1, x_2, x_3, x_4, x_5\}, x_i \in [0,1], i = 1,2,\dots,5$
Parametre aralıkları	:	FPR = $1,5 + x[0] \cdot 0,4$; // Fan basınç oranı CPR = $4,0 + x[1] \cdot 26,0$; // kompresör basınç oranı BPR = $0,2 + x[2] \cdot 15,0$; // Fan bypass oranı dP4 = $(4,0 + x[3] \cdot 6,0) / 100,0$; // yanma odası basınç kaybı mf = $0,005 + x[4] \cdot 0,09$; // yakıt debisi oranı (%w25) mb = $0,01 + x[5] \cdot 0,08$; // bleed havası miktarı (% w25)
Kısıt fonksiyonları	:	$İtki = 30000\ N$ $T_4 \leq T_{4_limit}$ $0,9 \leq M_{18} \leq 1,0$ $0,95 \leq M_8 \leq 1,0$

Tablo 6.2. Sabit alınan parametreler

Parametre	Değer	Açıklaması
Ecomp	0,80	Kompresör verimi
Efan	0,80	FAN verimi
Ecomb	0,99	Yanma verimi
ETU	0,87	Türbin verimi
eta_mech	0,99	Mekanik verim
fan_ductloss	0,99	Fan kayıpları
Cd_fan	0,95	FAN çıkış kaybı katsayısı
Cd_core	0,95	Çekirdek motor çıkış katsayısı

Tablo 6.3. *Turbofan motor optimizasyonu analiz sonuç özeti*

Hedef fonksiyon	FPR	CPR	BPR	T4	T6	Özgöl İtki	M8	Toplam Verim
Hedef 0	Orta	Yüksek	Orta	Yüksek	Düşük	Düşük	Düşük	Yüksek
Hedef 1	Orta	Orta	Orta	Yüksek	Orta	Düşük	Düşük	Orta
Hedef 2	Yüksek	Yüksek	Orta	Yüksek	Düşük	Düşük	Düşük	Yüksek
Hedef 3	Yüksek	Düşük	Düşük	Yüksek	Yüksek	Yüksek	Yüksek	Düşük
Hedef 4	Orta	Yüksek	Orta	Yüksek	Düşük	Düşük	Düşük	Yüksek
Hedef 5	Orta	Orta	Orta	Yüksek	Orta	Düşük	Düşük	Orta
Hedef 1+ Hedef 5	Orta	Orta	Orta	Yüksek	Orta	Düşük	Düşük	Orta

Hibrit optimizasyon yönteminin hedef 2 çalışmasında takıldığı gözlenmiştir. Sonuçlar aşağıdaki şekilde yorumlanmıştır;

- Yüksek verimin elde edildiği tasarımlarda özgül itki düşük dolayısı ile motorlar büyüktür.
- Toplam verimin yüksek olması için HPC basınç oranı yüksek olmalıdır.
- Seçilen itki ve T₄ seviyesi için by-pass oranları orta seviyelerde en iyi değerler bulunmuştur. T₄ ün 1900 K seçilmesi ile by-pass oranı iki katına çıkmıştır ve çekirdek motor hava debisi yarıya inmiştir. İtki ihtiyacı 8 katına çıkarıldığında motor boyutları büyümüş ancak parametreler benzer kalmıştır.
- By-pass oranının 14 seviyesine çıkabilmesi için T₄ ün 2500 K ve kompresör türbin verimlerinin son derece yüksek seviyelere çıkarılması gerektiği görülmüştür. Bu sıcaklık seviyesinin mevcut teknolojik imkânlarla elde edilmesi çok zordur.

Optimizasyon ile elde edilen sonuçlar, verilen parametre sınırlarına, diğer kısıtlamalara ve hedef fonksiyonlarına bağlı olarak oldukça değişmektedir. Bu nedenle sonuçların kolay değerlendirilebilmesi amacıyla sayısal sonuçlar verilmemiştir.

İkinci olarak yine turbofan motor üzerinde, (Homaifar, Lai, & McCormic, 1994) tarafından genetik algoritma kullanılarak yapılan çalışma ACOPSO uygulanarak çözülmüş ve bulunan sonuçlar karşılaştırılmıştır. Homaifar (1994), (Pinkel & Karp, 1947) tarafından daha önceden yayınlanan [74] çalışma sonuçları ile karşılaştırma yapmıştır. Pinkel ve Karp (1947) çalışmalarında bazı parametreleri sabit tutarak optimizasyon yapmışlardır. Homaifar

(1994), parametrelerin bazılarının sabit tutulmasının diğer parametrelerin optimum değerlerini etkilediğini bu nedenle hepsinin aynı anda optimize edilmesi gerektiğini belirtmiştir. Girdi parametreleri olarak uçuş hızı, kompresör basınç oranı, fan basınç oranı ve bypass oranı alınmıştır. $T_4 = 811, 1089$ ve 1367 K yanma odası çıkış sıcaklıkları için özgül itkiyi ve toplam verimi maksimize edecek parametreler araştırmıştır.

Tablo 6.4. Optimizasyon tanımlama denklemleri

Girdi vektörü	:	$X = \{x_0, x_1, x_2, x_3\}, x_i \in [0,1], i = 0,1,2,3$
Parametre aralıkları	:	$Mach_0 = 0,3 + x[0] \cdot (0,9 - 0,3)$; // uçuş mach sayısı $CPR = 3,0 + x[1] \cdot (25,0 - 3,0)$; // kompresör basınç oranı $FPR = 1,0 + x[2] \cdot (3,0 - 1,0)$; // Fan basınç oranı $BPR = 0,1 + x[3] \cdot (6,0 - 0,1)$; // Fan bypass oranı

Tablo 6.5. Sabit alınan parametreler

Parametre	Değer	Açıklaması
Ecomp	1,0	Kompresör verimi
Efan	1,0	FAN verimi
Ecomb	1,0	Yanma verimi
ETU	0,85	Türbin verimi
eta_mech	0,99	Mekanik verim
fan_duct_eff	1,0	Fan kayıpları
Cd_fan	1,0	FAN çıkış kaybı katsayısı
Cd_core	1,0	Çekirdek motor çıkış katsayısı

$$TMA = \frac{F}{w_{25}} = ((1 + f) \cdot V_8 - V_0) + BPR \cdot (V_{18} - V_0) \quad (6.1)$$

$$V_{eşdeğer} = (1 + f) \cdot V_8 + BPR \cdot (V_{18} - V_0) \quad (6.2)$$

$$\eta_{toplam} = \frac{F \cdot V_{eşdeğer}}{f \cdot LHV} \quad (6.3)$$

$$Minimize f(x) = \frac{1}{TMA} \text{ (durum 1)} \quad (6.4)$$

$$Minimize f(x) = \frac{1}{\eta_{toplam}} \text{ (durum 2)} \quad (6.5)$$

Durum 1:

Pinkel ve Karp (1947), $Mach_0=0,66$ için optimizasyon sonuçlarını paylaşmıştır. Kullandıkları bileşen verimleri bilinmemektedir, bu nedenle benzer sonuçları elde edecek şekilde Tablo 6.5'teki verimler kabul edilmiştir. Yukarıda verilen parametre ve verimler ile Tablo 6.6'daki sonuçlar elde edilmiştir.

Tablo 6.6. Optimizasyon sonuçlarının karşılaştırılması

		Ref [75]		Bu tez	
Mach0	T4	TMA	CPR	TMA	CPR
0,66	811	245	3,0	243,4	3,1
0,66	1089	400	6,0	488,3	5,4
0,66	1367	610	9,0	776,6	9,0

Tablo 6.7. Sabit alınan parametreler

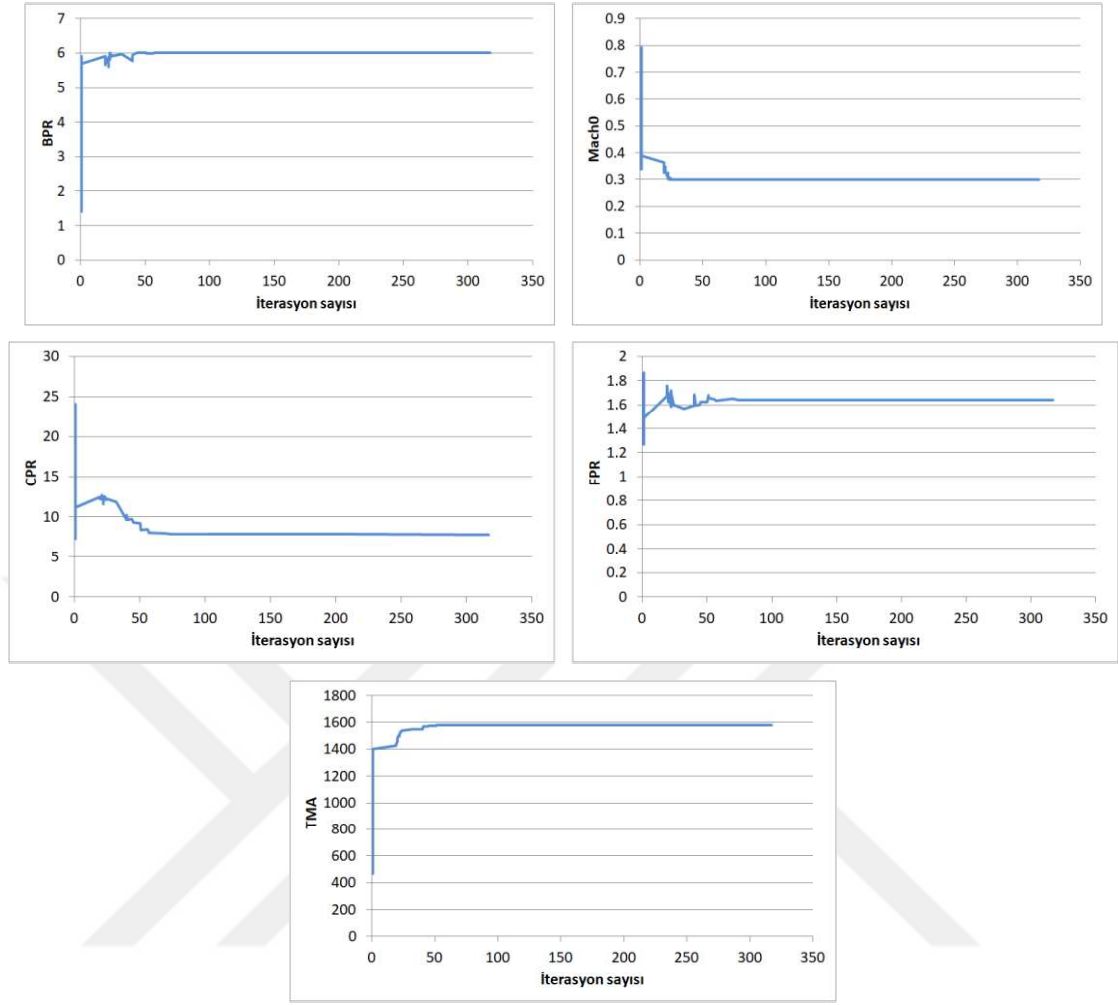
Parametre	Değer	Açıklaması
Ecomp	0,85	Kompresör verimi
Efan	0,85	FAN verimi
Ecomb	0,99	Yanma verimi
ETU	0,85	Türbin verimi
eta_mech	0,99	Mekanik verim
fan_duct_eff	1,0	Fan kayıpları
Cd_fan	1,0	FAN çıkış kaybı katsayısı
Cd_core	1,0	Çekirdek motor çıkış katsayısı

Kompresör, fan ve yanma odası verimleri Tablo 6.7’deki gibi değiştirildiğinde Tablo 6.8’deki sonuçlar elde edilmiştir.

Tablo 6.8. Değiştirilen verimler ile optimizasyon sonuçları

Mach0	T4	TMA	CPR
0,0	811	665,0	3,4
0,0	1089	1420,3	6,1
0,0	1367	1901,2	8,3
0,66	811	146,4	3,2
0,66	1089	379,6	3,6
0,66	1367	609,2	5,9

Uçuş hızı arttığında optimum özgül itki ve paralel olarak kompresör basınç oranı azalmıştır. Optimizasyon iterasyonları sırasında önemli parametrelerin değişimi Şekil 6.2’de gösterilmiştir. İterasyonların sonlandırılması için bir kriter kullanılmamıştır. Bunun yerine iyileşme olduğu sürece hesaplamaları devam edilmiştir. Yakınsamanın gösterilmesi amacıyla uygun bir zaman aralığı seçilmiştir.



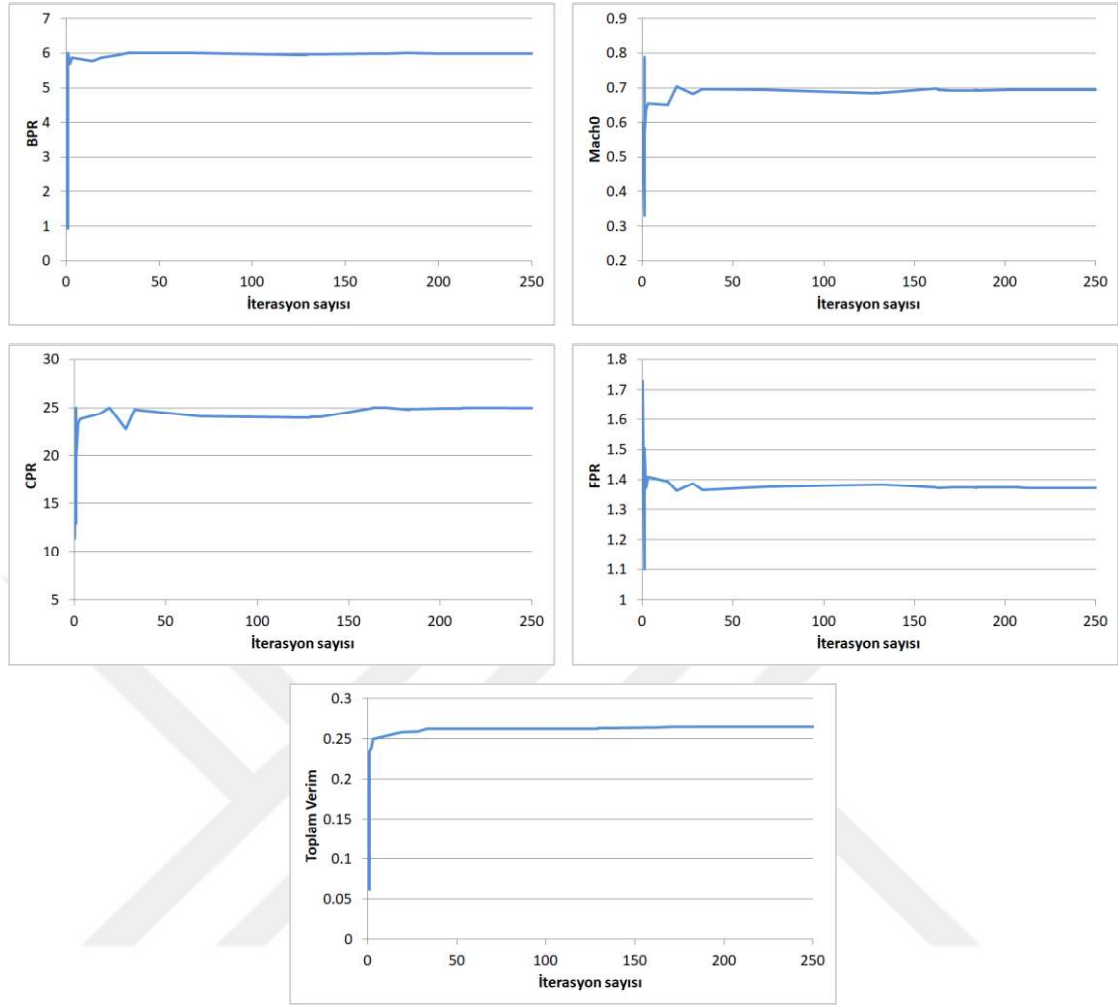
Şekil 6.2. Durum 1 Optimizasyon parametrelerinin iterasyon adımları ile değişimi

Tablo 6.9. Durum 1 Optimizasyon sonuçlarının karşılaştırılması

Parametre	Homaifar [60]	Bu tez
BPR	6	6
M0	0,3	0,3
CPR	18,0	7,78
FPR	1,6	1,64
TMA	1480	1580

Durum 2:

Toplam verimin en iyi olduğu yeri bulmak için yapılan optimizasyon sonucunda elde edilen sonuçların karşılaştırması Tablo 6.10’da ve önemli parametrelerin değişimi Şekil 6.3’te verilmiştir. Yukarıda belirtildiği gibi iterasyonların sonlandırılması için bir kriter kullanılmamıştır.



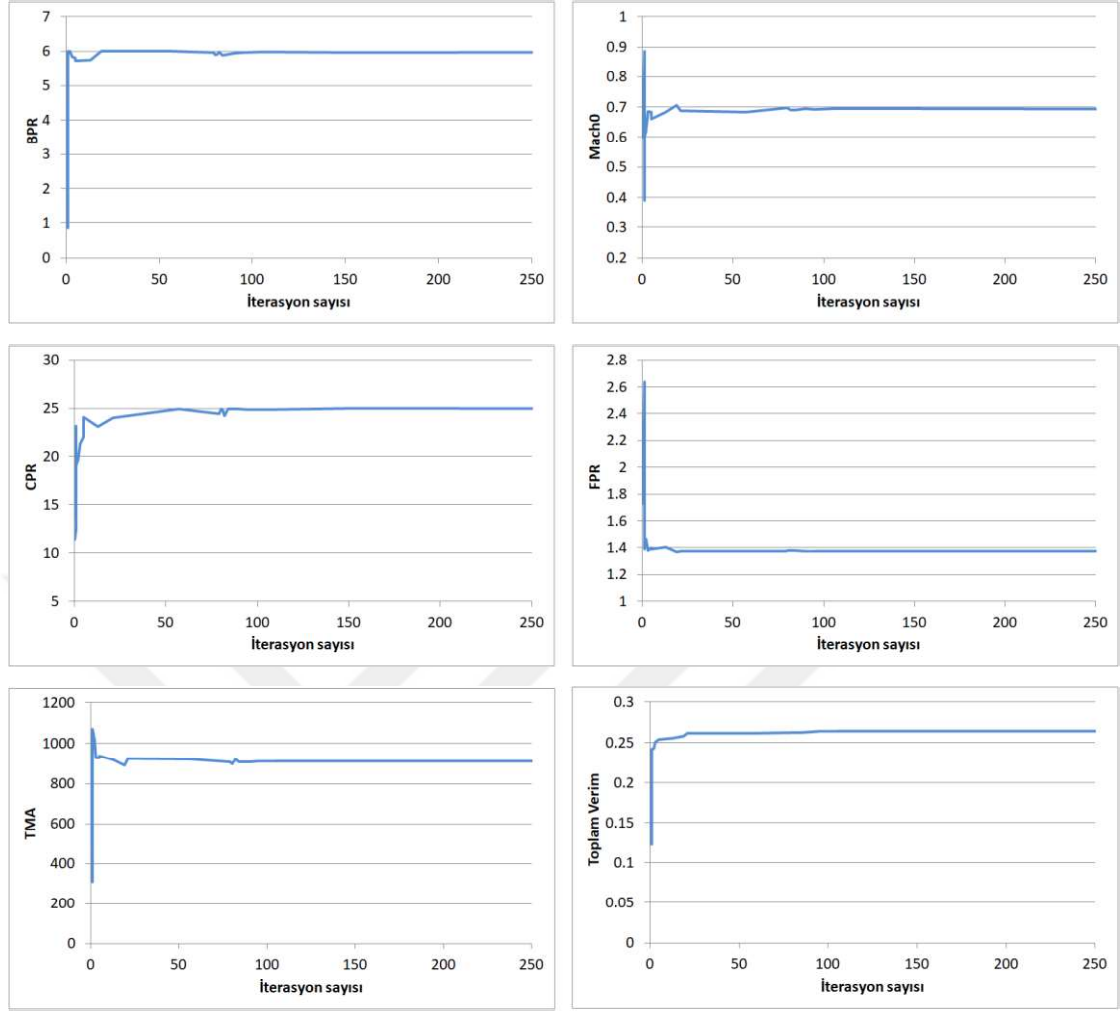
Şekil 6.3. Durum 2 Optimizasyon parametrelerinin iterasyon adımları ile değişimi

Tablo 6.10. Durum 2 Optimizasyon sonuçlarının karşılaştırılması

Parametre	Homaifar [60]	Bu tez
BPR	4,1	6
M0	0,9	0,69
CPR	25,0	24,9
FPR	1,55	1,37
Toplam verim	0,22	0,265

Durum 3:

Hem TMA hem de toplam verimi maksimum yapacak parametreler aranmıştır. Önemli parametrelerin değişimi Şekil 6.4'te, sonuçların karşılaştırması Tablo 6.11'de verilmiştir. Yukarıda belirtildiği gibi iterasyonların sonlandırılması için bir kriter kullanılmamıştır.



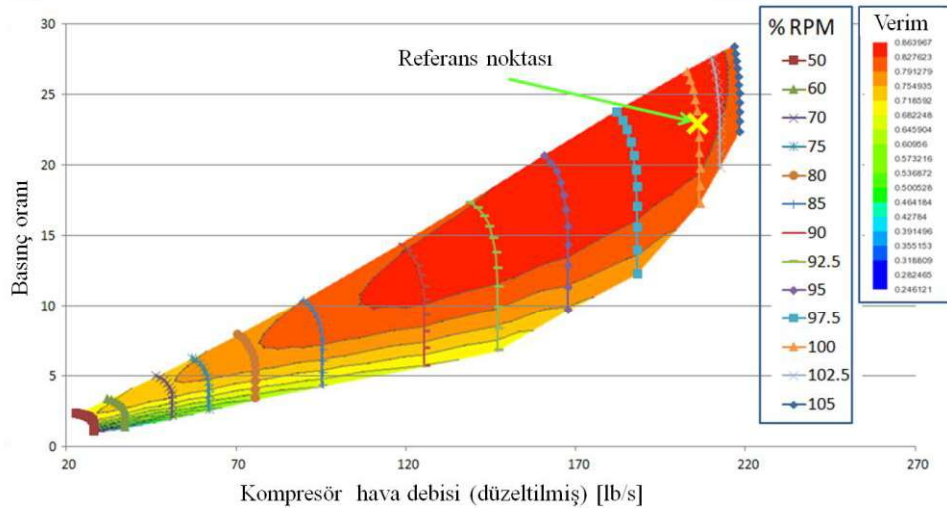
Şekil 6.4. Durum 3 Optimizasyon parametrelerinin iterasyon adımları ile değişimi

Tablo 6.11. Durum 3 Optimizasyon sonuçlarının karşılaştırılması

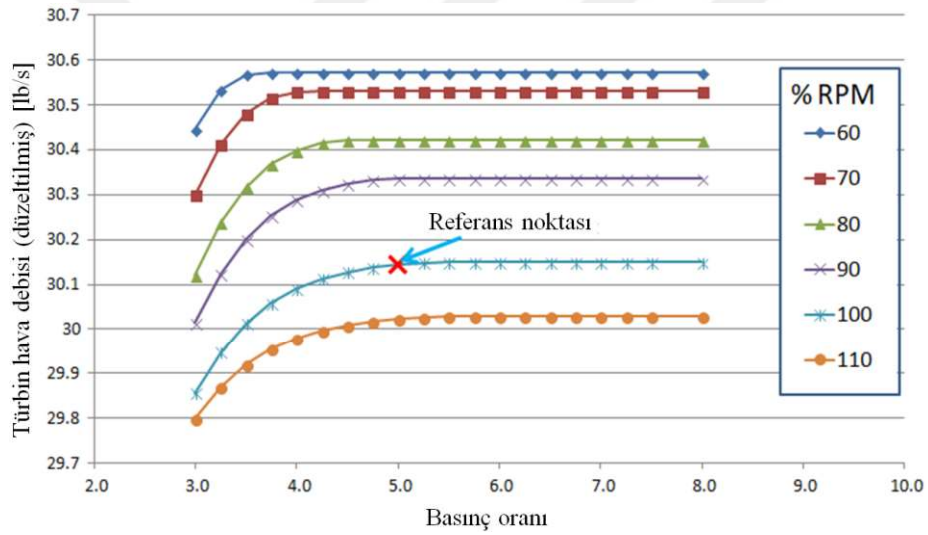
Parametre	Homaifar [60]	Bu tez
BPR	5,5	6
M0	0,78	0,693
CPR	16,5	25,0
FPR	1,5	1,372
TMA	850	912
Toplam verim	0,21	0,265

6.2. Tasarım Noktası ve Tasarım Noktası Dışı Çevrim Optimizasyonu

Tasarım noktasında optimizasyon yapılması için bileşenlerin haritalarına ihtiyaç yoktur. Teknolojik olarak kullanılabilecek bileşenlerin seviyelerinin tahmin edilmesi yeterlidir. Tasarım dışında optimizasyon yapılabilmesi için ise bileşen haritalarına ihtiyaç bulunmaktadır. Tasarım noktasında dışında bileşenler uyumlu şekilde çalışmadıkları zaman bir hata oluşturacaklardır. Motorun herhangi bir çalışma noktasında kararlı şekilde çalışabilmesi için oluşan hatanın kabul edilebilir bir seviyenin altında olması gereklidir.



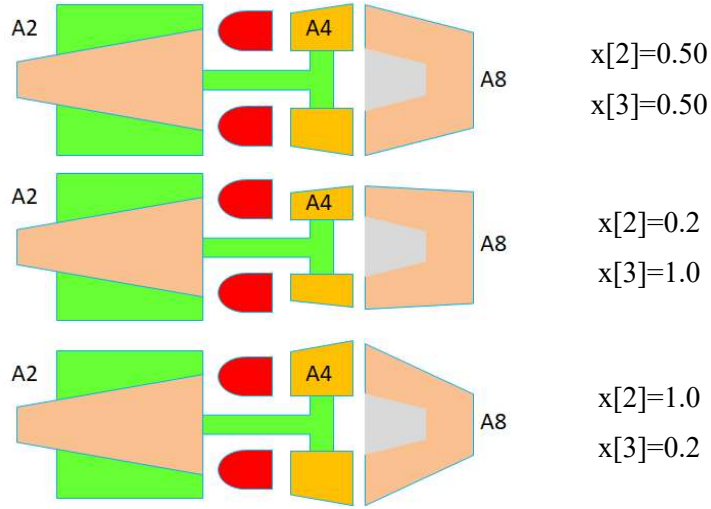
Şekil 6.6. NPSS örneğine ait kompresör haritası [75]



Şekil 6.7. NPSS örneğine ait türbin haritası [75]

Örnek olarak verilen turbojet motoru hesaplamaları kullanılarak tasarım noktasındaki motor termodinamik modeli için öncelikle el hesaplamaları yapılmıştır. Buna göre turbojet fonksiyonu içerisinde parametreler girilmiştir. Verilen kompresör ve türbin haritaları bunu destekleyecek şekilde ölçeklendirilmiştir.

Kompresör giriş alanı sabit tutulmuştur. Türbin ve egzoz alanları parametrik olarak değiştirilmiştir. Parametreler 0,2 ve 1,0 arasında değiştirilerek, türbin büyüklüğü ve egzoz alanı için optimum bir bölge olup olmadığı araştırılmıştır (Şekil 6.8). Parametrelerin 0,5 değeri, el hesaplamalarında kullanılan alanlara karşılık gelmektedir. A4 alanı yanma odası çıkışı alanını ifade etmektedir, aynı zamanda referans alınan türbinin kapasitesini ve büyüklüğünü belirlemektedir. A8 alanı egzoz çıkış alanına karşılık gelmektedir.



Şekil 6.8. Türbin ve egzoz alanı parametrik değişimi

Parametrik çalışma sonucunda elde edilen sonuçlar Tablo 6.12’de özetlenmiştir.

Tablo 6.12. A4 ve A8 parametrik analiz sonuçları özet tablosu

FN [N]						T4 [K]						ERRx10 ⁴					
x[2]/x[3]	0.2	0.4	0.6	0.8	1	x[2]/x[3]	0.2	0.4	0.6	0.8	1	x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	28107	30239	32398	35794	36410	0.2	1111.1	1115.4	1130.1	1193.2	1131.6	0.2	2422.8	429.6	97.733	97.733	97.733
0.4	28428	29471	46235	46103	43717	0.4	1130.8	1115.4	1760	1674.4	1560	0.4	2954.6	1810.3	633.83	0	0
0.6	27481	43641	46432	48302	44537	0.6	1111.1	1760	1760	1760	1607.6	0.6	3911.6	2391.1	1072.3	60.362	0
0.8	40968	43562	45263	46880	46673	0.8	1760	1760	1760	1760	1718.1	0.8	4142.3	2839.4	1769.4	666.91	0
1	40900	42824	44499	45912	48049	1	1760	1760	1760	1760	1760	1	4441.6	3265.9	2156.9	1399	1399

(a) İtki

W2 [kg/s]					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	44.841	44.862	45.058	44.851	45.055
0.4	45.07	45.506	45.052	45.388	45.503
0.6	45.592	45.231	45.589	45.678	45.684
0.8	45.379	45.646	45.684	45.684	45.684
1	45.648	45.684	45.684	45.684	45.684

(b) T4

CPR					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	23.192	23.14	22.65	23.167	22.663
0.4	22.613	20.673	22.671	21.392	20.7
0.6	19.923	22.088	19.956	18.167	17.261
0.8	21.445	19.179	17.274	16.214	15.1
1	19.112	17.274	16.103	15.1	15.1

(c) Oluşan hata

TPR					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	4.5	4.5	4.5	4.5	4.5
0.4	4.5	4.5	3.682	3.7841	4.1632
0.6	4.5	3.5877	3.3201	3.1545	3.5241
0.8	3.4974	3.242	3.0844	3.0087	3.0314
1	3.2362	3.0844	3.0017	2.9355	2.9355

(d) W2

Wfuel [g/s]					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	443.56	450.7	482.4	558.38	484.15
0.4	484.06	496.71	1370.1	1278.2	1127.4
0.6	501.52	1385.3	1425.9	1448.6	1239
0.8	1399.5	1436.9	1457.6	1467.5	1417.1
1	1437.7	1457.6	1468.4	1477.3	1477.3

(e) CPR

Compressor Efficiency					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	0.845	0.8454	0.8486	0.8452	0.8485
0.4	0.8487	0.8514	0.8485	0.8519	0.8515
0.6	0.8497	0.8509	0.8498	0.8389	0.8303
0.8	0.8519	0.8463	0.8304	0.8181	0.8024
1	0.8458	0.8304	0.8165	0.8024	0.8024

(f) TPR

Turbine Efficiency					
x[2]/x[3]	0.2	0.4	0.6	0.8	1
0.2	0.8982	0.8982	0.8982	0.8982	0.8982
0.4	0.8982	0.8982	0.9129	0.9113	0.9054
0.6	0.8982	0.9146	0.9191	0.9201	0.9157
0.8	0.9161	0.9202	0.9201	0.92	0.92
1	0.9202	0.9201	0.92	0.9199	0.9199

(g) Yakıt debisi

(h) Kompresör verimi

(i) Türbin verimi

Tabloya göre motorun isterleri sağladığı ve kararlı çalışabildiği bir bölge bulunmaktadır. Her bir alt tablo içerisindeki sonuçların yorumları aşağıda maddeler halinde verilmiştir;

- a) Tablo 6.12. (a): Sol üst köşede itki çok düşüktür. Birçok noktada itki sağlanamamaktadır. İtkinin fazla olduğu yerlerde T_4 çok yüksektir.
- b) Tablo 6.12. (b): Alt orta bölümde T_4 limit değerdedir.
- c) Tablo 6.12. (c): Bileşenlerin uyum hatası tablonun sağ tarafında minimum olmaktadır. Motorun bu bölgede kararlı çalışacağı söylenebilir.
- d) Tablo 6.12. (d): Sağ alt bölgede, kompresör hava debisi çok artmıştır ve kompresör boğulmuştur.
- e) Tablo 6.12. (e): Sol üst köşede kompresör stall çizgisine çok yakındır. Ve sağ alt tarafta boğulma debisine ulaşmıştır.
- f) Tablo 6.12. (f): En üst sırada ve sol üst köşede türbin maksimum basınç oranına ulaşmıştır.
- g) Tablo 6.12. (g): Yeşil olan bölgede yakıt debisi çok düşüktür. Bu gerçekçi olmayan bir durumu gösteriyor olabilir. Alt sıralarda ise yüksek yakıt sarfiyatı görülmektedir.
- h) Tablo 6.12. (h): Kompresör verimi sağ alt köşede düşüktür.
- i) Tablo 6.12. (i): Türbin verimi yeşil bölgede en yüksektir.

NPSS örneğinde verilen motorun geometrik olarak bilgileri olmadığından, teorik olarak alan hesaplamaları yapılmıştır. Tasarım noktasında ve tasarım noktası dışında en uygun A4 ve A8 alanlarını bulmak amacıyla optimizasyon problemi aşağıdaki şekilde tanımlanmıştır.

$$Uyum\ hatası = \max \left(\frac{(W_4)_{hesaplanan} - (W_4)_{harita}}{(W_4)_{harita}} \cdot 100, \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{eğri}} \cdot 100 \right) \quad (6.7)$$

$$Tasarım\ hedefi = (1 - \eta_{kompresör}) \cdot 100 \quad (6.8)$$

Tablo 6.13. Optimizasyon tanımlama denklemleri

Minimize,	:	$f(x) = Toplam_{hata} = Uyum\ hatası\ terimi + Tasarım\ hedefi\ terimi$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3\}^T, x_i \in [0,1], i = 0,1,2,3$
Parametreler	:	$Beta = x_0, W_2 = Map(Beta, N), CPR = Map(Beta, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_{4ratio} = 0,5 + x_2$ $A_8 = A_8 \cdot (0,5 + x_3)$
Kısıt fonksiyonları	:	$g_{1(x)} = (\dot{itki})_{tasarım} - (\dot{itki})_{hesaplanan} \leq 0$ $g_{2(x)} = (W_4)_{hesaplanan} - (W_4)_{harita} \leq 0$ $g_{3(x)} = (W_8)_{hesaplanan} - (W_8)_{egzoz\ eğrisi} \leq 0$ $g_{4(x)} = \dot{İş}_{kompresör} - \eta_{mekanik} \cdot \dot{İş}_{turbın} \leq 0$

Optimizasyon sonunda, türbin ve egzoz alanlarını belirleyen parametreler, $x[2] = 0,485344$ ve $x[3] = 0,741851$ olarak bulunmuştur. El hesabında bulunan alanlar ile optimizasyon sonrası bulunan alanlar tablo Tablo 6.14'te gösterilmiştir.

Tablo 6.14. El hesabı ile optimizasyon sonucu karşılaştırması

Alan	El hesabı	Optimizasyon sonucu	Aradaki Fark %
$A4 [mm^2]$	25113,5	24745,4	-1,5
$A8 [mm^2]$	69478,8	78180,2	12,5

NPSS örneğinde verilen performans verileri ile el hesaplamaları ve optimize edilmiş alanlar ile yapılan performans hesapları tablo Tablo 6.15'te karşılaştırılmıştır.

Tablo 6.15. NPSS, el hesapları ve optimize edilmiş alanlarla bulunan sonuçlar

	NPSS- SI birimi		El hesabı		Optimum/Eşleştirilmiş	
	SLS	M=1, Z=6096 m	SLS	M=1, Z=6096 m	SLS	M=1, Z=6096 m
N	100	100	100	100	100	100
P0	103,393	89,969	101,325	88,141	101,325	88,138
P2	101,324	88,168	101,325	86,378	101,325	88,138
P3	2026,499	1764,598	2026,500	1727,559	2023,460	1762,750
P4	1925,175	1676,374	1925,175	1641,181	1922,290	1674,610
P5	630,346	549,819	630,493	570,404	565,858	475,929
T0	288,150	298,311	288,150	248,526	288,150	298,219
T1	288,150	298,311	288,150	298,230	288,150	298,219
T3	730,994	755,106	745,913	773,177	747,648	773,722
T4	1702,783	1760,106	1702,783	1760,106	1733,870	1760,000
T5	1360,556	1409,922	1370,962	1418,831	1360,610	1372,710
W2	45,359	38,796	45,359	39,339	45,588	38,978
Wfuel	1,361	1,218	1,364	1,465	1,388	1,190
Fnet	47411,1	32080,0	47655,1	32323,8	47450,3	31169,9
CPR	20,00	20,01	20,00	20,00	19,97	20,00
TPR	3,054	3,054	3,053	2,877	3,397	3,519
NPR	6,036	11,456	6,222	12,250	5,585	10,221

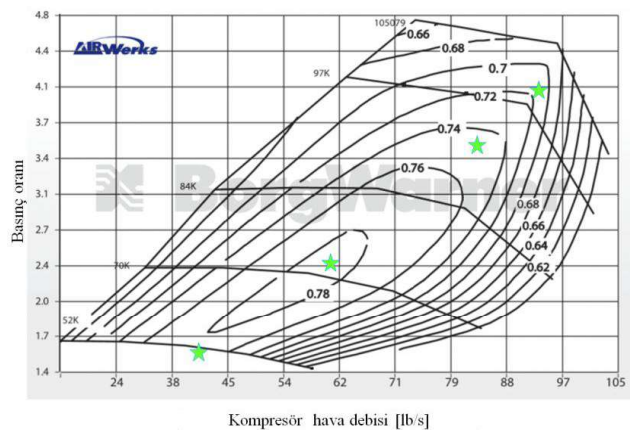
6.2.2. Jetcat P400 turbojet modellemesi

(Grannan, Hoke, McClearn, Litke, & Schauer, 2017) JetCat motorlarının test ölçümleri ile performans hesaplamalarını karşılaştırmışlardır [76]. JetCat motorlarının bileşenleri hakkında ellerinde bilgi olmadığından, turboşarjör kompresörlerin haritalarını kullanmışlardır. Üç motor tipinin performans hesabında kullanmak amacıyla, ölçüleri en yakın olan turbocharger kompresörleri seçilmiş ve buna göre yapılan hesaplamalar ile motor testinden elde edilen değerler karşılaştırılmıştır. (JetCat, 2021) motor üreticisi tarafından belirtilen değerler Tablo 6.16 da verilmiştir [77].

Tablo 6.16. Jetcat motoru üreticisi tarafından yayınlanmış veriler [76,77]

Performans parametresi	P80-SE	P200-SX	P400-RX-G
Maksimum devir [d/d]	125,000	112,000	98,000
Maksimum itki [N]	97	230	395
Yakıt debisi [kg/s]	16,32	35,04	62,40
Özgül yakıt tüketimi [kg/(N-sa)]	0,168	0,152	0,158
İtki-ağırlık oranı [N/N]	6,8	9,2	11,4
Ağırlık (starter dahil) [kg]	1,45	2,56	3,55
Motor çapı [mm]	112	132	178,6
Hava debisi [kg/s]	0,25	0,45	0,67
Basınç oranı	2,4	3,7	3,8
Egzoz gaz sıcaklığı aralığı [C]	510-700	480-750	480-750

JetCat motoru hobi amaçlı uçaklarda kullanılan gerçek bir turbojet motordur. Turboşarjörlerde kullanılan salyangoz yapısı ile bu motordaki difüzör yapısı farklıdır. Difüzör ile birlikte kullanıldığında santrifüj kompresör haritasında az miktar değişiklik olması beklenmektedir. Şekil 6.9’da görüldüğü gibi kompresörün en verimli olduğu çalışma noktası 70000 d/d civarındadır (BorgWarner, 2021).



Şekil 6.9. S400SX3 kompresör haritası üzerinde çalışma noktaları [76,78]

Maksimum itki bölgesine çıkıldığında verim düşmektedir. Genel olarak makalede bulunan değerler ile JetCat üreticisi tarafından yayınlanan veriler uyumlu olmakla birlikte bazı noktalarda farklılıklar bulunmuştur. Örneğin P400 motorunun maksimum itki noktasında verilen basınç oranı 3,8 iken referans çalışmada [76] basınç oranı 4,15 olarak bulunmuştur.

Tablo 6.17. P400-RX-G Katalog bilgileri [78]

Parametre adı	Değeri
Maksimum itki [N]	395
Rölanti itkisi [N]	13
Kompresör çapı [mm]	147
Toplam boyu [mm]	350
Devir [d/d]	30-98 x1000
Egzoz gaz sıcaklığı [°C]	< 750
Yakıt sarfıyatı [ml/d]	1300
Yakıt tipi	Jet A1, Kerosen

İtkinin debi ile ikinci dereceden bağlantılı olduğu kabul edilerek ara devirlerdeki itki değerleri hesaplanmıştır. JetCat motorunda kullanılan diğer bileşenlerin özellikleri bilinmediğinden dolayı bazı kabuller ve varsayımlar yapılmıştır.

Tablo 6.18. Referans çalışmada sunulan test verisi [76]

Devir [d/d]	Devir [%]	H.Debisi [kg/s]	PR	Verim	İtki [N]
29731	30,5	0,16	1,20	0,84	14,0
49391	50,7	0,30	1,61	0,83	62,1
69324	71,1	0,46	2,39	0,80	157,9
84341	86,6	0,60	3,21	0,76	277,5
89434	91,8	0,65	3,51	0,75	328,4
94395	96,9	0,68	3,86	0,73	360,9
97437	100,0	0,71	4,12	0,72	395,0

Tablo 6.19. Çok hedefli tasarım gereksinimleri

Parametre adı	Rölanti Çalışması	Maksimum kalkış
Motor devri [%]	30	100
İrtifa [m]	0	0
M0	0	0
İtki isteri [N]	14	395
Max T5 [K]	753	1023
Hedefler	İtki, T4 sınırı	İtki, SFC T4 sınırı
Yakıt debisi [g/s]	2,667	17,333

Hedef fonksiyonu içerisinde aşağıdaki uyum hatası ve tasarım hedefi seçilmiştir;

$$Uyum\ hatası = \max \left(\frac{(W_4)_{hesaplanan} - (W_4)_{harita}}{(W_4)_{harita}} \cdot 100, \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{eğri}} \cdot 100 \right) \quad (6.9)$$

$$Tasarım\ hedefi = (1 - \eta_{kompresör}) \cdot 100 \quad (6.10)$$

Tablo 6.20. Optimizasyon tanımlama denklemleri

Minimize,	:	$f(x) = Toplam_{hata} = Uyum\ hatası\ terimi + Tasarım\ hedefi\ terimi$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3\}^T, x_i \in [0,1], i = 0,1,2,3$
Parametreler	:	$Beta = x_0, W_2 = Map(Beta, N), CPR = Map(Beta, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_{4ratio} = 0,5 + x_2$ $A_8 = A_8 \cdot (0,5 + x_3)$
Kısıt fonksiyonları	:	$g_{1(x)} = (\dot{itki})_{tasarım} - (\dot{itki})_{hesaplanan} \leq 0$ $g_{2(x)} = (W_4)_{hesaplanan} - (W_4)_{harita} \leq 0$

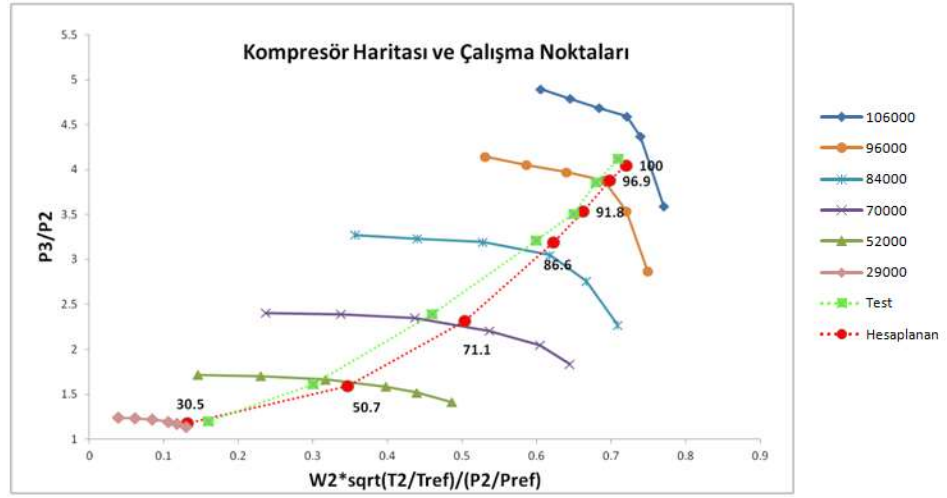
Tablo 6.20. Optimizasyon tanımlama denklemleri (devam)

Kısıt fonksiyonları	:	$g_{3(x)} = (W_8)_{hesaplanan} - (W_8)_{egzozegrisi} \leq 0$ $g_{4(x)} = \dot{İ}_{şkompresör} - \eta_{mekanik} \cdot \dot{İ}_{şturbin} \leq 0$
---------------------	---	---

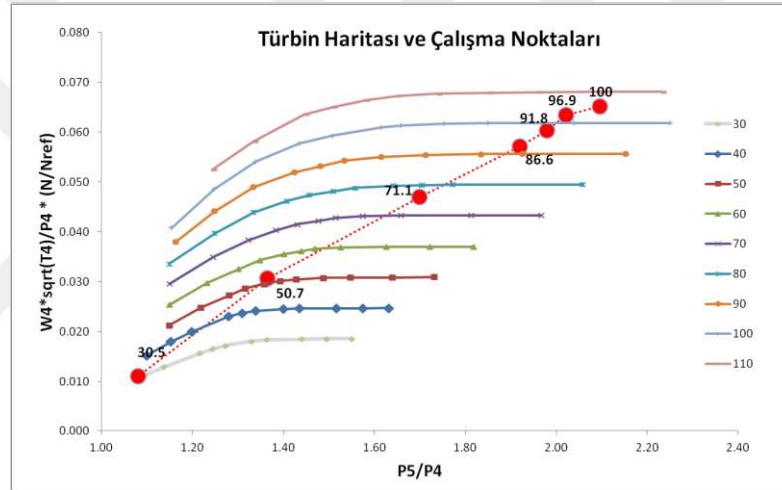
Tablo 6.21. El hesabı ile bulunan ve optimizasyon ile elde edilen alanların kıyaslaması

Parametre	El hesabı	Optimum	Fark %
A4 [mm ²]	1600,2	1668,7	4,3
A8 [mm ²]	2892,6	3315,7	14,6

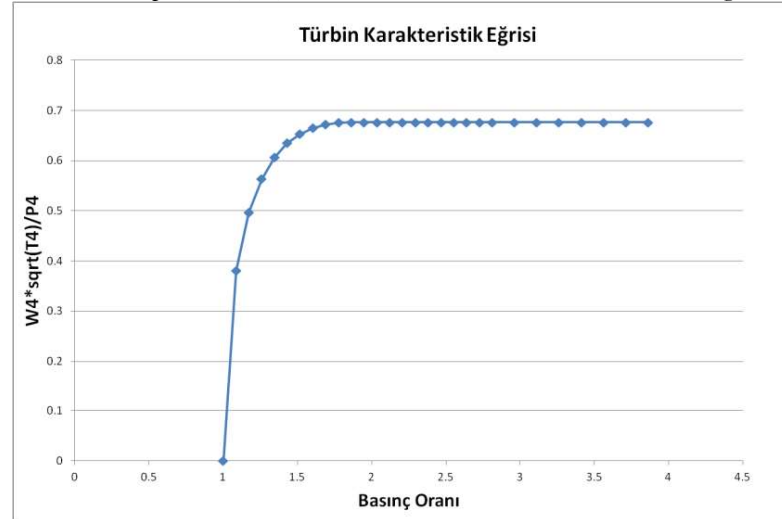
Şekil 6.10’da, optimize edilmiş motor modeli ile hesaplanan debi ve basınç oranları ve testte ölçülen değerler, kompresör çalışma eğrileri üzerinde gösterilmiştir. Sonuçların oldukça uyumlu olduğu görülmüştür.



Şekil 6.10. Hesaplanan değerler ile test verilerinin karşılaştırması



Şekil 6.11. Hesaplanan çalışma noktalarının türbin haritası üzerinde gösterimi



Şekil 6.12. Genel amaçlı türbin eğrisi

6.3. Motor Varyantlarının Tasarımı

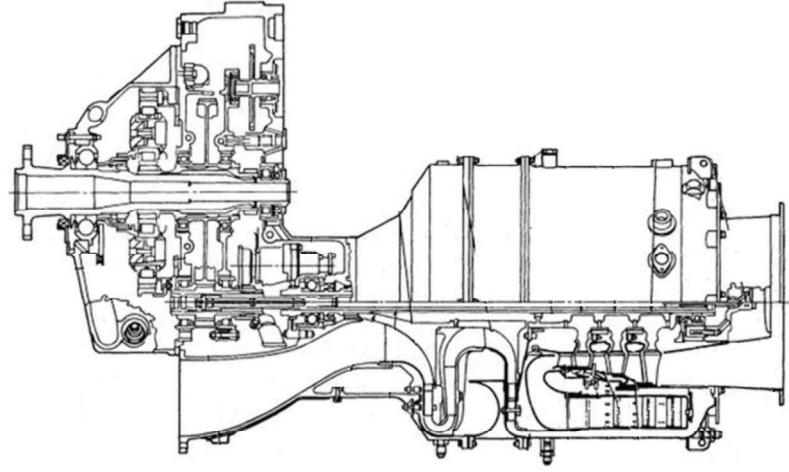
Uçak motorları incelendiğinde, aynı üreticiler tarafından geliştirilen motorların çekirdek yapılarının birbirine benzediği görülmektedir. Motor üreticileri, tamamen sıfırdan yeni bir motor tasarlamak yerine mevcut motorların tasarımlarını değiştirerek farklı güçlerde motorlar geliştirmeyi tercih etmektedirler. Özellikle başarılı bir çekirdek motor geliştirildiğinde bu çekirdek kullanılarak turboprop, turboşaft ve turbofan versiyonlarında yeni motorlar geliştirilmektedir. Literatürde bunlara ait birçok örnek bulunmaktadır. Tasarımı kanıtlanmış bir çekirdek kullanılması motor tasarım süresini kısaltmakta ve geliştirme maliyetlerini azaltmaktadır.

TPE-331 (Şekil 6.13), LHTEC-T800 (Şekil 6.14) ve TFE-109 (Şekil 6.15) motorları aynı kompresör yapısına sahip farklı mimarilerdeki motorlardır. Genel özellikleri Tablo 6.22’de verilmiştir. TPE-331 ve LHTEC-T800 uzun yıllardır kullanılmakta olan ticarî motorlardır. Bu nedenle farklı güçlerde birçok türevleri bulunmaktadır ve literatürde farklı karakteristik özellikler verilebilmektedir. TFE109 motoru ise ticarî olarak kullanılmayan bir motor olduğundan literatürde detay bilgisi az bulunan bir motordur.

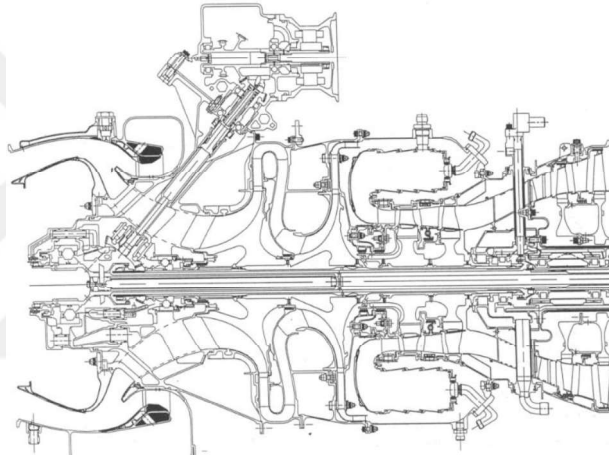
TPE-331 ve LHTEC-T800 motorlarında yanma odası çıkışındaki gaz basıncı ve sıcaklığı tamamıyla şaft gücü üretilmek amacıyla kullanılmaktadır. Bu nedenle egzoz tasarımı, turbojet motorlara göre farklılık göstermektedir ve egzozda oluşan itki seviyesi düşük olmaktadır.

Tablo 6.22. Motor karşılaştırma tablosu

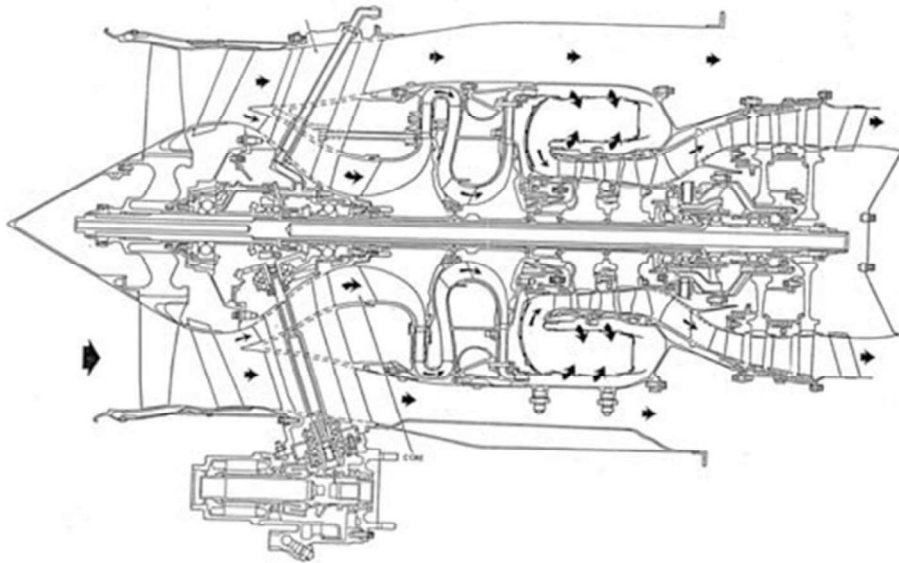
Motor modeli	Hava Debisi [kg/s]	Basınç oranı	T4 [K]	Türbin yapısı	Güç/ İtki	Motor tipi	Üretici
TPE-331	3,49	10,2	1341	3 kademe	953 SHP	Sabit şaftlı turboprop	Garrett/ Honeywell
LHTEC-T800	3,35	15	≥ 1341	2 HPT 2 LPT	1300-1563 SHP	Serbest türbinli turboşaft	Honeywell /RR
TFE-109	4,83 (3,17düz.)	1,6/14,4	≥ 1341	2 HPT 2 LPT	1330-1500 lbf	İki şaftlı turbofan	Garrett



Şekil 6.13. Tek şaftlı turboprop motor kesiti (Garrett Service Manual, 2021)[79]



Şekil 6.14. Serbest türbinli turboşaft motor kesiti (Becker & Frounfeld, 1992) [80]



Şekil 6.15. İki şaftlı ayırık akışlı turbofan motor kesiti
(Davis, Beale, Lt. Col. Boyer, & Maj. O'Dowd, 2006) [81]

LPT tarafından çekilebilecek enerji miktarı; hava debisi, türbin gaz sıcaklığı ve basınç oranına bağlı olarak değişmektedir. Kompresörde elde edilen basınç oranı önemli bir faktördür. LPT sonrası basınç ile atmosfer basıncı oranı aşağıdaki şekilde yazılabilir;

$$\frac{P_5}{P_0} = \left(\frac{P_5}{P_{45}}\right) \cdot \left(\frac{P_{45}}{P_4}\right) \cdot \left(\frac{P_4}{P_3}\right) \cdot \left(\frac{P_3}{P_2}\right) \cdot \left(\frac{P_2}{P_1}\right) \cdot \left(\frac{P_1}{P_0}\right) \quad (6.11)$$

$$P_5 = P_8 \approx P_0 \quad (6.12)$$

$$1 \cong \left(\frac{P_5}{P_{45}}\right) \cdot \left(\frac{P_{45}}{P_4}\right) \cdot \left(\frac{P_4}{P_3}\right) \cdot \left(\frac{P_3}{P_2}\right) \cdot \left(\frac{P_2}{P_1}\right) \cdot \left(\frac{P_1}{P_0}\right) \quad (6.13)$$

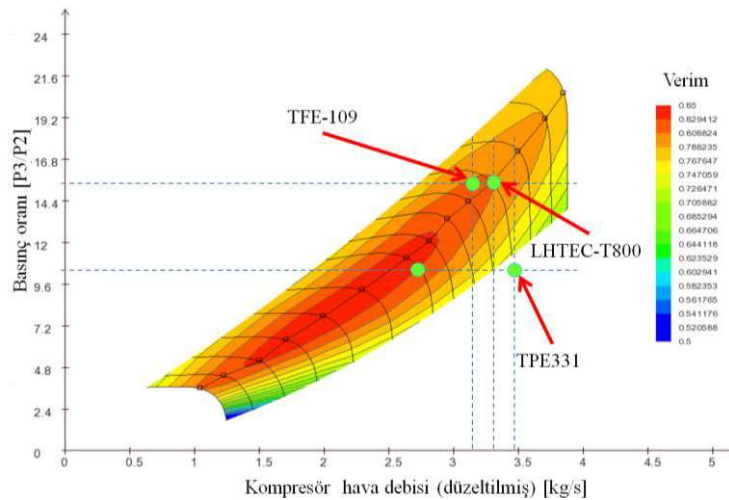
$$1 \cong \left(\frac{1}{PR_{LPT}}\right) \cdot \left(\frac{1}{PR_{HPT}}\right) \cdot (PR_{COMB}) \cdot (CPR) \cdot (PR_{DUCT}) \cdot (PR_{RAM}) \quad (6.14)$$

Basınç artışı ve genleşmeler eşitliğin sol ve sağ tarafından toplanırsa;

$$(CPR) \cdot (PR_{RAM}) \cong (PR_{LPT}) \cdot (PR_{HPT}) \cdot \left(\frac{1}{PR_{COMB}}\right) \cdot \left(\frac{1}{PR_{DUCT}}\right) \quad (6.15)$$

Buradan, inlet ve kompresörde elde edilen basınç artışının, yaklaşık olarak, LPT ve HPT tarafından paylaşıldığı görülmektedir. Dolayısı ile kompresör basınç oranı çok kritik bir rol oynamaktadır.

TPE-331, LHTEC-T800 ve TFE-109 basınç oranı ve düzeltilmiş debileri LHTEC-T800 kompresör haritası Şekil 6.16'da gösterilmiştir. TPE-331 motorunun basınç oranının diğer iki motora göre daha düşük olduğu ve haritanın dışında kaldığı görülmektedir. Bu nedenle tüm hesaplamalarda TPE-331'e göre ölçeklenmiş olan kompresör haritası kullanılmıştır.



Şekil 6.16. İki kademeli santrifüj kompresör performans haritası [81]

LHTEC-T800 turboşaft motoru tarihsel olarak TPE-331 ve TFE-109 motorlarından sonra geliştirilmiştir bu nedenle de daha iyi bir kompresöre sahip olduğu bilinmektedir. Bu turboşaft motorun genel özellikleri Tablo 6.23'te verilmiştir.

Tablo 6.23. T800-LHT-800 teknik özellikleri [81]

Parametre	Açıklaması
Kompresör	2 kademe santrifüj
Türbin HPT	2 kademe eksenel
Güç Türbini LPT	2 kademe eksenel
Yanma odası	Halka şeklinde
Hava debisi [kg/s]	3,3 kg/s
Basınç oranı	14:1
Kompresör verimi	>0,8
Surge magjin	>%15
Ağırlık	136 kg
Şaft gücü	895 kW (1200 shp)
Dönüş hızı	43800 d/d

(Palmer & Waterman, 1994) bu motorda kullanılan iki kademeli santrifüj kompresörün tasarımını anlatmıştır [82]. Buna göre verimliliğini arttıran faktörler şunlardır;

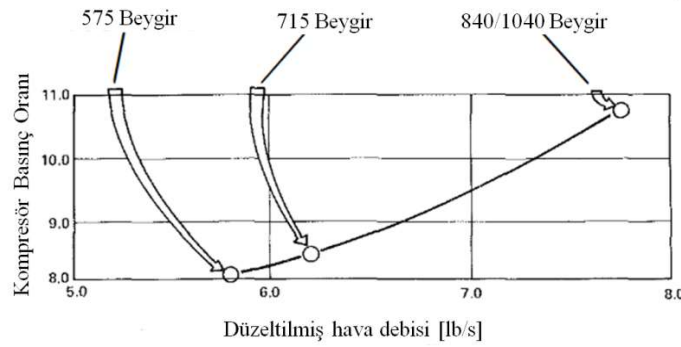
- Hava giriş kanatçıkları kök/uç oranı (0,51) kullanılarak kanatçık sayısının artırılması
- Girişteki mach sayısı düşürülmesi ($M_2=0,52$)
- Her iki impeller çıkışında backsweep açısının (47°) artırılması
- Yanma odasına hava giriş hızının düşürülmesi ($M_3=0,15$) (Palmer & Waterman, 1994)

Bu tezde geliştirilen optimizasyon kodu ve performans hesaplama kodları kullanılarak yukarıda belirtilen motor mimarîlerine ait termodinamik hesaplamalar yapılmış ve kompresör, türbin ve egzoz alanları boyutlandırılmıştır. Tasarım optimizasyonu açısından aşağıdaki konular ilerideki bölümlerde detaylı olarak ele alınmıştır;

- Tek şaftlı sabit bir turboprop motorun tasarım dışı noktalarında performans hesaplaması yapılması.
- Aynı motorun serbest türbinli bir turboşaft motora dönüştürülmesi için gerekli yüksek basınç ve düşük basınç türbinlerinin boyutlandırılması ve iki motorun performanslarının karşılaştırılması.
- Serbest türbinli turboşaft motorun, iki şaftlı ayrıklı akışlı bir turbofan motoruna dönüştürülmesi için en yüksek itkiyi verecek fan konfigürasyonunun bulunması.
- Bulunan konfigürasyona göre boyutlandırılacak bir turbofan motorunun turboşaft motor ile benzer geometrilerde ve koşullarda üreteceği itki ve performans parametrelerinin hesaplanması.

6.4. Tek Şaftlı Turboprop Motorunun Modellemesi

Temel olarak ele alınan turboprop motor, iki kademe santrifüj kompresöre ve üç kademeli eksenel türbine sahip bir turboprop motordur. Türbinin ürettiği tüm güç ana şaft üzerinden paylaştırılır; bir kısmı kompresörü çevirmek için diğer kısmı ise pervane dişli kutusu çevirmek için kullanılır. Hafifliği, düşük yakıt tüketimi, bakım kolaylığı, kolay kullanımı gibi özellikleri nedeni ile birçok uçakta tercih edilen bir motordur. İlk defa 1963 yılında üretilen bu motorda zaman içerisinde tasarımsal olarak iyileştirmeler yapılmış ve birçok farklı güç seviyesine sahip farklı versiyonları üretilmiştir. Motor şaft gücünün artırılması için kompresör basınç oranı, hava debisi ve yanma odası çıkış sıcaklığının artırılması gerekmektedir. Bu tezde incelenen motor modeli kalkış noktasında yaklaşık olarak 953 şaft beygir gücü vermektedir (Tablo 6.24).



Şekil 6.17. Örnek turboprop motorun farklı güçteki modelleri (Frignac & Privoznik, 1979) [83]

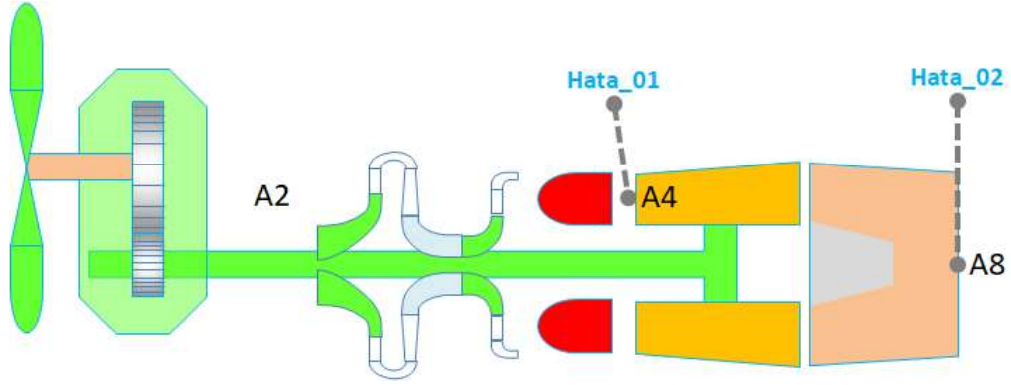
Kompresör gücü ile türbin gücü arasındaki bağlantı şu şekilde ifade edilmektedir;

$$\text{Kompresör gücü} = W_2 * (T_3 - T_2) * c_{p_{hava}} \quad (6.16)$$

$$\text{Türbin gücü} = \frac{\text{Kompresör gücü} + \text{Pervane şaft gücü} + \text{Aksesuar gücü}}{\text{mekanik verim}} \quad (6.17)$$

Tablo 6.24. Tek şaftlı turboprop motor performans tablosu (Ghorbanian, et al., 2019) (AIAA, 2021)[84,85]

	Çalışma koşulu	İrtifa [m]	Mach	W25[kg/s]	Basınç Oranı	Şaft beygir gücü [shp]	T4 [K]
1)	Kalkış	0,0	0	3,493	10,55	953,1	1342,4
2)	Seyir	3810,0	0,3478	2,663	11,6	690,0	1302,3
3)	Dolanma	2133,6	0,294	2,461	8,61	603,0	1248,4



Şekil 6.18. Tek şaftlı turboprop bir motorun şematik gösterimi

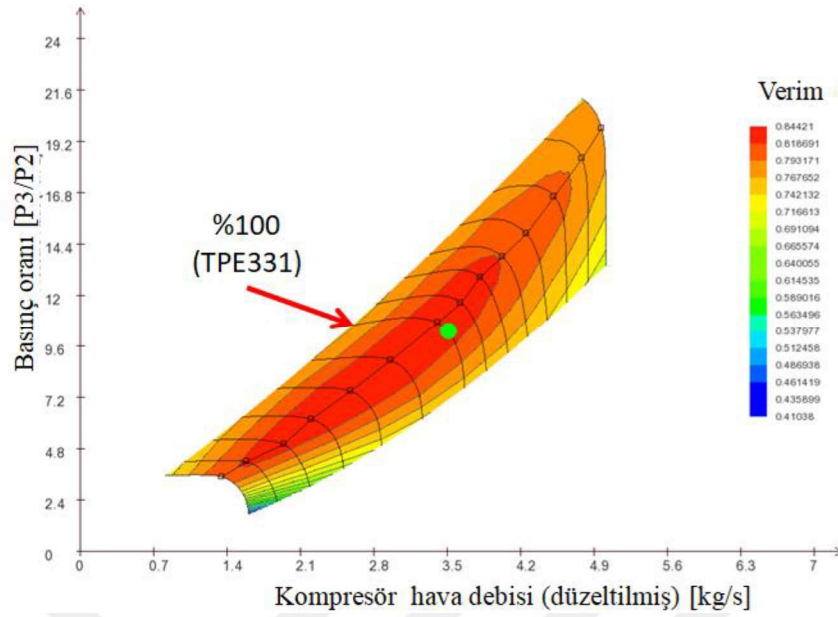
Performans hesaplamaları ve optimizasyonda izlenen adımlar şu şekildedir;

- El hesapları ile motorun %100 motor devrindeki performans verileri kullanılarak A_4 , A_8 alanları kabaca boyutlandırılır.
- Optimizasyon kodu içerisinde kompresör haritası ölçeklendirilerek kaydedilir.
- Optimizasyon kodu ile tasarım noktasında NPR=1,12 yi sağlayacak şekilde A_4 ve A_8 alanları bulunur.
- Optimizasyon kodunda tasarım dışı noktalarda hesaplama yapılarak test verileri ile karşılaştırılmıştır. Eğer sonuçlarda bir uyumsuzluk varsa tasarım modunda çözüm alınması adımı tekrar edilir.

Serbest türbinli turboşaft motorunun kompresör haritasında %90 devire karşılık gelen nokta tek şaftlı turboprop motorun %100 çizgisindeki tasarım noktasına karşılık olarak alınmıştır. Buna göre ölçeklenen harita ve seçilen tasarım noktası Şekil 6.19'da gösterilmiştir. Türbin boğulma basınç oranı 4.5 olacak şekilde bir türbin eğrisi kullanılmıştır. Egzozdan itki isteri olmadığından egzoz çıkış alanı A_8 yeterince büyük yapılmıştır. Optimizasyonda bu değer tasarım noktasını sağlayacak şekilde seçilmiştir.

Tablo 6.25. Ölçekleme noktaları

Referans noktalar	W25 [kg/s]	P3/P2	Devir [%]
Orijinal nokta	2,703	10,59	90,0
Ölçekleme sonrası	3,49	10,55	100,0



Şekil 6.19. Ölçeklenmiş kompresör haritası

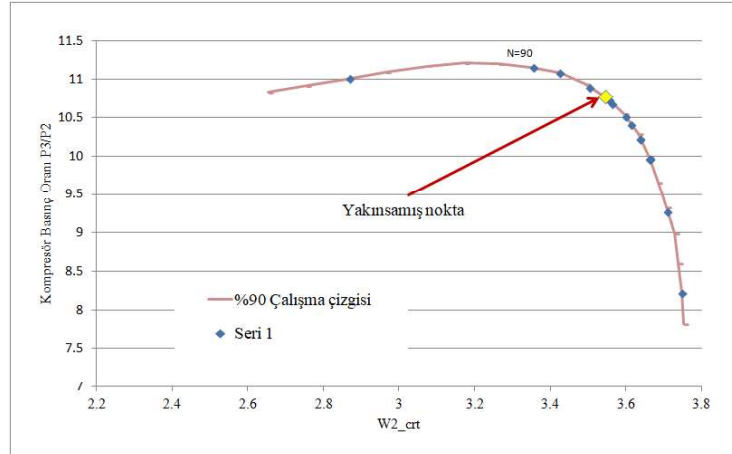
Optimizasyonda kullanılan denklemler Tablo 6.26’da gösterilmiştir;

Tablo 6.26. Optimizasyon tanımlama denklemleri

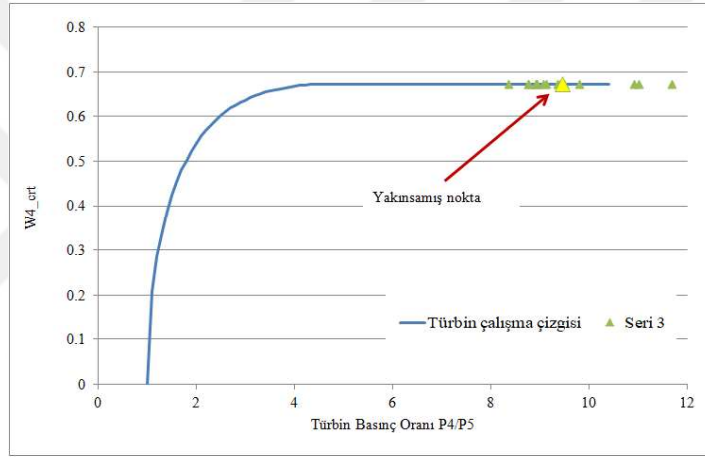
Minimize,	:	$f(x) = Toplam_{hata} = Uyum\ hatası\ terimi + Tasarım\ hedefi\ terimi$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3\}^T, x_i \in [0,1], i = 0,1,2,3$
Parametreler	:	$Beta = x_0, W_2 = Map(Beta, N), CPR = Map(Beta, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_4 = A_{4_0} \cdot (0,5 + x_2)$ $A_8 = A_{8_0} \cdot (0,5 + x_3)$
Hata değerleri	:	$Hata_{01} = \frac{(W_4)_{hesaplanan} - (W_4)_{eğri}}{(W_4)_{hesaplanan}} \cdot 100$ $Hata_{02} = \frac{(W_{45})_{hesaplanan} - (W_{45})_{eğri}}{(W_{45})_{hesaplanan}} \cdot 100$ $Hata_{03} = \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{hesaplanan}} \cdot 100$
Tasarım hedefi	:	$Hata_{03} = \frac{1 - Ecomp}{Ecomp}$

Optimizasyon iterasyonları sırasında ziyaret edilen noktalar ve yakınsamış olan nokta, kompresör ve türbin eğrileri üzerinde gösterilmiştir (Şekil 6.20, Şekil 6.21).

Motorun kalkış, seyir ve son yaklaşma sırasındaki “tasarım-dışı” performans hesaplamaları ile motor testlerine göre verilen referans değerler Tablo 6.27’de karşılaştırılmıştır.



Şekil 6.20. Kompresör çalışma çizgisi ve iterasyon noktaları



Şekil 6.21. Türbin çalışma çizgisi ve iterasyon noktaları

Tablo 6.27. Hesaplanan ve verilen çalışma noktası performans değerleri

Çalışma Noktası	W2 hava debisi [kg/s]	P3/P2 -	Şaft gücü [shp]	Yakıt debisi [g/s]	T4 [K]	RPM [%]
Hesaplanan						
100_953	3,547	10,8	953,1	76,1	1340,5	90,0
%97_Seyir	2,661	11,5	690,0	52,1	1251,7	89,5
%95_Dolanma	2,422	8,6	603,0	48,7	1238,9	82,0
Referans						
100_953	3,493	10,6	953,1	69,0	1342,0	100,0
%97_Seyir	2,663	11,6	690,0	49,0	1302,0	97,0
%95_Dolanma	2,461	8,6	603,0	44,0	1248,0	95,0

6.5. Serbest Türbinli Varyant Motorun Modellenmesi

Tek şaftlı turboprop motorda kullanılan üç kademeli türbin, iki kademeli iki ayrı türbine dönüştürülerek serbest türbinli bir turboşaft motor elde edilmiştir. Serbest türbinli

motorlarda yüksek hızda dönen bir kompresör/türbin şaftı ve bundan bağımsız olarak dönen bir güç türbini şaftı bulunmaktadır. Yüksek basınç türbini kompresörü çevirmek için güç üretirken, serbest türbin sadece şaft gücü elde etmek kullanılır. Kompresör ve türbin arasındaki güç dengesi şu şekilde ifade edilmektedir;

$$\text{Kompresör Gücü} = W_2 \cdot (T_3 - T_2) \cdot cp_{hava} \quad (6.18)$$

$$\text{Yüksek Türbin Gücü} = \frac{\text{Kompresör Gücü} + \text{Aksesuar Gücü}}{\text{Mekanik verim}} \quad (6.19)$$

Kompresör gücü bilindiğinden, buradan gerekli türbin gücü hesaplanabilir.

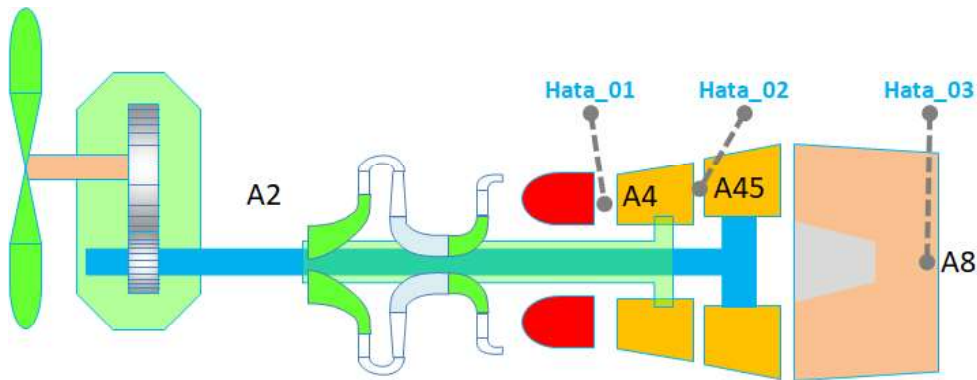
$$\text{Yüksek Türbin Gücü} = W_4 \cdot (T_4 - T_{45}) \cdot cp_{gaz} \quad (6.20)$$

Türbin gücünü verebilecek T_{45} sıcaklığı hesaplanır. Benzer şekilde şaftın üretmesi gereken güç bilindiğinden T_5 sıcaklığı hesaplanabilir.

$$\text{Alçak Türbin Gücü} = \frac{\text{Şaft gücü}}{\text{Mekanik verim}} \quad (6.21)$$

$$\text{Alçak Türbin Gücü} = W_5 \cdot (T_{45} - T_5) \cdot cp_{gaz} \quad (6.22)$$

Sıcaklıkların oranlarından basınç oranlarına geçilerek türbinlerin çalışma noktası bulunur.



Şekil 6.22. Serbest türbinli bir turboşaft motorun şematik gösterimi

Yüksek basınç türbin karakteristiği A_4 alanı ile düşük basınç (serbest) türbin karakteristiği A_{45} alanı ile kontrol edilmektedir. Bu motorun bileşen eşleştirme hesaplamaları için beş parametrelili bir girdi vektörü ve dört adet optimizasyon hedefi kullanılmıştır (Tablo 6.28). Tek şaftlı ve iki şaftlı motor performansları Tablo 6.29’da

karşılaştırılmıştır. Elde edilen sonuçlar, TPE-331 motorunun yayınlanmış test verileri ile Tablo 6.30’de sayısal olarak ve Şekil 6.23’te grafiksel olarak karşılaştırılmıştır.

Tablo 6.28. Optimizasyon tanımlama denklemleri

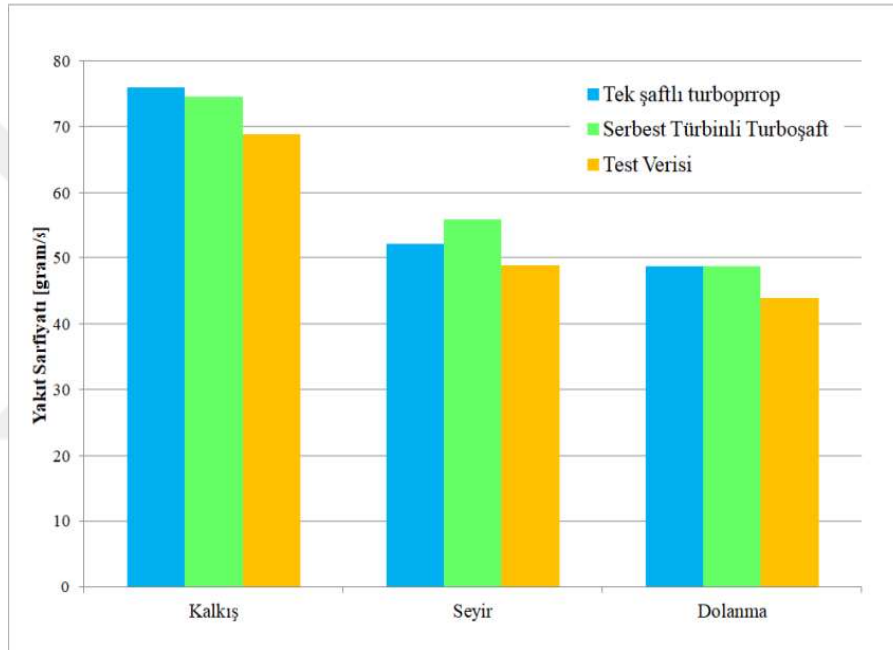
Minimize,	:	$f(x)_{Tasarım} = \text{maximum}(Hata_{01}, Hata_{02}, Hata_{03}) + Hata_{04}$ $f(x)_{Tasarım\ Dışl} = \text{maximum}(Hata_{01}, Hata_{02}, Hata_{03}) + Wf$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3, x_4\}^T, x_i \in [0,1], i = 0,1,2,3,4$
Parametreler	:	$Beta = x_0, W_2 = \text{Map}(Beta, N), CPR = \text{Map}(Beta, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_4 = A_{4_0} \cdot (0,5 + x_2)$ $A_{45} = A_{45_0} \cdot (0,5 + x_3)$ $A_8 = A_{8_0} \cdot (0,5 + x_4)$
Hata değerleri	:	$Hata_{01} = \frac{(W_4)_{hesaplanan} - (W_4)_{eğri}}{(W_4)_{hesaplanan}} \cdot 100$ $Hata_{02} = \frac{(W_{45})_{hesaplanan} - (W_{45})_{eğri}}{(W_{45})_{hesaplanan}} \cdot 100$ $Hata_{03} = \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{eğri}} \cdot 100$ $Hata_{04} = \frac{1 - E_{comp}}{E_{comp}}$

Tablo 6.29. Tek ve iki şaftlı motor performans sonuçlarının karşılaştırması

Parametre adı	Birim	Tek şaft	İki şaftlı
CPR	-	10,768	10,737
TPR(HPT)	-	9,467	3,583
TPR(LPT)	-	-	2,637
NPR	-	1,031	1,030
W25	kg/s	3,547	3,554
Wyakıt	g/s	76,07	74,64
Eff_comp	-	0,836	0,835
Eff_turb	-	0,900	0,900
T4	K	1340,5	1324,0
T8	K	833,68	819,75
A4	mm ²	3085,0	3096,6
A45	mm ²	-	10729,4
M8	-	0,220	0,215
A8	mm ²	70235,9	71154,0
F	N	438,2	426,9

Tablo 6.30. Test verileri ile hesaplamaların karşılaştırılması

Motor tipi	Çalışma noktası	Hava Debisi [kg/s]	P3/P2	Yakıt debisi [g/s]	T4 [K]
Tek şaftlı	Kalkış	3,547	10,8	76,1	1340,5
	Seyir	2,661	11,5	52,1	1251,7
	Dolanma	2,422	8,6	48,7	1238,9
Serbest türbinli	Kalkış	3,554	10,7	74,6	1324,0
	Seyir	2,637	11,7	56,1	1314,9
	Dolanma	2,496	8,4	48,7	1214,6
Test verisi	Kalkış	3,493	10,6	68,8	1342,0
	Seyir	2,663	11,6	48,9	1302,0
	Dolanma	2,461	8,6	44,0	1248,0



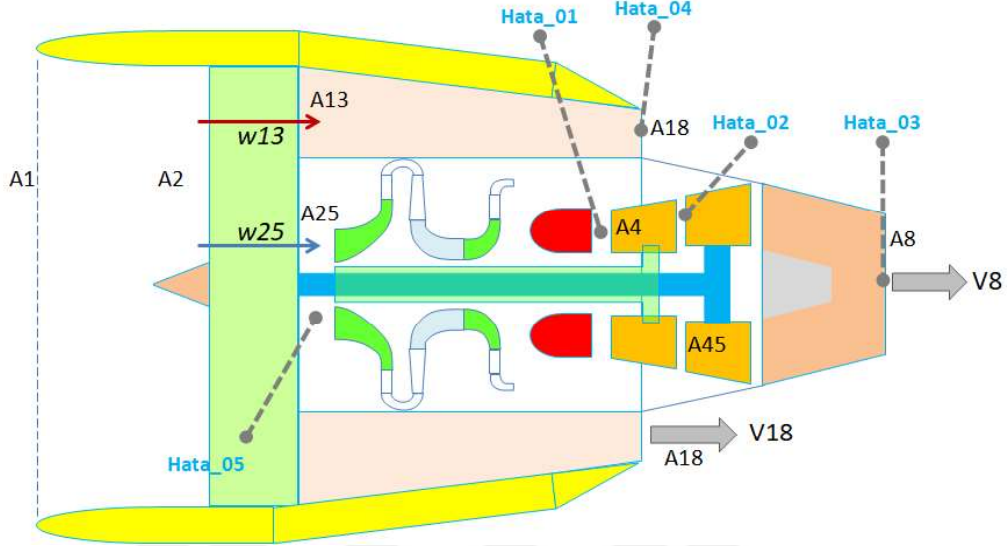
Şekil 6.23. Tek şaftlı, iki şaftlı model sonuçları ve test verisi

6.6. Turbofan Motoru Varyantının Modellemesi

Serbest türbinli bir turboşaft motor, şaft ucuna tek kademeli ayrık akışlı turbofan eklenerek, turbofan bir motora dönüştürülmüştür (Şekil 6.24). Bu turbofan motorda iki şaft ve itki üretmeye odaklanmış iki egzoz (soğuk ve sıcak egzoz çıkışları) bulunmaktadır. Yüksek basınç türbini kompresör gücü ve aksesuarların gücünü karşılarken, alçak basınç türbini fan gücünü karşılamaktadır.

Tek şaftlı motorun pervaneye iletmış olduğu şaft gücü korunarak serbest türbinli motor elde edilmiştir. Serbest türbinli şaftın ucuna, yine aynı şaft gücünü koruyarak bir fan ekleneceği öngörülmüştür. Bu şartlar altında en fazla itki üretecek fanın tasarlanabilmesi için temel olarak hangi basınç oranı ve by-pass oranında çalışacağının bulunması

gereklidir. Bu nokta fan tasarım noktası olarak düşünülebilir. Optimizasyon kodu yardımı ile öncelikle tasarım noktası değerleri hesaplanmıştır. Daha sonra mevcut bir fan haritası bu noktaya göre ölçeklenerek tasarım dışı noktalarda hesaplamalar yapılmıştır.



Şekil 6.24. İki şaftlı, ayırık akışlı bir turbofan motoru şematik gösterimi

Kompresör gücü, yüksek basınç türbini gücü, fan gücü ve alçak basınç türbini gücü arasındaki bağlantılar aşağıdaki denklemlerde verilmiştir.

$$\text{Kompresör Gücü} = W_{25} * (T_3 - T_{25}) * cp_{hava} \quad (6.23)$$

$$\text{Yüksek Basınç Türbin Gücü} = \frac{\text{Kompresör Gücü} + \text{Aksesuar Gücü}}{\text{Mekanik verim}} \quad (6.24)$$

$$\text{Alçak Basınç Türbin Gücü} = \frac{\text{Fan gücü}}{\text{Mekanik verim}} \quad (6.25)$$

$$\text{Fan Gücü} = \text{Şaft gücü} \quad (6.26)$$

$$\text{Fan Gücü} = (W_{13} + W_{25}) * (T_{13} - T_2) * cp_{hava} \quad (6.27)$$

$$\text{Fan Gücü} = W_{25} * (BPR + 1,0) * (T_{13} - T_2) * cp_{hava} \quad (6.28)$$

$$BPR = \frac{\text{Fan gücü}}{(T_{25} - T_2) * cp_{hava} * W_{25}} - 1,0 \quad (6.29)$$

W_{13} ve T_{13} parametreleri fan by-pass oranı ve basınç oranına bağlı parametrelerdir.

$$W_{13} = BPR \cdot W_{25} \quad (6.30)$$

$$T_{13} = T_2 \cdot FPR^{\left(\frac{\gamma-1}{\gamma}\right)} \quad (6.31)$$

Fan basınç oranı ve dolayısı ile T_3 sıcaklığının değişmesi kompresör çalışma noktasını dolayısı ile çekirdek motordaki bileşenlerin uyumunu değiştirmektedir. Buna bağlı olarak çekirdek motordan geçen hava debisi, çekirdek motor jet itkisi, alçak basınç türbini tarafından üretililecek itki de değişmektedir. Fan çalışma basıncı ve by-pass oranı harcanan fan gücünü belirlemektedir. Güç bütçesi sabit olduğundan, fan basınç oranı seçildiğinde, kompresör debisine bağlı olarak, fandan geçebilecek olan hava debisi de belirlenmiş olmaktadır. Elde edilen sonuçlar seçilen fan parametre aralığına bağlı olarak değişecektir. Bu çalışmada BPR değeri 0,5-6,0 arasında sınırlandırılmıştır.

Beta parametresi kompresör haritasında hız çizgisi üzerindeki bir noktaya işaret etmektedir. Bu noktadaki kompresör hava debisi ve basınç oranı haritadan alınarak W_{25} , P_{25} ve T_{25} değerleri hesaplanmaktadır. Tasarım noktasında optimum fan parametrelerini bulmak için altı parametrelili bir girdi vektörü ve beş adet optimizasyon hedefi kullanılmıştır (Tablo 6.31).

Tablo 6.31. Optimizasyon tanımlama denklemleri

Minimize,	:	$f(x)_{Tasarım} = \text{maximum}(Hata_{01}, Hata_{02}) + \text{maximum}(Hata_{03}, Hata_{04}) + Hata_{05}$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3, x_4, x_5\}^T, x_i \in [0,1], i = 0,1,2,3,4,5$
Parametreler	:	$Beta = x_0, W_2 = \text{Map}(Beta, N), CPR = \text{Map}(Beta, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_4 = A_{40} \cdot (0,5 + x_2)$ $A_{45} = A_{450} \cdot (0,5 + x_3)$ $A_8 = A_{80} \cdot (0,5 + x_4)$ $FPR = 1,0 + 0,6 \cdot x_5$
Hata değerleri	:	$Hata_{01} = \frac{(W_4)_{hesaplanan} - (W_4)_{eğri}}{(W_4)_{hesaplanan}} \cdot 100$ $Hata_{02} = \frac{(W_{45})_{hesaplanan} - (W_{45})_{eğri}}{(W_{45})_{hesaplanan}} \cdot 100$ $Hata_{03} = \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{hesaplanan}} \cdot 100$ $Hata_{04} = \frac{(W_{18})_{hesaplanan} - (W_{18})_{eğri}}{(W_{18})_{hesaplanan}} \cdot 100$

Tablo 6.32. Optimizasyon tanımlama denklemleri (devam)

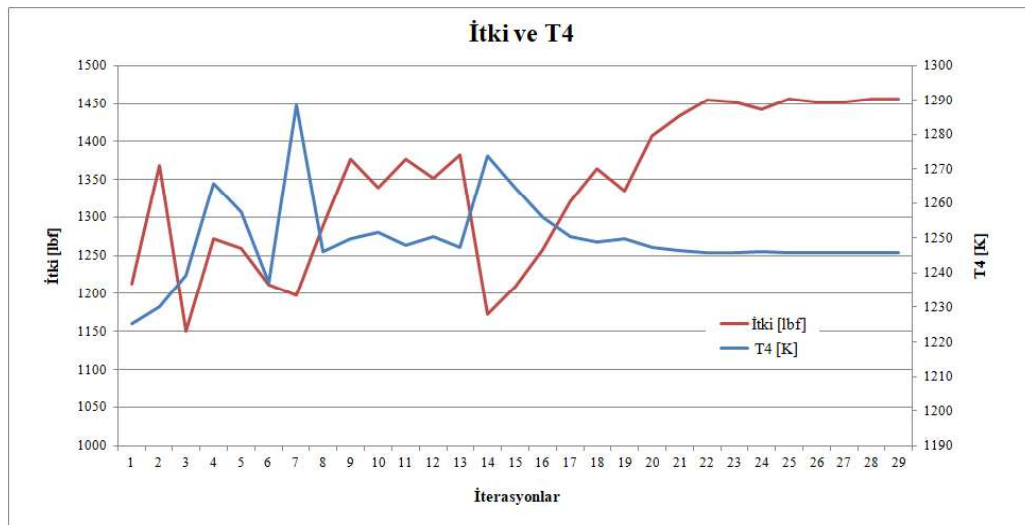
Tasarım hedefi	:	$Hata_{05} = c2 \cdot \left(\frac{100,0}{İtki} \right)$
----------------	---	--

Serbest türbinli motor konfigürasyonunda üretilen şaft gücü ile elde edilebilecek en yüksek turbofan motoru itkisini bulmak için birkaç farklı optimizasyon çalışması yapılmıştır. İlk çalışmada A_4 ve A_{45} değerleri iki şaftlı turboşaft motordaki değerler ile aynı tutulmuştur. A_8 ve A_{18} yüksek itki verecek şekilde seçilmiştir. Daha sonra A_4 ve A_{45} alanlarının da değişmesine izin verilmiş ve olası en yüksek itki bulunmuştur.

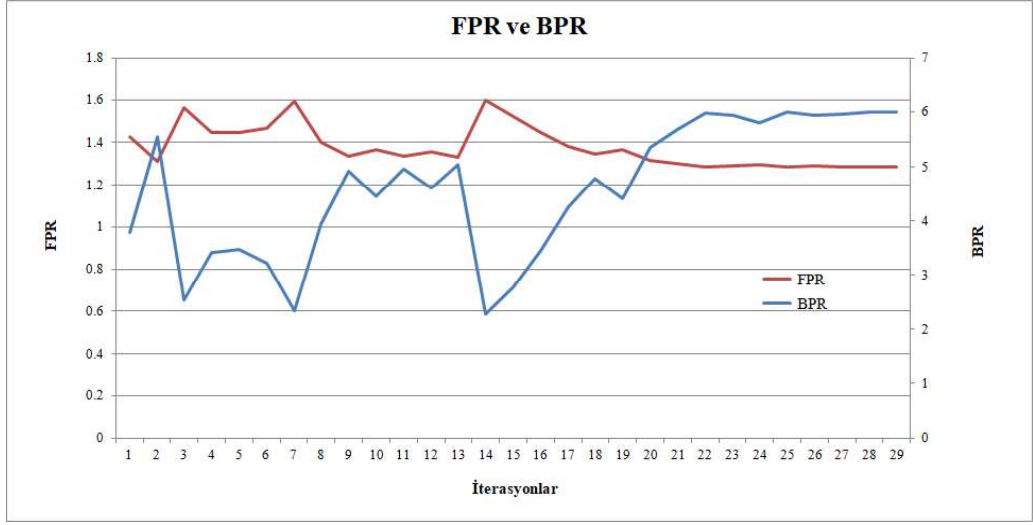
Yüksek basınç ve düşük basınç türbinlerinin kapasiteleri değiştirilmediğinde bulunan en yüksek itki 1456,3 lbf (6485,8 N) dur. Fan basınç oranı 1,286 ve by-pass oranı 5,999 değerine yakınsamıştır (Şekil 6.25, 6.26, 6.27).

A_4 ve A_{45} değerleri değiştirilmesine izin verildiğinde ise; bulunan en yüksek itki 1572,1 lbf (7001,7 N) dur. Fan basınç oranı 1,288 ve by-pass oranı 5,999 değerlerine yakınsamıştır. İki durumda bulunan FPR ve BPR aynı değere yakınsamıştır (Şekil 6.28, 6.29, 6.30). Türbinlerin alanlarındaki değişimlerden dolayı oluşan farklar Tablo 6.32’de gösterilmiştir. Aralarındaki itki farkı, sıcak egzozdaki itki artışından kaynaklanmaktadır.

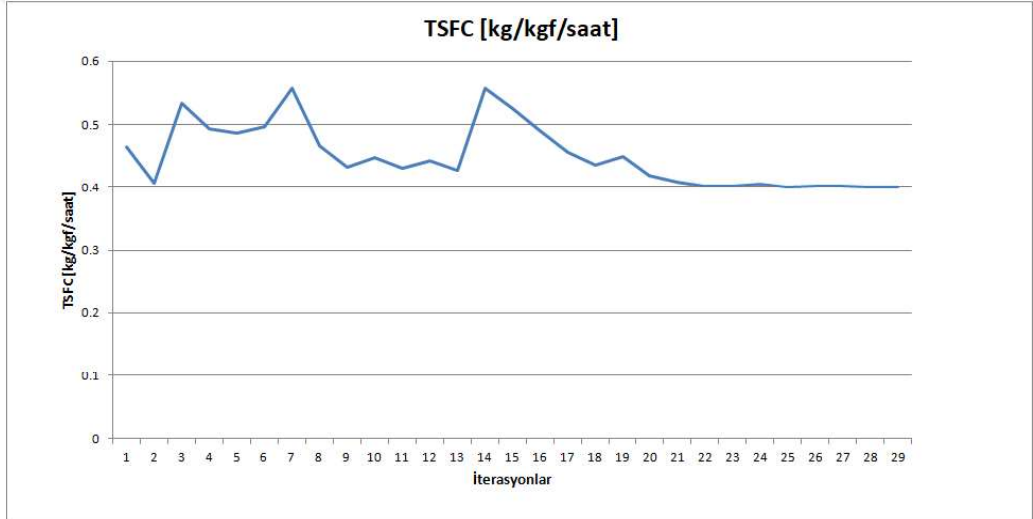
İtki seviyesindeki artış %7,95 iken T_4 sıcaklığındaki artış 91,5 K ve yakıt sarfiyatındaki artış %12,7 olmuştur. Türbin ömrü açısından bakıldığında sabit alanların kullanılması daha uygun görülmektedir. Buna göre fan tasarım noktası olarak Tablo 6.33’teki değerler temel alınmıştır.



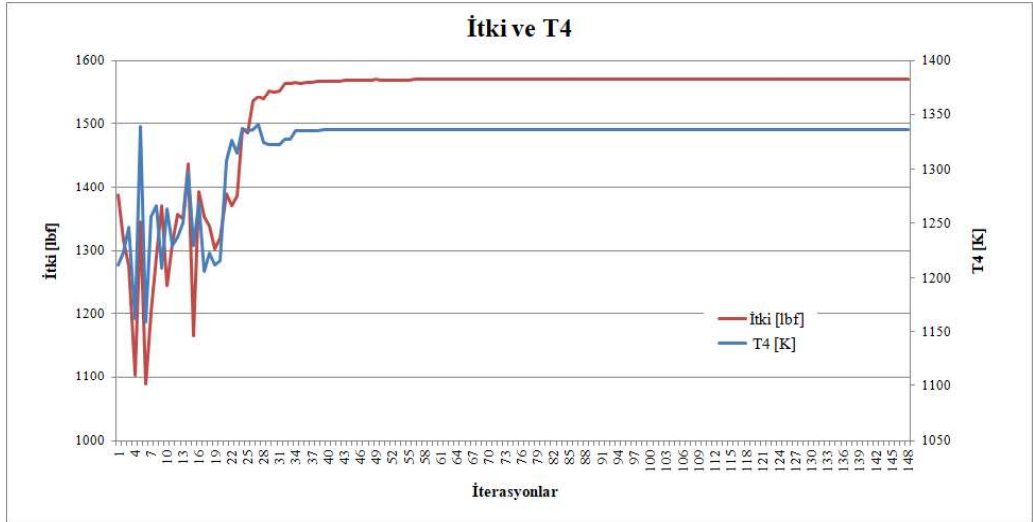
Şekil 6.25. İtki ve T_4 değişimi



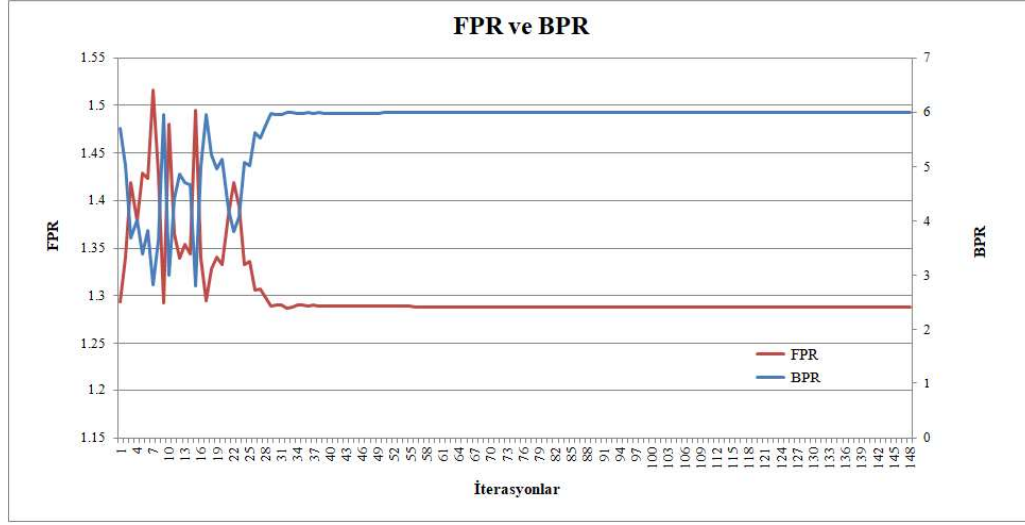
Şekil 6.26. FPR ve BPR değişimi



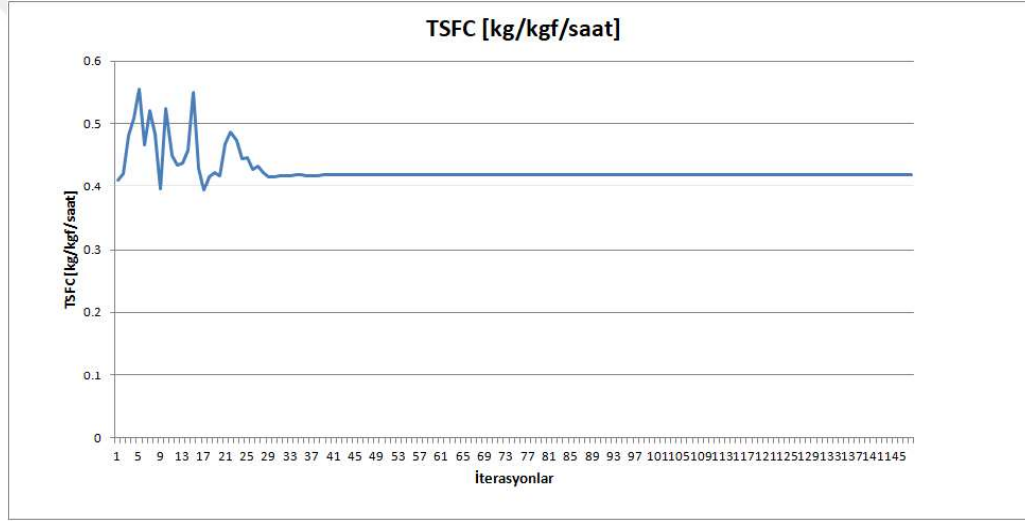
Şekil 6.27. TSFC Değişimi



Şekil 6.28. İtki ve T4 değişimi



Şekil 6.29. FPR ve BPR değişimi



Şekil 6.30. TSFC değişimi

Tablo 6.32. Türbin alanlarının değiştirilmesinin etkisi

Parametre adı	Birim	A ₄ , A ₄₅ Sabit	A ₄ , A ₄₅ değiştirilmiş
Şaft gücü	-	953,1	953,1
CPR	-	9,191	9,298
TPR	-	3,640	3,293
LPTPR	-	2,250	2,093
NPR	-	1,276	1,536
FPR	-	1,286	1,288
BPR	-	5,999	5,999
W25_crt	[kg/s]	3,279	3,252
W2	[kg/s]	4,046	4,018
Wyakıt	[g/s]	73,65	83,02
W18	[kg/s]	24,274	24,107
T4	[K]	1245,7	1337,2

Tablo 6.32. *Türbin alanlarının değiştirilmesinin etkisi (devam)*

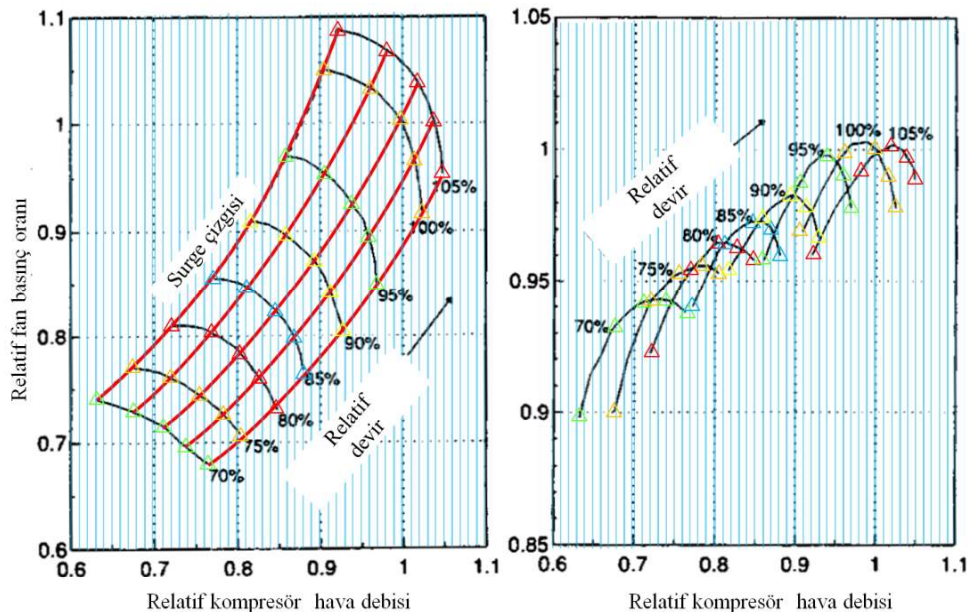
T8	[K]	764,5	849,1
V8	[m/s]	343,1	474,6
V18	[m/s]	208,9	209,7
V8/V18	-	1,641	2,263
M8	-	0,67	0,89
M18	-	0,61	0,61
A4	[mm ²]	3096,6	3170,5
A45	[mm ²]	10729,4	10131,5
A8	[mm ²]	24480,5	18684,6
A18	[mm ²]	95895,7	94910,0
Net İtki	[lbf]	1456,3	1572,1
F _{sıcak}	[lbf]	317,3	437,0
F _{soğuk}	[lbf]	1139,0	1135,1

Tablo 6.33. *Seçilen fan tasarım noktası değerleri*

Parametre adı	Değeri
FPR	1,286
BPR	5,999
W ₀ [kg/s]	28,32

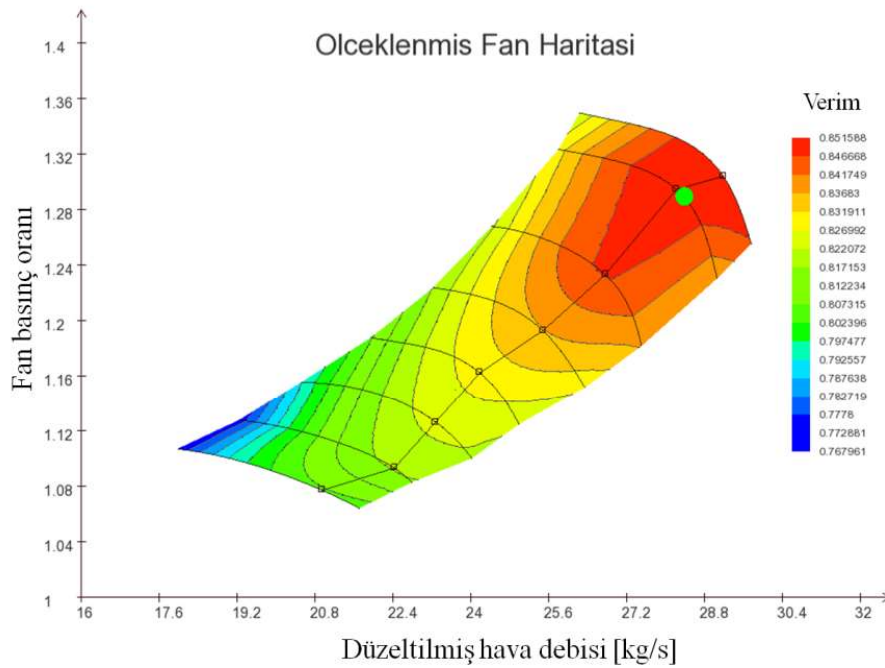
6.7. Turbofan Motoru Tasarım Dışı Hesaplamaları

Turbofan motorun tasarım dışı noktalarındaki performans hesaplamalarının yapılabilmesi için fan performans haritası gereklidir. (J.D. Stevenson, 1993) verilen JT15D motoru fan karakteristiği grafikleri (Şekil 6.31) kullanılarak Şekil 6.32'deki fan haritası elde edilmiştir [86].



Şekil 6.31. *JT15D Fan karakteristiği [86]*

Çekirdek motorun kalkış noktasında fan devrinin %100 olacağı kabul edilmiştir. Turbofan motorda çekirdek motorun devri yakıt debisi değiştirilerek kontrol edilebilmektedir ancak fan devri dolaylı olarak kontrol edilebilmektedir. Tasarım dışı noktalarda çekirdek motorun çalışma noktasına bağlı olarak fan devrinin ne olması gerektiği bilinmemektedir. Fan devrinin bulunabilmesi için optimizasyon parametrelerine fan devri ve hata fonksiyonlarına kompresör ile fan arasındaki hava debisi uyumsuzluğu eklenmiştir. Bu optimizasyon çalışması için sekiz parametrelili bir girdi vektörü kullanılmıştır (Tablo 6.34).



Şekil 6.32. Ölçeklenmiş fan performans haritası

Tablo 6.34. Optimizasyon tanımlama denklemleri

Minimize,	:	$f(x)_{Tasarım} =$ $maximum(Hata_{01}, Hata_{02}) + maximum(Hata_{03}, Hata_{04}) + Hata_{05}$ $f(x)_{Tasarım\ dışı} = maximum(Hata_{01}, Hata_{02}, Hata_{03}, Hata_{04})$ $+ Hata_{05} + Wf$
Girdi vektörü	:	$\vec{X} = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}^T, x_i \in [0,1], i = 0,1,2,3,4,5,6,7$
Parametreler	:	$Beta_{komp} = x_0, W_{25} = Map(Beta_{komp}, N), CPR = Map(Beta_{komp}, N)$ $T_4 = T_{4min} + x_1 \cdot (T_{4max} - T_{4min})$ $A_{18} = A_{18_0} \cdot (0,5 + x_2)$ $A_4 = A_{4_0} \cdot (0,5 + x_3)$

Tablo 6.34. Optimizasyon tanımlama denklemleri (devam)

Parametreler	:	$A_{45} = A_{45_0} \cdot (0,5 + x_4)$ $A_8 = A_{8_0} \cdot (0,5 + x_5)$ $Beta_{fan} = x_6, W_2 = Map(Beta_{fan}, N_{fan}), FPR = Map(Beta_{fan}, N_{fan})$ $N_{fan} = 70,0 + x_7 \cdot (105,0 - 70,0)$
Hata değerleri	:	$Hata_{01} = \frac{(W_4)_{hesaplanan} - (W_4)_{harita}}{(W_4)_{harita}} \cdot 100$ $Hata_{02} = \frac{(W_{45})_{hesaplanan} - (W_{45})_{eğri}}{(W_{45})_{eğri}} \cdot 100$ $Hata_{03} = \frac{(W_8)_{hesaplanan} - (W_8)_{eğri}}{(W_8)_{hesaplanan}} \cdot 100$ $Hata_{04} = \frac{(W_{18})_{hesaplanan} - (W_{18})_{eğri}}{(W_{18})_{hesaplanan}} \cdot 100$ $Hata_{05} = mutlak\left(\frac{\frac{W_2}{1 + BPR} - W_{25}}{W_{25}}\right) \cdot 100$
Tasarım hedefi	:	$Hata_{06} = c2 \cdot \left(\frac{1,0}{İt ki}\right)$

Tablo 6.24'te verilen, kalkış, seyir ve dolanma güç seviyelerine karşılık gelen çekirdek motor çalışma noktalarında elde edilen turbofan motor performansı Tablo 6.35'da verilmiştir. İrtifa şartlarındaki atmosfer koşulları için (NASA, 1976) dokümanından faydalanılmıştır [87].

Buna göre tek şaftlı turboprop motorun, serbest türbine sahip bir turbofan motoruna dönüştürülmesi ile 1505,0 lbf civarında itki üretecek bir motor elde etme potansiyeli bulunmaktadır. Kalkış sırasında türbinden yaklaşık 968,6 shp güç alınabilirken irtifa şartlarında bu güç seviyesi 564,4 ve 299,5 shp seviyelerine kadar düşmüştür. Turboprop motorun itki seviyesi bilinmediği için itki karşılaştırması yapılamamıştır. Şaft gücü açısından bakıldığında, turbofan motorun seyir ve dolanma şartlarında turboprop motora göre daha az şaft gücü harcadığı görülmektedir. Seyir sırasındaki T4 sıcaklığı kalkıştakine kıyasla 38 K daha fazla bulunmuştur.

Tablo 6.35. Turbofan motor optimizasyon hesaplama sonuçları

Parametre adı	Birim	Kalkış Tasarım noktası	Kalkış Tasarım dışı	Seyir Tasarım dışı	Dolanma Tasarım dışı
İrtifa	[m]	0	0	3810	2133
Mach	-	0	0	0.35	0.29

Tablo 6.35. *Turbofan motor optimizasyon hesaplama sonuçları (devam)*

SHP	[shp]	953,0	953,0	690,0	603,0
N komp	[%]	100,0	100,7	99,3	76,9
N fan	[%]	90,0	90,0	89,5	82,0
CPR	-	9,35	9,32	10,84	8,34
TPR	-	3,45	3,46	3,30	2,83
LPTPR	-	2,16	2,18	1,81	1,42
NPR	-	1,42	1,40	1,85	1,85
FPR	-	1,28	1,29	1,24	1,13
BPR	-	6,00	6,00	6,00	6,00
W25 crt	[kg/s]	4,059	4,067	2,939	2,605
W2	[kg/s]	28,412	28,466	20,571	18,231
Wyakıt	[g/s]	78,768	78,742	61,585	53,215
W18	[kg/s]	24,35	24,40	17,63	15,63
T4	[K]	1289,8	1288,5	1326,6	1264,3
T8	[K]	806,7	803,4	868,1	900,3
V8	[m/s]	395,3	387,1	533,4	543,2
V18	[m/s]	207,4	208,9	218,6	172,3
V8/V18	-	1,91	1,85	2,44	3,15
M8	-	0,74	0,73	1,00	1,00
M18	-	0,61	0,61	0,67	0,52
A4	[mm ²]	3107,2	3107,2	3107,2	3107,2
A45	[mm ²]	10208,2	10208,2	10208,2	10208,2
A8	[mm ²]	21923,8	21923,8	21923,8	21923,8
A18	[mm ²]	97028,7	97028,7	97028,7	97028,7
Net İtki	[lbf]	1501,2	1505,0	702,0	529,3
F sıcak	[lbf]	367,3	360,4	284,7	267,1
F soğuk	[lbf]	1133,9	1144,6	417,3	262,2
Fan gücü	[shp]	953,0	968,6	564,4	299,5

7. SONUÇ VE ÖNERİLER

Havacılıkta kullanılan itki sistemlerinin performans optimizasyonu, tasarımda çok önemli bir yere sahiptir. Yeni nesil optimizasyon kodlarının geliştirilmesi ve bunların itki sistemlerinin tasarımında uygulanması ile sürdürülebilir havacılık işletmeciliğinin ilerlemesine katkıda bulunmak mümkün olabilecektir.

İtki sistemlerinin optimizasyonunda; sistemin karmaşıklığına bağlı olarak doğrusal olmayan ve birden fazla yerel minimum değerine sahip fonksiyonlarla karşılaşılmaktadır. Bu tarz fonksiyonların çözülebilmesi için biyolojik hayattan esinlenen optimizasyon algoritmalarının uygun olduğu görülmektedir. Tez çalışması sırasında elde edilen kazanımlar ve çıkarımlar aşağıda maddeler halinde sıralanmıştır;

- i) Karınca kolonisi optimizasyonu ve parçacık sürüsü optimizasyonu yöntemleri popüler iki optimizasyon algoritmasıdır. Bu iki algoritmanın birlikte kullanımı ile çok daha iyi yakınsama başarımı (doğruluk) sağlayabilen hibrit bir kod (ACOPSO) geliştirilmiştir. Literatürde hibrit olarak geliştirilmiş birçok algoritma bulunmaktadır. Ancak bunların çoğunun detayları bilinmemektedir. Tez kapsamında, yeni ve özgün bir kod geliştirilmiştir.
- ii) Havacılıkta kullanılan gaz türbinli motorları; kullanım amacına göre farklı yapılarda tasarlanmaktadır. Turbojet ve turbofan motorlarda itki üretecek egzoz tasarımı yapılırken, turboprop ve turboşaft motorlarda şaft gücü ön plânda tutulmaktadır ve egzozda itki üretimi arka plândadır. Motor yapısına göre termodinamik denklemlerde bir kısım değişiklikler yapılması gerekmektedir.
- iii) İtki sistemleri performans hesaplamaları gaz dinamiği denklemleri ile yapılmaktadır ve bunlar oldukça iyi bilinmektedir. Ders kitaplarında genellikle tasarım noktası için hesaplamalar verilmektedir. Tasarım dışı noktalarda hesaplama yapılabilmesi için ise daha ileri seviyede iteratif çözüm alabilen kodlara ve performans haritalarına ihtiyaç duyulmaktadır. Performans haritalarının uygun şekilde ölçeklenmesi ve kod tarafından kullanılabilir hale getirilmesi gerekmektedir.
- iv) Literatürde, çeşitli konfigürasyonlara ait kompresör performans haritaları genel amaçlı olarak verilmekte ve kolayca uygulanması mümkün olabilmektedir. Türbin haritaları ise genellikle az bulunan ve uygulaması daha zor olan haritalardır. Türbin testlerinin yapıldığı ortam sıcaklığı ve basınç değerlerinin net şekilde bilinmesi gerekmektedir. Bu bilgiler mevcut olmadığında; egzoz karakteristik

eğrisine benzer şekilde türbin karakteristik eğrilerinin uygulanması ile iyi sonuçlar alındığı görülmektedir. Özellikle türbin boğulma basınç oranı ve iş üretebilme üst sınırının bilinmesi hesaplamalar açısından önem taşımaktadır. Kompresör ve türbin bileşenlerinin performans haritalarının kalitesi sonuçları önemli derece etkilemektedir.

- v) Hesaplaması yapılan itki sisteminde kullanılan mimariye bağlı olarak sıralı şekilde bir hesaplama ile performans parametreleri hesaplanmaktadır. Bu işlem tasarım noktasında yapılan hesaplama ile çok benzemektedir. Kütlenin ve enerjinin korunumu kanunları uygulanarak bileşenler arasındaki uyumluluk sağlanmaktadır. Fiziksel olarak gerçekleşen değerler ile olması gereken değerler arasındaki farklar hatalara neden olmaktadır. Tasarım noktasında veya tasarım noktası dışında bu hataların toplamı veya içlerindeki en büyükleri hedef fonksiyonu olarak alınmaktadır. Buna ilâve olarak tasarım kısıtlamaları; örneğin maksimum itki, minimum yakıt debisi, minimum yanma odası çıkış sıcaklığı gibi birçok parametre hedef fonksiyonuna dahil edilebilmektedir.
- vi) İtki sistemi hesaplamalarını ve optimizasyonu aynı anda yapan bir kod hem karmaşık olacaktır hem de böyle bir kodun geliştirilmesi daha zor olacaktır. Bu nedenle itki sistemi hesaplamalarının ayrı yerde yapıldığı ve optimizasyon koduna entegre edildiği bir yapı tasarlanmıştır. Bu şekilde algoritma ile birçok test fonksiyonu denendikten sonra hiç bir değişiklik yapmadan motor performans kodları ile entegre edilmiştir. Optimizasyon kodu tarafından her bir parçacık için bir girdi vektörü oluşturulup hedef fonksiyona gönderilmektedir. Elde edilen sonuçlar optimizasyon koduna geri gönderilmekte ve burada değerlendirilmektedir. Değerlendirme sonucuna göre yeni girdi vektörleri gönderilmektedir.

Özet olarak bu tez kapsamında;

- Hibrit bir optimizasyon algoritması kodu geliştirilmiştir
- Bu kodun test fonksiyonları ile yakınsama başarımının yeterliliği değerlendirilmiştir
- Turbojet, Turboşaft, Turboprop ve Turbopan mimarilerine sahip itki sistemleri için performans hesaplama fonksiyonları geliştirilmiştir
- İtki sistemlerinin en uygun noktalarda çalışması için gerekli türbin ve egzoz alanları hesaplanmıştır.

- Belirlenmiş alanlara sahip itki sistemlerinin tasarım dışı noktalarda performans hesaplamaları yapılmıştır

Tezin devamı niteliğinde yapılabilecek çalışmalar olarak şu konular önerilebilir;

- Karışık akışlı ve art yanmalı turbofan motor optimizasyonu yapılabilir. Çekirdek bir motordan veya ayırık akışlı bir turbofan motorda art yanmalı bir motora geçişte elde edilebilecek performans araştırılabilir.
- Motor performans fonksiyonları daha detaylandırılarak farklı bileşenlerin optimize edilmesi mümkün olabilir. Buna bağlı olarak daha verimli çevrimler bulunabilir.
- Motor testinden elde edilen veriler ile uyumun sağlanabilmesi için en uygun parametreler araştırılabilir.
- Motor sağlığı üzerinde çeşitli parametrelerin etkisi araştırılabilir ve operasyondaki motorların bazı arızalarının önceden tahmin edilebilmesi mümkün olabilir.

KAYNAKÇA

- [1] Sonar. (2022, Nisan 15). *coursepage.php*. [https://sonarlearning.co.uk/https://sonarlearning.co.uk/coursepage.php?Topic=game & course=sfml](https://sonarlearning.co.uk/https://sonarlearning.co.uk/coursepage.php?Topic=game&course=sfml) adresinden alınmıştır
- [2] Barr, D. (2022, Nisan 15). *One lone coder*. Code-It-Yourself! 3D Graphics Engine: <https://www.youtube.com/c/javidx9/videos> adresinden alınmıştır
- [3] Singiresu, S. (2009). *Engineering Optimization Theory and Practice Fourth Edition*. Hoboken, New Jersey: John Wiley & Sons inc.
- [4] Surjanovic, S., & Bingham, D. (2020, Nisan 15). *Virtual Library of Simulation Experiments: Test Functions and Datasets*. <https://www.sfu.ca/https://www.sfu.ca/~ssurjano/optimization.html> adresinden alınmıştır
- [5] Kalyanmoy, D. (1995). *Optimization for Engineering Design: Algorithms and Examples*. New Delhi: Prentice-Hall of India Private Limited.
- [6] Kumar, A. K. (2016, 04 18). Bio Inspired Computing – A Review of Algorithms and Scope of Applications. *Expert Systems with Applications*, 20-32.
- [7] Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. Massachusetts, United States: MIT Press.
- [8] Reeves, W. T. (1983). Particle Systems - A Technique for Modeling a Class of Fuzzy Objects. *Computer Graphics, Vol 17(3)*, 359-376.
- [9] Shiffman, D. (2022, Nisan 15). *Coding train, The nature of coding – particle systems*. TheCodingTrain: <https://www.youtube.com/c/TheCodingTrain/videos> adresinden alınmıştır
- [10] Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan)*, 39-43.
- [11] Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. (I. W. Intelligence, Dü.) *IEEE International Conference on Evolutionary Computation Proceedings, Cat. No.98TH8360*, 69-73.

- [12] Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. *Proc IEEE Congr Evol Comput.*2.(2), 1671 - 1676.
- [13] Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99.*, 3(Cat. No. 99TH8406), 1958-1962.
- [14] Van den Bergh, F., & Engelbrecht, A. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences, Vol 176*(8), 937-971.
- [15] Kamaruzaman, A., Zain, A., Yusuf, S., & Udin, A. (2013). Levy Flight Algorithm for Optimization Problems – A Literature Review. *Applied Mechanics and Materials*, 421-496.
- [16] Haklı, H., & Uğuz, H. (2014). A novel particle swarm optimization algorithm with Levy flight. *Applied Soft Computing*, 23, 333-345.
- [17] Shelokar, P. S., Siarry, P., Jayaraman, V. K., & Kulkarni, B. .. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, 188(1), 129-142.
- [18] Kaveh, A., & Talatahari, S. (2008). A hybrid particle swarm and ant colony optimization for design of truss structures. *ASIAN JOURNAL OF CIVIL ENGINEERING (BUILDING AND HOUSING)*, 9(4), 329-348.
- [19] Kıran, M. S., Gündüz, M., & Baykan, Ö. K. (2012). A novel hybrid algorithm based on particle swarm and ant colony optimization. *Applied Mathematics and Computation*, 219(4), 1515-1521.
- [20] Mosevich, J. (1986). Balancing hydraulic turbine runners-A discrete combinatorial optimization problem. *European Journal of Operational Research*, 26 (2), 202-204.
- [21] Laporte, G., & Mercure, H. (1988). Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3), 378-381.
- [22] Fathi, Y., & Ginjaipalli, K. K. (1993). A mathematical model and a heuristic procedure for the turbine balancing problem. *European Journal of Operational Research*, 65(3), 336-342.

- [23] Sinclair, M., Von Backström, T. W., & Du Buisson, J. J. (1993, November 1). Heuristic methods for arranging blades to minimize rotor. *Engineering Optimization*, 22(1), 19-26.
- [24] Von Backström, T. W., & Du Buisson, J. (1994). Blade Arrangement Strategies for Minimal Rotor Unbalance. *International Journal of Turbo and Jet Engines*, 11, 193-200.
- [25] Mason, A., & Rönnqvist, M. (1994). Solutions Methods for the Balancing of Jet Turbine. (D. o. Auckland, Dü.) *ORSNZ-proceedings*, 34, 184-189.
- [26] Choi, W., Kang, H., & Baek, T. (1999). A turbine-blade balancing problem. *Int. J. Production Economics*, 405-410.
- [27] Amiouny, S. V., Bartholdi, J. J., & Vate, J. H. (2000). Heuristic for balancing turbine Fans. (INFORMS, Dü.) *Operations Research*, 48(4), 591-602.
- [28] Dierckx, P. (1996). *Curve and Surface Fitting With Splines*. New York, NY, USA: Oxford University Press Inc.
- [29] Lu, F., & Milios, E. E. (1994). Optimal spline fitting to planar shape. *Signal Processing*, 37(1), 129-140.
- [30] De Boor, C., & Rice, J. R. (1968). Least Squares Cubic Spline Approximation I- Fixed knots. *CDS TR20*, 1-16.
- [31] De Boor, C., & Rice, J. R. (1968). Least Squares Cubic Spline Approximation II- Variable knots. *CDS TR 21*, 1-42.
- [32] Jupp, D. L. (1978). Approximation to data by splines with free knots. *SIAM Journal on Numerical Analysis*, 15(2), 328-343.
- [33] Kokkonis, P. A., & Leute, V. (1996). Least squares splines approximation applied to multicomponent diffusion data. *Computational Materials Science*, 6, 103- 111.
- [34] Blair, G. (2021, August 15). *extended the code here in Tension Splines by adding several new splines with more controls*. <http://wookie.tcl.tk/10530>: <http://wookie.tcl.tk/10530> adresinden alınmıştır

- [35] Uyar, K., & Ulker, E. (2017). B-spline curve fitting with invasive weed optimization. *Applied Mathematical Modeling*, 52, 320-340.
- [36] Iglesias, A., Galvez, A., & Collantes, M. (2015). Bat Algorithm for Curve Parameterization in Data Fitting. *International Conference on Cyberworlds, IEEE*.
- [37] Galvez, A., & Iglesias, A. (2011). Efficient particle swarm optimization approach for data fitting with free knot B-splines. *Computer-Aided Design*, 43, 1683-1692.
- [38] Galvez, A., & Iglesias, A. (2013). Firefly Algorithm for Explicit B-Spline Curve Fitting to Data Points. *Mathematical Problems in Engineering*.
- [39] Mishra, S. K. (2020, Eylül 24). *Global Optimization by Particle Swarm Method: A Fortran Program*. (G. University Library of Munich, Düzenleyen) 874.html: <https://ideas.repec.org/p/pramprapa/874.html> adresinden alınmıştır
- [40] Mishra, S. K. (2020, Eylül 26). *Some new test functions for global optimization and performance of repulsive particle swarm method*. (G. University Library of Munich, Düzenleyen) 2718: <https://mpra.ub.uni-muenchen.de/2718> adresinden alınmıştır
- [41] Ke, X., Zhang, Y., Li, Y., & Du, T. (2016). Solving Design of Pressure Vessel Engineering Problem Using a Fruit Fly Optimization Algorithm. *International Journal of Simulation Systems, Science & Technology*, 17(43), 5.1-5.7.
- [42] Yang, X.-S., Huyck, C., Karamanoglu, M., & Khan, N. (2013). True global optimality of the pressure vessel design problem: A benchmark for bio-inspired optimisation algorithms. *International Journal of Bio-Inspired Computation*, 5(6), 329-335.
- [43] Koziel, S., & Yang, X. S. (2011). *Computational Optimization, Methods and Algorithms*. Springer Berlin Heidelberg: Springer.
- [44] Gandomi, A., Yang, X.-S., & Alavi, A. (2011). Mixed variable structural optimization using Firefly Algorithm. *Computers & Structures*, 89, 2325-2336.
- [45] Mcknight, R. L., Maffeo, R. J., Tipton, M. T., & Weber, G. İ. (1992). *Component-Specific modeling*. NASA. Cincinnati, OH, United States: NASA.
- [46] Misté, G. A., & Benini, E. (2012). Performance of a Turboshaft Engine For Helicopter Applications Operating at Variable Shaft Speed. ASME (Dü.), *Proceedings of*

- ASME 2012 Gas Turbine India Conference* içinde (s. 701-715). Mumbai, Maharashtra, India: ASME.
- [47] Walsh, P. P., & Fletcher, P. (1998). *Gas Turbine Performance*. Malden, MA 02148-5020, USA: lackwell Science Publications.
- [48] Mattingly, J. D., Heiser, W. H., & Pratt, D. T. (2002). *Aircraft Engine Design*. Reston, Virginia: AIAA Educational Series.
- [49] Cohen, H., Rogers, C. G., & Saravanamuttoo, H. I. (1972). *Gas Turbine Theory*. New York: Wiley.
- [50] Kurzke, J. (1998). Gas Turbine Cycle Design Methodology: A Comparison of Parameter Variation with Numerical Optimization. *ASME 98-GT-343* (s. 1-7). Stockholm-Sweden: ASME.
- [51] Kurzke, J. (2005). How to create a performance model of a gas turbine from limited amount of information. S. a. ASME Turbo Expo 2005: Power for Land (Dü.), *GT2005-68536* içinde (s. 145-153). Reno-Tahoe, Nevada, USA: ASME.
- [52] Duponchel, J. P., Loisy, J., & Carrillo, R. (1992). Steady and Transient Performance Prediction of Gas Turbine Engines. *AGARD-LS-183* (s. 3.1- 3.21). 27-28 May 1992 at England and United States, on 9-10 June 1992 at Germany: AGARD.
- [53] Mirza Baig, F. S., & Saravanamuttoo, H. (1992). Off-Design Prediction of Turbofans using Gas dynamics. *Presented at International Gas turbine and Aeroengine congress and exposition* (s. 1-8). Orlando, FL June 3-6, 1992,: ASME.
- [54] Hünecke, K. (1997). *Jet Engines: Fundamentals of Theory, Design, and Operation*. Osceola, WI, USA: Motorbooks International.
- [55] Mattingly, J. D. (2006). *Elements of Propulsion Gas Turbine and Rockets*. Reston, Virginia: AIAA Educational Series.
- [56] Cesar, C., Vishal, S., Riti, S., & Pericles, P. (2015). On Optimisation of Environmentally Friendly Aircraft Engine Cycles. *Journal Of Aerospace Technology And Management*, 2, 143.

- [57] Atashkari, K., Nariman-Zadeh, N., Pilechi, A., Jamali, A., & Yao, X. (2005). Thermodynamic Pareto optimization of turbojet engines using multi-objective genetic algorithms. *International Journal of Thermal Sciences*, 1061–1071.
- [58] Boulkeraa, T., & Ghenaiet, A. (2010). Multi-objectif optimization of a turboprop thermal cycle. *7th International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics* (s. 353P357). Antalya, Turkey: Taylor&Francis Online.
- [59] Choi, J., Chung, W., & Sung, H. G. (2015). Multi-Objective Optimization of Turbofan Engine Performance Using Particle Swarm Optimization. *Journal of the Korean Society for Aeronautical & Space Sciences*, 43(4), 326–333.
- [60] Homaifar, A., Lai, H. Y., & McCormic, E. (1994). System optimization of turbofan engines using genetic algorithms. *Appl. Math. Modeling*, 18, 72-83.
- [61] Rifai, H. (2005). Turbojet Engine Performance Modelling Using Multi-Objective Optimization Algorithms. *Master Thesis, Department of mathematics*. Göteburg , Göteburg , Sweden : Chalmers University of Technology Göteburg University.
- [62] Reffold, C. N. (1995). Spreadsheet Modelling Of Turbojet Performance. *Int.J. Engng. Ed.*, 11(2), 103-110.
- [63] Ashwood, P. F. (1990). *The Uniform Engine Test Program*. Tullahoma, Tennessee 37388-0370, United States: AGARD-AR-248.
- [64] Bakalis, D. P., & Stamatis, A. G. (2010). Extended instrumentation and model calibration for a small micro turbine. *GT2010-22837* (s. 711-719). Glasgow, UK.: Proceedings of ASME Turbo Expo 2010: Power for Land, Sea and Air.
- [65] Choi, W., Lee, I. W., & Yang, J. H. (2011). The Performance Modeling of a Low Bypass Turbofan Engine with Afterburner for Supersonic Aircraft. *Journal of the Korean Society for Aeronautical & Space Sciences*, 39(3), 269–278.
- [66] Bourget, S., Kelmer, V., & Leonard, O. (2006). Cycle Optimization of a Turbine Engine: an Approach Based on Genetic Algorithm. *7th Belgian Congress on Theoretical and Applied Mechanics*. Mons, Belgium: Liege Universitesi.
- [67] Turan, Ö. (2007, Eylül). TURBOFAN MOTORLARININ ELİTİZMLİ GENETİK ALGORİTMA YÖNTEMİYLE OPTİMİZASYONU. *Doktora Tezi*. Eskişehir,

Türkiye: Anadolu Üniversitesi Fen Bilimleri Enstitüsü Sivil Havacılık Anabilim Dalı.

- [68] Mavris, D., & Denney, R. (2009). *Optimal Robust Matching of Engine Models to Test Data*. Atlanta, GA 30332-0150: Georgia Institute of Technology School of Aerospace Engineering.
- [69] L.Doel, D. (1992). TEMPER-A Gas Path Analysis Tool for Commercial Jet Engines. 92-GT-315 (s. 1-9). Cologne, Germany: ASME.
- [70] G.Evans, D. (1970). *Method For Determining Component Matching And Operating Characteristics For Turbojet Engines*. NASA. Washington D.C.: NASA Lewis Research Center.
- [71] Suraweera, J. K. (2011). Off-Design Performance Prediction of Gas Turbines without the use of Compressor or Turbine Characteristics. *Master Thesis*. Ottawa, Ontario, Canada: Carleton University.
- [72] Kurzke, J., & Halliwell, I. (2018). *Propulsion and Power An Exploration of Gas Turbine Performance Modeling*. Switzerland: Springer International Publishing AG.
- [73] Koenig, R. W., & Fishbach, L. H. (1972). *GENENG: A program for calculating design and off-design performance for turbojet and turbofan engines*. Cleveland, OH, United States: NASA Lewis Research Center.
- [74] Pinkel, B., & Karp, I. M. (1947). *A thermodynamic study of the turbojet engine*. Cleveland, Ohio: NACA.
- [75] Jones, S. M. (2007). *An Introduction to Thermodynamic Performance Analysis of Aircraft Gas Turbine Engine Cycles Using the Numerical Propulsion System Simulation Code*. Cleveland, Ohio: NASA Glenn Research Center.
- [76] Grannan, N. D., Hoke, J., McClearn, M. J., Litke, P., & Schauer, F. (2017). Trends in JetCAT Microturbojet-Compressor Efficiency. *55th AIAA Aerospace Sciences Meeting* (s. 1-8). Grapevine, Texas: AIAA.
- [77] JetCat. (2021, Subat 28). *strahlturbinen.htm*. [www.jetcat.de:www.jetcat.de/jetcatturbinen/strahlturbinen.htm](http://www.jetcat.de/jetcat.de:www.jetcat.de/jetcatturbinen/strahlturbinen.htm) adresinden alınmıştır

- [78] BorgWarner. (2021, Subat 25). *Performance Turbochargers 2017 Catalog*.
www.borgwarnerboosted.com: www.borgwarnerboosted.com adresinden alınmıştır
- [79] Garrett Service Manual. (2021, Temmuz 20). *Garrett Service Manual, 72-00-00, page 4*.
reheavenly.weebly.com: <https://reheavenly.weebly.com/blog/tpe331-14-maintenance-manuals> adresinden alınmıştır
- [80] Becker, M. D., & Frounfelker, B. (1992). Gyroscopic Test for the T800-LHT-800 Turboshift Engine. *SAE Transactions*, 101(1), 1715-1724.
- [81] Davis, M., Beale, D., Lt. Col. Boyer, K., & Maj. O'Dowd, D. (2006). An Example for Integrated Gas Turbine Engine Testing and Analysis Using Modeling and Simulation. *Proceedings of ITEA International Test and Evaluation Association Modeling and Simulation Conference*, (s. 1-10). Las Cruces, New Mexico.
- [82] Palmer, D. L., & Waterman, W. F. (1994). Design and Development Of An Advance Two-stage Centrifugal Compressor. A. I. Exposition (Dü.), 94-GT-202 içinde (s. 1-11). The Hague, Netherlands: ASME.
- [83] Frignac, J. P., & Privoznik, E. J. (1979). The Growth and Development of the TPE331. *ASME Paper 79-GT-164*, (s. 1-7). San Diego.
- [84] Ghorbanian, P. K., AminiMagham, M. R., Ebrahimi, A., Nazemi, A., Zolfaghari, A., Etemadianmofrad, H., . . . Saffizadeh, A. (2019). *STS-1000: A Candidate Turboshift Engine for Hybrid Electric Medium Altitude Long Endurance Search and Rescue UAV*. Tehran, Iran: Sharif University of Technology Azadi Ave.
- [85] AIAA. (2021, Ekim 10). *AIAA Foundation Student Design Competition 2018/19 Undergraduate Team – Engine*. AIAA Web sitesi:
https://www.aiaa.org/docs/default-source/uploadedfiles/education-and-careers/university-students/design-competitions/undergraduate-team-engine-design-rfp-candidate-engines-for-hybrid-electric.pdf?sfvrsn=cd35c4c7_0 adresinden alınmıştır
- [86] J.D. Stevenson, H. S. (1993). Simulating Indirect Thrust Measurement Methods For High Bypass Turbofans. *ASME Paper 93-GT-335*, (s. 1-10). Cincinnati, Ohio.

[87] NASA. (1976). *Technical Memorandum (TM) NASA-TM-X-74335*, “U.S. Standard Atmosphere”. Washington D.C.



EKLER

EK-1. Ant Colony Optimizasyon Kodu (Excel Visual Basic)

```
Public yol_deger(100) As Double
Public yol_pheromone(100) As Double
Public o_siniri(100) As Double
Public karınca(100) As Double
Public sonuc(100) As Double
Public kutu_secimi(100) As Integer
Public kutu_carpani(100) As Integer
Public yol_sayisi As Integer
Public iterasyon_sayisi As Integer
Public analiz_sayisi As Integer

Sub ACO_ana()
    Dim a_sinir As Double
    Dim u_sinir As Double
    Dim k_sayisi As Integer
    Dim y_sayisi As Integer
    Dim ol As Double
    Dim bo As Double
    Dim tol As Double

    a_sinir = 0          'aranan degerin alt siniri
    u_sinir = 1          'aranan degerin ust siniri
    k_sayisi = Init_Menu.karınca_sayisi.Value    'kac karınca gonderildigi
    y_sayisi = k_sayisi * 2 + 1    'her bir noktada secilebilecek yol sayisi
    ol = 1.3              'feromon artis orani
    bo = 0.3              'feromon bozulma orani
    tol = 0.000001       'karınca yaklasma toleransi
    analiz_sayisi = 0
    son_3sans = 0
```

If (k_sayisi < 4) Then

 k_sayisi = 4

End If

If (y_sayisi < 7) Then

 y_sayisi = 7

End If

ilk_a_sinir = a_sinir

ilk_u_sinir = u_sinir

ilk_k_sayisi = k_sayisi

ilk_y_sayisi = y_sayisi

ilk_ol = ol

ilk_bo = bo

ilk_tol = tol

Worksheets("Sheet1").Range("E2:E102") = ""

bilgi_yaz (" Analiz baslangici : " & Time)

bilgi_yaz ("-----")

optimum_x = ACO(a_sinir, u_sinir, k_sayisi, y_sayisi, ol, bo, tol)

Worksheets("Sheet1").Cells(2, 5) = optimum_x

bilgi_yaz ("-----")

bilgi_yaz ("Ilk bulunan deger -> x = " & optimum_x)

For tekrar = 1 To 50

 ilk_x = optimum_x

 gcc_a_sinir = optimum_x - (u_sinir - a_sinir) / (y_sayisi * 1#)

 gcc_u_sinir = optimum_x + (u_sinir - a_sinir) / (y_sayisi * 1#)

 If (gcc_a_sinir > a_sinir) Then

 a_sinir = gcc_a_sinir

 End If

```

If (gcc_u_sinir < u_sinir) Then
    u_sinir = gcc_u_sinir
End If

```

```

If (y_sayisi > 7) Then
    y_sayisi = y_sayisi - 1
End If

```

```

tol = tol / 1.5
bilgi_yaz ("Alt sinir: " & a_sinir & " , Ust sinir: " & u_sinir)
optimum_x = ACO(a_sinir, u_sinir, k_sayisi, y_sayisi, ol, bo, tol)
Worksheets("Sheet1").Cells(2 + tekrar, 5) = optimum_x
bilgi_yaz ("-----")
bilgi_yaz ("Bulunan basarili deger -> x = " & optimum_x)
yakinsama = Abs((ilk_x - optimum_x) / ilk_x)

```

```

If (yakinsama < 0.000001) Then
    bilgi_yaz ("> Cozum yakinsadi .")
    bilgi_yaz ("> zoom = " & tekrar)
    bilgi_yaz ("> Analiz sayisi = " & analiz_sayisi)
    Exit For
End If

```

```

Next

```

```

End Sub

```

```

Function ACO(alt_sinir As Double, ust_sinir As Double, karınca_sayisi As Integer,
yol_sayisi As Integer, olcek As Double, bozulma As Double, tolerans As Double)

```

```

    iterasyon_sayisi = 200
    toplam_pheromone = 0
    ' her bir yola dusen degeri hesaplayip diziye aktar
    ' her yol icin ilk pheromone degerini ata
    For I = 1 To yol_sayisi

```

```

        yol_deger(I) = alt_sinir + (I - 1) * (ust_sinir - alt_sinir) / (yol_sayisi - 1)
        yol_pheromone(I) = 1           'ilk deger atamasi yap
        toplam_pheromone = toplam_pheromone + yol_pheromone(I)
        'bilgi_yaz ("yol_degeri[" & i & "]=" & yol_deger(i) & ", pheromon[" & i & "]="
" & yol_pheromone(i))
    Next

```

'olasilik sinirlari yolsayisindan 1 fazla

o_siniri(1) = 0

o_siniri(yol_sayisi + 1) = 1

'olasilik sinirlari feromon degerine bagli olarak belirleniyor

'birinci ve sonun degerleri 0-1 oldugundan aradaki degerler hesaplaniyor

For I = 2 To yol_sayisi

o_siniri(I) = o_siniri(I - 1) + yol_pheromone(I) / toplam_pheromone

'bilgi_yaz ("Alt sinir : " & o_siniri(i - 1) & " - Ust sinir : " & o_siniri(i))

Next

'bilgi_yaz ("Alt sinir : " & o_siniri(i - 1) & " - Ust sinir : " & o_siniri(i))

kutu_ciz (yol_sayisi)

iterasyon = 1

Do While iterasyon < iterasyon_sayisi

basarili_deger = 0

min_deger = 9999999

max_deger = -9999999

basarim = 0

bilgi_yaz ("iterasyon no: " & iterasyon)

' karincalara rastgele degerler ver, yeni fonksiyonlari hesapla

For j = 1 To karinca_sayisi

Randomize

20 karinca(j) = Rnd(1)

```

        kutu_secimi(j) = kutu_sec(karinca(j), yol_sayisi)
    If (iterasyon < karinca_sayisi) Then
        For k = 1 To j - 1
            If (kutu_secimi(j) = kutu_secimi(k)) Then
                GoTo 20
            End If
        Next
    End If
    sonuc(j) = fx(yol_deger(kutu_secimi(j)))
    analiz_sayisi = analiz_sayisi + 1
    'bilgi_yaz ("Karinca [" & j & "]=" & karinca(j) & " Kutu secimi = " &
    kutu_secimi(j) & ", sonuc = " & sonuc(j))

    If (sonuc(j) > max_deger) Then
        max_deger = sonuc(j)
    End If
    If (sonuc(j) < min_deger) Then
        min_deger = sonuc(j)
        basarili_deger = yol_deger(kutu_secimi(j))
    End If
Next

' hedef minimum degeri yakalamak
' basarili olanlari say
For j = 1 To karinca_sayisi
    kutu_no = kutu_secimi(j)
    If (sonuc(j) = min_deger) Then
        basarim = basarim + 1
        kutu_carpani(kutu_no) = 1
        'bilgi_yaz ("[Basarili Kutu : " & kutu_no)
    Else
        kutu_carpani(kutu_no) = 0
    End If
Next

```

End If

Next

' En iyi ve en kotu degere gore yeni pheromone degeri hesapla

' en iyi ve en kotu orani 1.001 ile 0.999 arasinda ise donguden cik

$gcc_oran = Abs(min_deger / max_deger)$

If $gcc_oran > 1$ Then

$yeni_pheromone_miktar = olcek * basarim * gcc_oran$

If $gcc_oran < (1\# + tolerans)$ Then

Exit Do

End If

Else

$yeni_pheromone_miktar = olcek * basarim / gcc_oran$

If $gcc_oran > (1\# - tolerans)$ Then

Exit Do

End If

End If

'bilgi_yaz ("***** HESAPLANAN PHEROMONE = " &
yeni_pheromone_miktari & " *****")

' Bozulma miktarina gore tum yollardaki pheromone degerlerini yariya indir

For $jjj = 1$ To yol_sayisi

$yol_pheromone(jjj) = (1 - bozulma) * yol_pheromone(jjj)$

Next

' Karincalarin gittigi yollardaki pheromone degerlerini duzenle

' basarili yol carpani 1,

' basarisiz yol carpani 0

For $kkk = 1$ To karinca_sayisi

$kutu_no = kutu_secimi(kkk)$


```

        yol_pheromone(kutu_no) = (1 - kutu_carpani(kutu_no)) *
yol_pheromone(kutu_no) + kutu_carpani(kutu_no) * (yeni_pheromone_miktari +
yol_pheromone(kutu_no) / (1 - bozulma))

```

```

        'bilgi_yaz ("Karıncı No: " & kkk & " , kutu no: " & kutu_no & " ," & "
pheromone : " & yol_pheromone(kutu_no) & " , Kutu carpanı : " & kutu_carpani(kutu_no))

```

```

    Next

```

```

For jjj = 1 To yol_sayisi

```

```

    'bilgi_yaz ("yol_pheromone(" & jjj & ") = " & yol_pheromone(jjj))

```

```

Next

```

```

toplaml_pheromone = 0

```

```

For j = 1 To yol_sayisi

```

```

    toplaml_pheromone = toplaml_pheromone + yol_pheromone(j)

```

```

Next

```

```

For I = 2 To yol_sayisi

```

```

    o_siniri(I) = o_siniri(I - 1) + yol_pheromone(I - 1) / toplaml_pheromone

```

```

    'bilgi_yaz ("Alt sinir : " & o_siniri(i - 1) & " - Ust sinir : " & o_siniri(i))

```

```

Next

```

```

    'bilgi_yaz ("Alt sinir : " & o_siniri(i - 1) & " - Ust sinir : " & o_siniri(i))

```

```

kutu_ciz (yol_sayisi)

```

```

iterasyon = iterasyon + 1

```

```

For j = 1 To yol_sayisi

```

```

    kutu_carpani(j) = 0

```

```

Next

```

```

Loop

```

```

ACO = basarili_deger

```

```

End Function

```

```
Function fx(x As Double)
```

```
    Pi = WorksheetFunction.Pi()
```

```
    'fx = x * x - 2 * x - 11
```

```
    fx = 1 - Exp(4 * x) * (Sin(5 * x * Pi)) ^ 4
```

```
End Function
```

```
Function kutu_sec(karinca_degeri As Double, yol_sayisi As Integer)
```

```
    For I = 2 To yol_sayisi
```

```
        If (karınca_degeri >= o_siniri(I - 1) And karınca_degeri < o_siniri(I)) Then
```

```
            kutu_sec = I - 1
```

```
        Exit For
```

```
    End If
```

```
Next
```

```
End Function
```

EK-2. ACOPSO Parçacık Sürüşü Struct tanımları (C++)

```
//{ Struct Definition begins
```

```
struct sParticle {
```

```
    double x[6]; // particle position X = { x0, x1, x2, ... }
```

```
    double vx[6]; // velocity of particle VX = { vx0, vx1, vx2, ... }
```

```
    double pBest ; // particle evaluated function value
```

```
    int id;
```

```
    int leader; // which leader to follow for food :)
```

```
};
```

```
struct food {
```

```
    double x[6]; // box position X = { x0, x1, x2, ... }
```

```
    double p_min; // lower limit for pheromone tournament
```

```
    double p_max; // upper limit for pheromone tournament
```

```
    double feromon; // pheromone level at this path
```

```
    double deger; // value of the path or box
```

```

    int followers ; // number of followers
};

struct swarm {
    //{ swarm parameters setup
    int sSize ;          // swarm size
    int sBestParticle ;   // best particle
    double swarm_best ;
    int iteration = 0 ;    // number of iterations
    int improved = 1;      // check is there is improvement
    double sBestValue=1.0e6 ;    // best function value of the swarm - works to
minimize this
    double social_distance = 60 ;//60 ; // allowable distance between swarm
individuals
    double social_distance_x = 30 ; // allowable distance between swarm individuals
    double limit_xlow[6] ; // lower bound of the input parameters
    double limit_xup[6] ; // upper bound of the input parameters
    double speed_limit[6] ; // speed limit for each parameter = (limit_xup-
limit_xlow)*0.03;
    double speed_ratio = 0.03 ; // approximately 33 modes can be caught
    double acceleration_factor = -0.05 ;
    double inertia = 0.91 ; // velocity contribution
    double decelerate = 0.97 ; // when particle find food decrease its speed
    double accelerate = 1.02 ; // if particle could not find food increase towards to
leader position
    bool converged=false;
    sParticle target;
    vector<sParticle> Particles;

    food kutu[20]; // 10 food sources will be marked - change also max_path = 10 ;
    // initialize the swarm by given particle number
    void setSize(int number) {
        double gcc_best_value = 1e6 ;

```

```

srand(time(0));
auto nSeed = time(0);
mt.seed(nSeed);
// std::time_t result = std::time(nullptr);
//std::cout << std::asctime(std::localtime(&result))
cout << " Initializing the swarm ... " << std::asctime(std::localtime(&nSeed)) ;
cout << " Number of particles : " << number << " , Food locations = " <<
max_path << endl;
// setup boundaries for each parameter and also speed limit depending on the
range
limit_xlow[0] = 0.0 ;limit_xup[0] = 1.0 ; // double mdot = x[0]; // W2_try
limit_xlow[1] = 0.0 ;limit_xup[1] = 1.0 ; // double TPR = x[1]; // TPR
limit_xlow[2] = 0.0 ;limit_xup[2] = 1.0 ; // double TPR = x[2]; // A4_ratio
limit_xlow[3] = 0.0 ;limit_xup[3] = 1.0 ; // double mdot = x[3]; // A8_ratio
limit_xlow[4] = 0.0 ;limit_xup[4] = 1.0 ; // double Ecomb = x[4]; // TPR
MAKS
limit_xlow[5] = 0.0 ;limit_xup[5] = 1.0 ; // double Ecomb = x[5]; // mf
MAKS
// limit_xlow[6] = 0.0 ;limit_xup[6] = 1.0 ; // double A2_ratio = x[6]; // A2_
compressor scale
// limit_xlow[7] = 0.0 ;limit_xup[7] = 1.0 ; // double A4_ratio = x[7]; // A4_
turbine scale
// limit_xlow[8] = 0.0 ;limit_xup[8] = 1.0 ; // double A8_try = x[8]; // A8_
Exhaust area ratio

for (int j=0;j<boyut;j++){
    speed_limit[j] = (limit_xup[j]-limit_xlow[j])*speed_ratio ; // speed limit is
given according to parameter boundaries
}
// setup initial targets for the swarm
for (int k=0 ; k<max_path ; k++) {
    for (int j=0; j<boyut ;j++){
        kutu[k].x[j] = (limit_xup[j]-limit_xlow[j])*dis(mt)+limit_xlow[j] ;
        kutu[k].followers = 0 ;
    }
}

```

```

    }
    kutu[k].p_min = 1.0/(1.0*max_path)*(k+0.0);
    kutu[k].p_max = kutu[k].p_min+ 1.0/(1.0*max_path);
    kutu[k].feromon = 1.0 ;// ilk deger
    kutu[k].deger = (*fTest)(kutu[k].x, boyut); // best function value - initial

    if (kutu[k].deger<gcc_best_value) {
        gcc_best_value = kutu[k].deger;
    }
}
// arrange pheromone levels according to function values
yeni_total_feromon= 0.0 ;
double gcc_min = 999999.0 ;
double gcc_max = -999999.0;
// arrange kutu ranges so the random selection possibility will change
for (int kk=0; kk<max_path; kk++) {
    if (kutu[kk].deger<=gcc_min)
        gcc_min = kutu[kk].deger ;
    if (kutu[kk].deger>=gcc_max)
        gcc_max = kutu[kk].deger ;
}
double ort_deger = (gcc_min + gcc_max)*0.5 ;
for (int kk=0; kk<max_path; kk++) {
    if (kutu[kk].deger<ort_deger) {
        kutu[kk].feromon = kutu[kk].feromon * olcek ; // (1.0+(gcc_min-
        kutu[kk].deger)*(gcc_min-kutu[kk].deger)/(ort_deger*ort_deger)) ;
    } else {
        kutu[kk].feromon = kutu[kk].feromon * bozulma ; // (1.0-(gcc_min-
        kutu[kk].deger)*(gcc_min-kutu[kk].deger)/(ort_deger*ort_deger)) ;
    }
    yeni_total_feromon = yeni_total_feromon + kutu[kk].feromon ;
}

```

```

toplaml_feromon = yeni_total_feromon ;
// setup initial targets for the swarm
kutu[0].p_min = 0.0;
kutu[0].p_max = kutu[0].feromon/toplam_feromon;
//cout << kutu[0].deger << endl ;
//cout << "kutu no: " << 0 << " delta_p: " << kutu[0].p_max-
kutu[0].p_min << " , feromon: " << kutu[0].feromon << endl;

for (int kkk=1 ; kkk<max_path ; kkk++) {
    kutu[kkk].p_min = kutu[kkk-1].p_max;
    kutu[kkk].p_max=kutu[kkk].p_min+
kutu[kkk].feromon/toplam_feromon;
    // cout << kutu[kkk].deger << endl ;
    // cout << "kutu no: " << kkk << " delta_p: " << kutu[kkk].p_max-
kutu[kkk].p_min << " , feromon: " << kutu[kkk].feromon << endl;
}

/////
cout << endl;
sParticle sp; // dummy particle for filling array
sp.x[0] = 0.5;
sp.x[1] = (10.3-1.0)/29.0;
sp.x[2] = 4.0/11.0;
sp.x[3] = 0.762254;
sp.x[4] = 0.5;
sp.x[5] = 0.5;

echo_on = 1 ;
sp.pBest = (*fTest)(sp.x, boyut);
cout << " Design values : " << sp.pBest << endl ;
echo_on = 0 ;
// assign random positions and random velocities to the particles
// track best function value
for (int i=0; i<number; i++) {

```

```

for (int j=0; j<boyut; j++){
    // speed_ratio = dis(mt)*0.1 ;
    sp.x[j] = (limit_xup[j]-limit_xlow[j])*dis(mt)+limit_xlow[j] ;
    sp.vx[j] = (limit_xup[j]-limit_xlow[j])*(0.5-dis(mt)*speed_ratio);
}
sp.pBest = (*fTest)(sp.x, boyut);
sp.id = i;
sp.leader = -1 ;

// look for nearest leader
double gcc_mesafe = 1000000000.0 ;
for (int k=0; k<max_path; k++) {
    double gcc_toplam = 0.0;
    for (int j=0; j<boyut; j++){
        gcc_toplam = gcc_toplam + (sp.x[j]-kutu[k].x[j])*(sp.x[j]-kutu[k].x[j]);
    }
    if (gcc_toplam < gcc_mesafe) {
        gcc_mesafe = gcc_toplam ;
        sp.leader = k;
        kutu[k].followers += 1 ;
    }
}

// cout << "Particle : " << i << " , Leader : " << sp.leader << endl ;

if (sp.pBest < swarm_best ) {
    swarm_best = sp.pBest ;
    sBestParticle = sp.id ;
}
Particles.push_back(sp) ;
}

cout << " Particles are inserted " << endl;
iteration =1;

```

```

    }
//} init end

void update(){
    double ax; // acceleration towards new target
    double xcurr[boyut] ; // current position before update
    double temp_value ;
    double px_max=-999999.0 ; // social distance measure - right
    double px_min= 999999.0 ; // social distance measure - left
    // there are N boxes of food
    // among the boxes with fragrant pheromone select one direction
    // addition to boxes there is one random direction
    // sp the total number of possible directions is 10+1
    double toplam_feromon = 0.0 ;
    double gcc_min = 99999.0 ;
    double gcc_max = -99999.0;
    double yeni_feromon_miktari = 0.0 ;
    int selected_path = -1;
    int kutu_change=0;
    int tBest[max_path] ;

    // ensure particles stay in the boundaries
    for (int i=0; i<Particles.size() ; i++) {
        improved=0 ; // check if iterations goes better if not change leaders
        // during initializing closest boxes are assigned as target
        // if no leader assigned
        // make a selection among the possible food sources or another random location
        if (Particles[i].leader == -1) {
            double gcc_rastsal = dis(mt) ; //( rand() % 10000 )/10000.0 ;
            for (int k=0; k<max_path; k++) {
                if(gcc_rastsal>=kutu[k].p_min && gcc_rastsal<kutu[k].p_max) {

```



```

        Particles[i].leader = k;
        kutu[k].followers += 1 ;
    }
}
// linear perturbation towards the target - assuming optimum is in linear
path
for (int j=0; j<boyut; j++ ) {
    double base_distance = kutu[Particles[i].leader].x[j]-Particles[i].x[j] ;
    Particles[i].x[j] =
base_distance*dis(mt)*0.5+kutu[Particles[i].leader].x[j] ;
    // Particles.vx[j] = (limit_xup[j]-limit_xlow[j])*(0.5-dis(mt)*0.03);
    Particles[i].vx[j] = base_distance*(0.05-dis(mt)*speed_ratio);
    //      Particles[i].x[j] = (kutu[Particles[i].leader].x[j]-
speed_limit[j])*dis(mt)*0.5+kutu[Particles[i].leader].x[j] ;
}
}
for (int ii=0; ii<boyut ; ii++) {
    target.x[ii]=kutu[Particles[i].leader].x[ii];
}

// calculate new particle position using target vector, velocity and acceleration
// multiply velocity with -0.05
// so the particle will move towards target
// ax[j] is acceleration vector
for (int j=0; j<boyut ; j++) {
    ax = (Particles[i].x[j] - target.x[j])*acceleration_factor;
    Particles[i].vx[j] = ax + inertia*Particles[i].vx[j];
    // limit the velocity of particle
    if (Particles[i].vx[j] >speed_limit[j])
        Particles[i].vx[j] = speed_limit[j];
    if (Particles[i].vx[j]<-speed_limit[j])
        Particles[i].vx[j] = -speed_limit[j];
    Particles[i].x[j] += Particles[i].vx[j] ;
}

```

```

if (Particles[i].x[j]> limit_xup[j] ) {
    Particles[i].x[j]= limit_xup[j] ;
    Particles[i].vx[j] = -Particles[i].vx[j] ;
}
if (Particles[i].x[j] <limit_xlow[j]) {
    Particles[i].x[j]= limit_xlow[j] ;
    Particles[i].vx[j] = -Particles[i].vx[j] ;
}
}
// evaluate the particle at new position, send dimension of the x vector
temp_value = (*fTest)(Particles[i].x, boyut);
if (temp_value < Particles[i].pBest) {
    //cout << " Particle improved = " << Particles[i].id << endl;
    Particles[i].pBest = temp_value ;
    if (Particles[i].pBest<=kutu[Particles[i].leader].deger) {
        for (int j=0; j<boyut; j++ ) {
            Particles[i].vx[j] = decelerate*Particles[i].vx[j] ;
            kutu[Particles[i].leader].x[j]=Particles[i].x[j];
        }
    }
    // mark the improvement!
    improved = 1;
} else { // no improvement
    // particle will accelerate through the leader position
    for (int j=0; j<boyut ; j++) {
        Particles[i].vx[j] *= accelerate ;
    }
}
// if best values of the swarm is found then print the value
if (temp_value<sBestValue) {
    sBestParticle = Particles[i].id ;
}

```

```

//cout << " Swarm improved by = " << sBestParticle << endl;
sBestValue= temp_value;
cout << iteration << " , " << temp_value << " , " ;
cout.precision(16);
//      cout << (-15.0+Particles[sBestParticle].x[0]*10.0) << " , " ;
//      cout << (-3.0+6.0*Particles[sBestParticle].x[1]) << " , " ;
      for (int j=0; j<boyut; j++){
          cout << Particles[i].x[j] << " , " ;
      }
      End_time = std::chrono::system_clock::now();
      std::chrono::duration<double> diff = End_time-Start_time;
      cout << " Elapsed time = " << diff.count() << endl;
      echo_on = 1 ;
      double gcc_Best = (*fTest)(Particles[i].x, boyut);
      cout                                     <<                                     "
      << endl ;                                     << gcc_Best
      echo_on = 0 ;
      // is a better value is found food sources should be updated
      double gcc_worst = -99999.0 ;
      for (int kkk=0; kkk<max_path; kkk++) {
          if (kutu[kkk].deger > gcc_worst) {
              gcc_worst = kutu[kkk].deger ;
              kutu_change = kkk ; // Particles[i].leader ;
          }
      }
      // change box value and position with the new better solution
      // leader position will be updated according the particle position
      for (int jj =0;jj< boyut ; jj++) {
          kutu[kutu_change].x[jj] = Particles[i].x[jj] ;
          kutu[kutu_change].deger = temp_value ;
          // kutu[kutu_change].feromon = high_feromon ;

```

```

        }
    }
}

// one iteration completed

// if there is not improvement decrease counter

// if counter is below zero either change target or reset swarm positions

iteration +=1;
num_of_iter++;
if (improved < 1) {
    swarm_shuffle -= 1;
    if (swarm_shuffle <= 0) {
        // set leaders to -1 for re-assignment at next update command
        for (int iii=0; iii<Particles.size(); iii++) {
            Particles[iii].leader = -1 ;
            //cout << " social distance ....." << social_distance << endl;
        }
        for (int k=0; k<max_path ; k++) {
            kutu[k].followers = 0 ;
        }
        //cout << " New distribution !" << endl ;
        swarm_shuffle = feromon_update; // target changing count
    }
    yeni_total_feromon= 0.0 ;
    double gcc_min = 999999.0 ;
    double gcc_max = -999999.0;
    // arrange kutu ranges so the random selection possibility will change
    for (int kk=0; kk<max_path; kk++) {
        //if (kutu[kk].followers >=1) {
            if (kutu[kk].deger <= gcc_min)
                gcc_min = kutu[kk].deger ;
        }
    }
}

```

```

        if (kutu[kk].deger>=gcc_max)
            gcc_max = kutu[kk].deger ;
        //}
    }
    double ort_deger = (gcc_min + gcc_max)*0.5 ;
    double gcc_feromon = 10000.0 ;

    for (int kk=0; kk<max_path; kk++) {
        if (kutu[kk].deger<=ort_deger) {
            kutu[kk].feromon = kutu[kk].feromon * olcek ; /* (sBestValue-
            kutu[kk].deger)*(sBestValue-kutu[kk].deger)/(gcc_min*gcc_min) ;
        } else {
            kutu[kk].feromon = kutu[kk].feromon * bozulma ; /* (gcc_min-
            kutu[kk].deger)*(gcc_min-kutu[kk].deger)/(gcc_min*gcc_min) ;
        }
        yeni_total_feromon = yeni_total_feromon + kutu[kk].feromon ;
        if (kutu[kk].feromon < gcc_feromon) {
            gcc_feromon = kutu[kk].feromon ;
        }
    }
    low_feromon = gcc_feromon ;
    toplam_feromon = yeni_total_feromon ;
    // setup initial targets for the swarm
    kutu[0].p_min = 0.0;
    kutu[0].p_max = kutu[0].feromon/toplam_feromon;
    for (int kkk=1 ; kkk<max_path ; kkk++) {
        kutu[kkk].p_min = kutu[kkk-1].p_max;
        kutu[kkk].p_max = kutu[kkk].p_min+
        kutu[kkk].feromon/toplam_feromon;
    }
}
// if improvements stop re-shuffle the swarm

```

```

}

//{ draw particles on the screen in order to understand the behaviour

void draw(sf::RenderWindow& window)
{
    sf::CircleShape blob;
    sf::VertexArray line(sf::Lines,2);
    for (auto P: Particles) {
        line[0].position.x = P.x[0]*50 ; /*50;
        line[0].position.y = P.x[1]*100 ; /*100;
        line[1].position.x = P.x[0]*50 + P.vx[0]*50;
        line[1].position.y = P.x[1]*100 + P.vx[1]*100;
//        line[1].position.x = P.x[0] + P.vx[0];
//        line[1].position.y = P.x[1] + P.vx[1];
        blob.setRadius(0.5);
        blob.setFillColor(sf::Color::Red);
        blob.setPosition(P.x[0]*50-0.5, P.x[1]*100-0.5);
//        blob.setPosition(P.x[0]-5, P.x[1]-5);
        window.draw(blob) ;
        window.draw(line);
    }
    blob.setFillColor(sf::Color::Green);
    // blob.setPosition(target.x[0]*50-5, target.x[1]*100-5);
    for (int i=0; i<max_path; i++) {
        //blob.setRadius(sqrt(kutu[i].feromon)+0.1);
        //blob.setPosition(kutu[i].x[0]-5, kutu[i].x[1]-5);
        blob.setPosition(kutu[i].x[0]*50-0.5, kutu[i].x[1]*100-0.5);
        window.draw(blob) ;
    }
}

};

//} END OF STRUCT DEFINITION

```

EK-3. Turboprop motor fonksiyonu (tek şaftlı)

```
double fTurboShaftFixed(double* x, int array_size){
    int expected_size = 4; // number of input parameters
    // operating: idle maximum cruise
    // TPE331 rpm % -> 100.0 97.0 95.0
    string Nokta[3]={ "TAKE-OFF" , "CRUISE" , "LOITER"} ;
    double Ni[3] = {90.0, 87.3, 85.5} ; // Physical Engine Speed [%] tasarim noktasi
    %90, digerleri ona gore orantili
    double Zi[3] = { 0, 3810.0, 2133.6} ; // Altitude [m]
    double Mi[3] = { 0, 0.3478, 0.294} ; // Flight Mach number
    double PWRi[3]= { 953.1, 690.0, 603.0 } ; // Shaft power Requirement [hp] 953.1,
    690.0, 603.0
    double T5i[3]= { 1023.0, 1023.0, 1023.0} ; // maximum EGT temperature[K]
    double Wfi[3] = { 0.1515, 0.1077, 0.097} ; // Wf = [lb/s]
    double Pri[3] = { 10.53, 11.6, 8.6} ; // Wf = [kg/s]
    double Wai[3] = { 3.496, 2.663, 2.461} ; // Wf = [kg/s]
    double T4maxi[3] = { 1341.3, 1302.3, 1248.4} ; // maximum combustor exit
    temperature [K]
    double pr_scale = 1.0 ; //(10.2-1.0)/(11.12-1.0) ; //
    double mass_scale = 1.0 ; // compressor map mass flow scale - map design point
    206, calculated engine 100
    double comp_eff_scale = 1.0 ; ///0.84168/0.780 ;
    double TPR_max = 10.5 ; //8.6 ;
    double turb_eff_scale = 1.0 ;
    double Thrust = 0.0 ; // turboprop generates bonus thrust due to expansion in
    nozzle
    double HPT_choke = 4.5 ; // HP Turbine choking pressure
    double HD_hot = 1.33/(1.33-1.0) ;
    double HD_cold= 1.4/(1.4-1.0) ;
    //{
    double W2; ; // core engine air mass flow rate kg/s
    double CPR ; // compressor pressure ratio comes from map
```

```

double TPR ; // turbine pressure ratio comes from map
double NPR = 1.12 ; // Nozzle pressure ratio
double Wf ;
//double W3_crt_des = 0.103 ; // 3.270*sqrt(1124.33)1066.0523
// Guess constants
double W2_try ; // trial compressor mass flow for matching
double T4_try ; // trial turbine pressure ratio for matching
//
// Matching constants
double W4_turb_map ; // map value according calculated pressure ratio
double W4_turb_calculated ;
double Err_01 ;
double W8_exhaust_curve ; // nozzle characteristic curve value
double W8_exhaust_calculated ;
double Err_02 ;
//
double total_error[3] = {1.e9, 1.e9, 1.e9 } ; // errors are summed seperately for each
condition
double result = 0.0;
//}
// TPE331 baseline component information
double E_inlet = 1.0; // 0.98 ; // inlet diffuser efficiency
double Ecomp = 0.856 ; // compressor efficiency comes from map
double Eturb = 0.86 ; // constant
double Ecomb = 0.975 ; // combustion efficiency in percent
double Educt = (1.0-0.025) ; // 0.07 pressure loss at exhaust duct
double dP4 = (1.0-0.93) ; // PR_comb - combustor pressure drop in percent
double Emech = 0.98 ; // mechanical efficiency
double CD_nozzle = 0.99 ;
double mb1 = 0.0271 ; // bleed 1
double mb2 = 0.0374 ; // bleed 2

```



```
double mb = mb1+mb2 ; // blade cooling + NGV cooling - bleed_air/w2 ratio = 1
percent of main flow is bleed
```

```
double mf ; //fuel/air ratio wf/W2
```

```
double accessory_work =0.0 ; // accessory work and other power take offs
```

```
double A2 = 30.1*645.16/1.e6 ; // mm2 A2 compressor inlet M2=0.5
```

```
double A25= 12.5*645.16/1.e6 ; // mm2 A2 stage 2 inlet M2=0.4
```

```
double A3 = 6.4*645.16/1.e6 ; // mm2 A2 combustor inlet M3= 0.4
```

```
double A4_0 = 3079.56/1.e6 ; // mm2 A2 ngv inlet /28.5*645.16/1.e6/ M4 = 0.1
```

```
double A5 = 71.4*645.16/1.e6 ; // mm2 A2 turbin exit M5= 0.3
```

```
double A7 = 73.3*645.16/1.e6 ; // mm2 A7 duct cross section M7=0.3
```

```
double A8_0 = 52569.77/1.e6 ; // 55.6*645.16/1.e6 // 37133.0 ; // mm2 A8
Exhaust area
```

```
int MTO = 1 ;
```

```
int MCR = 0 ;
```

```
int LOT = 0 ;
```

```
////////////////////////////////////
```

```
int design_mode = 0 ; // design_mode = 1, operating condition = 0
```

```
// design_mode = 0, operating condition = 1
```

```
if (design_mode==0) {
```

```
    x[2]= 0.5 ;
```

```
    x[3]= 0.5 ;
```

```
}
```

```
////////////////////////////////////
```

```
// durum specifies engine operating point 0: Take Off, 1: Cruise, 2:Loiter
```

```
//
```

```
for (int durum=0; durum<1 ; durum++) {
```

```
//{ durumlari hesaplama loop u
```

```
    switch (durum) {
```

```
        case 0:
```

```
            MTO = 1; MCR = 0; LOT= 0;
```

```
            break;
```

```
        case 1:
```

```

        MTO = 0; MCR = 1; LOT = 0;
        break ;
    case 2:
        MTO = 0; MCR = 0; LOT = 1;
        break ;
    }

    // Calculate P0, T0 according altitude standard atmosphere
    double Pamb, Tamb ;
    Altitude_T0P0(Zi[durum], &Tamb, &Pamb) ;
    double M0 = Mi[durum] ;
    double PWR = PWRi[durum] ;
    double T5_limit = T5i[durum] ;
    double speed = Ni[durum] ;
    double V0 = M0 * sqrt(gama_cold*R_air*Tamb) ;
    double T0 = Tamb * (1.0 + (gama_cold-1.0)*0.5*M0*M0) ;
    double P0 = Pamb * pow(T0/Tamb,(gama_cold/(gama_cold-1)));
    double P1 = P0*E_inlet; // inlet losses
    double T1 = T0; // total temperature conserved
    double P2 = P1*1.0; // pressure loss at inlet ducts
    double T2 = T1;
    double sqr_teta2 = sqrt(T2/288.15) ;
    double speed_crt = speed/sqr_teta2 ;

    //{ find compressor speed curve - lower curve no: i0, upper curve no: i1,
    interpolation coefficient: tt_
    int i0 = 0;
    int i1 = 1;
    double tt_ = 0.0 ;

    for (int i=1;i<Ns_comp;i++) {
        if (speed_crt>=Speed_comp[i-1] && speed_crt<Speed_comp[i]) {
            i0 = i-1 ;
            i1 = i ;

```

tt_=(speed_crt-Speed_comp[i0])/(Speed_comp[i1]-Speed_comp[i0]); // tt_
is the ratio between the speed curves

}

}

// find turbine speed curve - lower curve no: it0, upper curve no: it1, interpolation
coefficient: ttt

int it0 = 0;

int it1 = 1;

double ttt = 0.0 ;

for (int i=0;i<Ns_turb;i++) {

if (speed>=Speed_turb[i] && speed<Speed_turb[i+1]) {

it0 = i ;

it1 = i+1 ;

ttt=(speed-Speed_turb[it0])/(Speed_turb[it1]-Speed_turb[it0]); // tt_ is the
ratio between the speed curves

}

}

//}

/**/**/**

double Rline_try = x[0] ; // try a rline on compressor speed line

double T4_try = 800+x[1]*(T4maxi[0]-800) ; // maximum and minimum
temperature range

/**/**/**

// mass_scale = 1.0 + (x[2]-0.5)/2.5 ;

// pr_scale = 1.0 + (x[3]-0.5)/5.0 ;

double A2_ratio = 1.0 ; // 0.8+ x[3]*0.4 ; // 0.874063 ; // A2_ratio - how much
compressor should be bigger? 1.085

double A4_ratio = (0.5+x[2]) ; // 0.649211; // Turbin mapi scale etmek ile
A4_ratio degistirmek ayni etkiyi olusturuyor / denedim gordum

double A4 = A4_0 *A4_ratio ;

double A8 = (A8_0)*(0.5+x[3]) ; //

/**/**/**

//{ compressor operating point trial

int j0 = (int)floor(Rline_try/0.1) ; // data number on the speed line

```

int j1 = j0+1 ;

if (j1==Data_comp) j1 = j0 ;

double Mdot0 = Mdot_comp_map[i0][j0]+(Mdot_comp_map[i1][j0]-
Mdot_comp_map[i0][j0])*tt_;

double Mdot1 = Mdot_comp_map[i0][j1]+(Mdot_comp_map[i1][j1]-
Mdot_comp_map[i0][j1])*tt_;

double PR0 = PR_comp_map[i0][j0]+(PR_comp_map[i1][j0]-
PR_comp_map[i0][j0])*tt_;

double EFF0 = Eff_comp_map[i0][j0]+(Eff_comp_map[i1][j0]-
Eff_comp_map[i0][j0])*tt_;

double PR1 = PR_comp_map[i0][j1]+(PR_comp_map[i1][j1]-
PR_comp_map[i0][j1])*tt_;

double EFF1 = Eff_comp_map[i0][j1]+(Eff_comp_map[i1][j1]-
Eff_comp_map[i0][j1])*tt_;

double fraction = Rline_try-j0*0.1 ;

W2_try = Mdot0 + fraction * (Mdot1-Mdot0) ; // W2 trial - corrected mass flow
rate W2_crt

CPR = PR0 + fraction * (PR1-PR0) ; //

Ecomp = EFF0 + fraction * (EFF1-EFF0) ; //

Ecomp = Ecomp *comp_eff_scale ;

// cout << " Rline_try = " << Rline_try << " , Mdot = " << W2_try << " , CPR = "
<< CPR << endl ;

//} compressor point defined

W2 = A2_ratio*W2_try * (P2/101325.0)/sqr_teta2 * mass_scale ; // physical mas
flow

CPR = pr_scale*(CPR-1.0)+1.0 ; // scaling due to different impeller diameters

W2_try = A2_ratio*W2_try * mass_scale ;

double cp_2 = cp_cold ; // Cp_gas(T2,0.0) ;

double gama_2 = 1.39310 ; // (cp_2)/(cp_2-R_air) ;

double T3_ideal = T2*pow((CPR),((gama_2-1.0)/gama_2)) ;

double T3_real = (T3_ideal-T2)/Ecomp + T2;

double P3 = CPR * P2 ;

double T3 = T3_real ;

double W3 = W2*(1-mb) ;

// double W3_crt = W3 * sqrt(T3)/P3 ;

```

```

double FAR3 = 0.0 ;
double cp_3 = cp_cold ; // Cp_gas(T3,FAR3) ;
double comp_work = W2*(T3_real-T2)*cp_3 ; // necessary work for compressor
double turb_work = (comp_work + (PWR + accessory_work)*hp_to_watt)/Emech
;

double P4 = P3 * (1.0-dP4) ; // dP4=7/100 constant
double T4 = T4_try ; // try T4 between Tmin - Tmax
double cp_4 = cp_hot ; // Cp_gas(T4,0.0) ;
Wf = W3*(T4*cp_4 - T3_real*cp_3)/(LHV*Ecomb-T4*cp_4);
mf = Wf/W2 ;
if (Wf<=0.001) {
    return 1000000.0 ;
}
double W4 = W2*(1+mf-mb) ; // physical mass flow - not corrected
// double FAR4 = mf ; // approximate value
// cp_4 = Cp_gas(T4,FAR4) ;
double gama_4 = 1.29933 ; //(cp_4)/(cp_4-R_air) ;

double T4_burned = (W3*T3_real*cp_3 + Wf*LHV*Ecomb)/(W4*cp_4) ; // if wf
is burned temperature would be T4_burned

// depending on the T4_burned Wfuel should be updated
Wf = Wf * 0.5*(T4_burned+T4)/(T4);
mf = Wf/W2 ;
W4 = W2*(1+mf-mb) ;

// double turb_work = comp_work/Emech+accessory_work ; // in order to support
compressor PTO and others are neglected but can be included

// calculate T5 for work balance
// fixed shaft turboshaft can expand from P4 to P0*NPR;
double T5_real = T4- turb_work/(W4*cp_4) ;
double T5_ideal = T4-(T4-T5_real)/Eturb ;
if (T5_ideal<T2)
    return 1e6 ;
TPR = pow((T4/T5_ideal), gama_4/(gama_4-1.0)) ;

```

```

        // pressure ratio according to work balance is TPR, if is it different from TPR_try
        generate error

        if (TPR>TPR_max) {
            TPR = TPR_max ;
        }

        int verify = 0 ;
        double delta_pr = 0 ;
        // look up Turbine map
        // for selected % speed turbine
        // for (int i=0; i<Data_turb; i++) {
        //     double tpr_alt = PR_turb_map[it0][i] + ttt * (PR_turb_map[it1][i]-
        PR_turb_map[it1][i]) ;
        //     double tpr_ust = PR_turb_map[it0][i+1] + ttt * (PR_turb_map[it1][i+1]-
        PR_turb_map[it0][i+1]) ;
        //     delta_pr = tpr_ust-tpr_alt ;
        //     if ((TPR>=tpr_alt)&&(TPR<tpr_ust)) {
        //         double mdot_alt = Mdot_turb_map[it0][i] + ttt * (Mdot_turb_map[it1][i]-
        Mdot_turb_map[it0][i]) ;
        //         double mdot_ust = Mdot_turb_map[it0][i+1]+ ttt *
        (Mdot_turb_map[it1][i+1]-Mdot_turb_map[it0][i+1]) ;
        //         //double EFF_alt = Eff_turb_map[it0][i] + ttt * (Eff_turb_map[it1][i]-
        Eff_turb_map[it0][i]) ;
        //         //double EFF_ust = Eff_turb_map[it0][i+1]+ ttt * (Eff_turb_map[it1][i+1]-
        Eff_turb_map[it0][i+1]) ;
        //         //E_turb_initial = 0.90 ; // EFF_alt + (EFF_ust-EFF_alt)*((TPR-
        tpr_alt)/(delta_pr)) ;
        //         W4_turb_map = mdot_alt + (mdot_ust-mdot_alt)*((TPR-tpr_alt)/(delta_pr))
        ; // physical mass flow rate - not the corrected one!
        //         verify = 1 ;
        //     }
        // }
        // if (verify==0) {
        //     if (W4_turb_map <= Mdot_turb_map[it0][0] ) {
        //         W4_turb_map = Mdot_turb_map[it0][0] ;
        //         E_turb_initial = 0.90 ; // Eff_turb_map[it0][0] ;

```

```

//      }
//      if (W4_turb_map >= Mdot_turb_map[it0][Data_turb-1]) {
//          W4_turb_map = Mdot_turb_map[it0][Data_turb-1] ;
//          E_turb_initial = 0.90 ; // Eff_turb_map[it0][Data_turb-1] ;
//      }
//  }

// calculation according to Turbine Characteristic Curve
double TPRC = pow((1.0+(1.33-1.0)/2.0),HD_hot) ;

double TPR_mod = (TPRC-1.)/(HPT_choke-1.) * (TPR-1.0) + 1.0 ; // scale turbine
to choke at PR=4.5

double W4_allowed ;
if (TPR_mod>=TPRC)
    TPR_mod = TPRC ;

double W4_crt = (1./TPR_mod)*sqrt(2.0*HD_hot*(1.0-
pow(1./TPR_mod,(1.0/HD_hot))))/((1.0-(1.0-pow(1./TPR_mod,(1.0/HD_hot)))))) ;
W4_allowed = W4_crt*(A4)/sqrt(R_air) ; // allowed corrected flow
W4_turb_calculated = W4*sqrt(T4)/P4; // calculated corrected flow
// E R R O R # 1 :

// Turbine capacity difference in percent

Err_01 = abs(W4_turb_calculated-W4_allowed)/W4_turb_calculated*100.0 ; //
should be less than 0.05 %

double P5 = P4/TPR ;

double P5_over_P0 = (1.0/TPR)*(P4/P3)*(P3/P2)*(P2/P1)*(P1/P0) ; // check
nozzle pressure is enough for thrust

//double T5_ideal = T4/pow((TPR),(gama_hot-1)/gama_hot) ;
//T5_real = T4 - Eturb*(T4-T5_ideal) ;

double FAR5 = Wf/W2 ;

double cp_5 = cp_hot ; //Cp_gas(T5_real,FAR5) ;

double gama_5 = gama_hot ; // (cp_5)/(cp_5-R_air) ;

double W5 = W2*(1+mf) ;

double T5 = T5_real ;

double T8 = (T5*cp_5*W4+T3*cp_3*W2*mb)/(W5*cp_5) ; // cooling air returns
to flowpath and decrease gas temperature

```

```

double P8 = P5*Educt ; // simple convergent nozzle - 0.025 loss at jet pipe - fully
expands into atmosphere

double W8 = W5 ; // assuming bleed returned in to the main flow

double P8s=Pamb, T8s=Tamb ;

double W8_exhaust_curve ;

double A8_star ;

double V8 , M8 ;

double NPR_1 = Pamb/P8 ; // nozzle da gerceklesen basinc oraninin tersi
if (P5_over_P0<=1) {
    return 1e6;
}
if (design_mode==1 && (abs(1./NPR_1-1.12)/1.12)>0.02) {
    return 1e6;
}
if (design_mode==0 && (1./NPR_1)<1.01) {
    return 1e6;
}

// A8_star = W8/((P8)/sqrt(T8))/Q_function(1.0, gama_5 , R_air); // 0.32 mach
number is coming from hand calculation

// A8 = A8_try ; // fixed at off-design
if ((1./NPR_1)>=TPRC)
    NPR_1 = 1./TPRC ;

double W8_allowed ;

double mnqrt ;

mnqrt =(NPR_1)*sqrt(2*HD_hot*CD_nozzle*(1.0-pow(NPR_1,(1.0/HD_hot)))/
((1.0-CD_nozzle*(1.0-pow(NPR_1,(1.0/HD_hot)))))) ;

// double mnqrt = (D94) * KAREKoK(2 * HD * $E$90 * (1-D94^(1/HD)))/((1-
$E$90*(1-D94^(1/HD)))) ;

W8_allowed = mnqrt*(A8)/sqrt(R_air) ; // allowable nozzle mass flow rate -
physical

W8_exhaust_calculated = W8*sqrt(T8)/(P8) ; // mult with 0.001 in order to turn
P8 from Pa -> kPa

//cout << " W8_allowed = " << W8_allowed << " , W8 Actual = " << W8 << endl;

// not choked nozzle

```



```

P8s = Pamb ;
T8s = T8*pow((P8s/P8),(gama_5-1.0)/gama_5) ;
double ro8 = P8s/(R_air*T8s) ;
//M8 = sqrt(2.0*(T8/T8s-1.0)/(gama_hot-1.0));
V8 = W8/(ro8*A8) ; // sqrt(2.0 * CD_nozzle * (T8-T8s) * 1000.0); // //
M8*sqrt(gama_hot*R_air*T8s)*CD_nozzle ; // exhaust velocity in m/s --
M8 = V8/sqrt(gama_5*R_air*T8s) ;
Thrust = W8*V8 - W2*V0; // Net thrust
// E R R O R # 2 :
// Turbine capacity difference in percent
if (design_mode == 1)
    Err_02 = abs(W8_exhaust_calculated-
W8_allowed)/W8_exhaust_calculated*100.0 ; // should be less than 0.05 %
    if (design_mode == 0) {
        Err_02 = (W8_exhaust_calculated-W8_allowed)/W8_exhaust_calculated*100.0
; // should be less than 0.05 %
        if (Err_02<0) {
            Err_02 = 0; // if calculates is less than allowable no error occurs ; compressor
can easily pass the air flow
        }
    }
// calculations completed now objective function and penalty calculations
double pwr_penalty=0.0; //
double tc8 = (T5_real+T8s)*0.5 ; // thermocouple sens the mean temperature
double TSFC = mf/((1+mf)*V8) ;
//double FuelConsumption = Thrust*TSFC ;
double Eff_overall = (0.5*(1+mf)*V8*V8)/(mf*LHV*Ecomb) ;
double specific_thrust = Thrust/W2/9.81 ;
//
double hedef0 = Err_01 ;
double hedef2 = Err_02 ; //abs(T4_burned-T4) ; //
double hedef3 = abs(1./NPR_1-1.12)/1.12*100.0;

```

```

double   hedef4   =   (1-Ecomp)*100   ;   //   abs(Wf/lb_to_kg-
Wfi[durum])/Wfi[durum]*1000.0+ abs(CPR-Pri[durum])/Pri[durum]*1000.0   ;   // +(1-
Eturb)*100;

if (design_mode == 1) {
    total_error[durum] = max(hedef0,hedef2)+abs(1./NPR_1-1.12)/1.12;
}

if (design_mode == 0) {
    total_error[durum] = hedef0;
}

if (echo_on >0) {
    RESULT << num_of_iter << ", " << CPR << ", " << W2_try << ", " << TPR << ",
" << NPR << ", " << T4 << ", " << P4 << ", " << W4_crt << ", " << A4 << ", " <<
W4_allowed << endl;

    if (design_mode == 0) {
        cout << "#####" << endl;
        cout << "### OFF - DESIGN MODE ###" << endl;
    } else {
        cout << "::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::"
<< endl;

        cout << " ::: DESIGN MODE ::: " << endl;
    }

    cout << " Fixed shaft Calisma noktasi : " << Nokta[durum] << endl;

    cout << "-----"
-----" << endl;

    cout << "Altitude = " << Zi[durum] << ", M0 = " << Mi[durum] << endl;

    cout << " CPR = " << CPR << ", TPR = " << TPR << ", TPR_mod = " <<
TPR_mod << ", NPR = " << 1.0/NPR_1 << endl;

    cout << " W2_crt = " << W2_try << ", tt_ = " << tt_ << ", speed = " << speed <<
", Speed_crt = " << speed_crt << endl;

    cout << " W4 = " << W4 << ", W4_allowed = " << W4_allowed*P4/sqrt(T4) << "
, Nozzle_allowed = " << W8_allowed*P8/sqrt(T8) << ", W8 = " << W8 << endl;

    cout << " W2_try [kg/s] = " << W2_try << ", W2 [kg/s] = " << W2 << ", Wfuel
[g/s] = " << Wf*1000.0 << endl;

```

```

        cout << " -----
-----" << endl ;

        cout << " A2_ratio = " << A2_ratio << " ,A4_ratio = " << A4_ratio << " , A8
[mm2] = " << A8*1000000.0 << endl;

        cout << " -----
-----" << endl ;

        cout << " P0 = " << P0 << " , P2 = " << P2 << " , P3 = " << P3 << " , Pamb = " <<
Pamb <<endl ;

        cout << " P4 = " << P4 << " , P5 = " << P5 << " , P8 = " << P8 << " , P8s = " <<
P8s << endl;

        cout << " -----
-----" << endl ;

        cout << " T0 = " << T0 << " , T2 = " << T2 << " T3_ideal = " << T3_ideal << " ,
T3_real = " << T3_real << " , Tamb = " << Tamb << endl;

        cout << " T4 = " << T4 << " ,T5_real = " << T5_real << " , T5_ideal = " <<
T5_ideal << " , T8 = " << T8 << " , T8s = " << T8s << endl;

        cout << " -----
-----" << endl ;

        cout << " Comp work = " << comp_work*watt_to_hp << " | Turb work = " <<
turb_work*watt_to_hp << " [hp] " << endl ;

        cout << " -----
-----" << endl ;

        cout << " Required PWR= " << PWRi[durum] << " | calculated PWR = " << PWR
<< " , bonus thrust = " << Thrust << endl;

        cout << " -----
-----" << endl ;

        cout << " EGT_limit = " << T5i[durum] << " | TC8 = " << tc8 << endl;

        cout << " -----
-----" << endl ;

        cout << "Ecomp = " << Ecomp << " ,Ecomb = " << Ecomb << " , dP4 = " << dP4
<< " , mb = " << mb << " , TurbEff = " << Eturb << endl;

        cout << "Eff_overall = " << Eff_overall << " , TSFC = " << TSFC*1000.0 << " ,
Thrust= " << Thrust <<endl;

        cout << " V8 [m/s] = " << V8 << " , M8 = " << M8 << endl ;

        cout << " hedef0 = " << hedef0 << " , hedef2 = " << hedef2 << " , hedef3 = " <<
hedef3 << " , hedef4 = " << hedef4 << endl;

        cout << " -----
-----" << endl ;

```


EK-5. Turboşaft/turboprop motor fonksiyonu (iki şaftlı)

```
double fTurboShaftFree(double* x, int array_size){
    int expected_size = 5; // number of input parameters
    // FREE shaft version of the TPE331 engine
    // operating: idle maximum cruise
    // TPE331 rpm % -> 100.0 97.0 95.0
    string Nokta[3]={ "TAKE-OFF" , "CRUISE" , "LOITER"} ;
    double Ni[3] = {90.0, 87.3, 85.5} ; // Physical Engine Speed [%] tasarim noktasi
    %90, digerleri ona gore orantili
    double Zi[3] = { 0, 3810.0, 2133.6} ; // Altitude [m]
    double Mi[3] = { 0, 0.3478, 0.294} ; // Flight Mach number
    double PWRi[3]= { 953.1, 690.0, 603.0 } ; // Shaft power Requirement [hp] 953.1,
    690.0, 603.0
    double T5i[3]= { 1023.0, 1023.0, 1023.0} ; // maximum EGT temperature[K]
    double Wfi[3] = { 0.1515, 0.1077, 0.097} ; // Wf = [lb/s]
    double Pri[3] = { 10.53, 11.6, 8.6} ; // Wf = [kg/s]
    double Wai[3] = { 3.496, 2.663, 2.461} ; // Wf = [kg/s]
    double T4maxi[3] = { 1341.3, 1302.3, 1248.4} ; // maximum combustor exit
    temperature [K]
    double pr_scale = 1.0 ; //(10.2-1.0)/(11.12-1.0) ; //
    double mass_scale = 1.0 ; // compressor map mass flow scale - map design point
    206, calculated engine 100
    double comp_eff_scale = 1.0 ; ///0.84168/0.780 ;
    double TPR_max = 10.5 ; //8.6 ;
    double turb_eff_scale = 1.0 ;
    double Thrust = 0.0 ; // turboprop generates bonus thrust due to expansion in
    nozzle
    double HPT_choke = 3.5 ; // HP Turbine choking pressure
    double LPT_choke = 3.5 ; // LP Turbine choking pressure
    double HD_hot = 1.33/(1.33-1.0) ;
    double HD_cold= 1.4/(1.4-1.0) ;
    //{
    double W2; ; // core engine air mass flow rate kg/s
```

```

double CPR ; // compressor pressure ratio comes from map
double TPR ; // turbine pressure ratio comes from map
double NPR = 1.12 ; // Nozzle pressure ratio
double Wf ;
// Guess constants
double W2_try ; // trial compressor mass flow for matching
double T4_try ; // trial turbine pressure ratio for matching
//
// Matching constants
// HPT
double W4_crt ;
double W4_allowed ; // map value according calculated pressure ratio
double W4_turb_calculated ;
double Err_01 ;
// LPT
double W45_crt ;
double W45_allowed ; //
double W45_turb_calculated ; //
double Err_02 ;
// EXHAUST
double W8_crt ;
double W8_allowed ; // nozzle characteristic curve value
double W8_exhaust_calculated ;
double Err_03 ;
//
double total_error[3] = {1.e9, 1.e9, 1.e9 };
double result = 0.0;
//}

// TPE331 baseline component information
double E_inlet = 1.0; // 0.98 ; // inlet diffuser efficiency
double Ecomp = 0.856 ; // compressor efficiency comes from map

```

```

double Eturb = 0.86 ; // constant
double Ecomb = 0.975 ; // combustion efficiency in percent
double Educt = (1.0-0.025) ; // 0.07 pressure loss at exhaust duct
double dP4 = 0.07 ; // PR_comb - combustor pressure drop in percent
double Emech = 0.98 ; // mechanical efficiency
double CD_nozzle = 0.99 ;
double mb1 = 0.0271 ; // bleed 1
double mb2 = 0.0374 ; // bleed 2
double mb = mb1+mb2 ; //
double mf ; //fuel/air ratio wf/W2
double accessory_work = 0.0 ; // accessory work and other power take offs
double A2 = 30.1*645.16/1.e6 ; // mm2 A2 compressor inlet M2=0.5
double A25 = 12.5*645.16/1.e6 ; // mm2 A2 stage 2 inlet M2=0.4
double A3 = 6.4*645.16/1.e6 ; // mm2 A2 combustor inlet M3= 0.4
double A4_0 = 3519.6/1.e6 ; // mm2 A2 ngv inlet /28.5*645.16/1.e6/ M4 = 0.1
double A45_0 = 12254.9/1.e6 ; // mm2 A2 ngv inlet /28.5*645.16/1.e6/ M4 = 0.1
double A5 = 71.4*645.16/1.e6 ; // mm2 A2 turbin exit M5= 0.3
double A7 = 73.3*645.16/1.e6 ; // mm2 A7 duct cross section M7=0.3
double A8_0 = 37506.8/1.e6 ; // 55.6*645.16/1.e6 // 37133.0 ; // mm2 A8 Exhaust
area
// initial values
int MTO = 1 ;
int MCR = 0 ;
int LOT = 0 ;
////////////////////////////////////
int design_mode = 1; // design_mode = 1, operating condition = 0
// design_mode = 0, operating condition = 1
if (design_mode==0) {
    x[2]= 0.5 ; // A4_ratio = 0.5+x[2]
    x[3]= 0.5 ; // A45_ratio = 0.5+x[3]
    x[4]= 0.5 ; // A8_ratio = 0.5+x[4]
}

```

```

////////////////////////////////////
// durum specifies engine operating point 0: Take Off, 1: Cruise, 2:Loiter
//
    for (int durum=0; durum<1 ; durum++) {
//{ durumlari hesaplama loop u
        switch (durum) {
            case 0:
                MTO = 1; MCR = 0; LOT= 0;
                break;
            case 1:
                MTO = 0; MCR = 1; LOT = 0;
                break ;
            case 2:
                MTO = 0; MCR = 0; LOT = 1;
                break ;
        }
// Calculate P0, T0 according altitude standard atmosphere
        double Pamb, Tamb ;
        Altitude_T0P0(Zi[durum], &Tamb, &Pamb) ;
        double M0 = Mi[durum] ;
        double PWR = PWRi[durum] ;
        double T5_limit = T5i[durum] ;
        double speed = Ni[durum] ;
        double V0 = M0 * sqrt(gama_cold*R_air*Tamb) ;
        double T0 = Tamb * (1.0 + (gama_cold-1.0)*0.5*M0*M0) ;
        double P0 = Pamb * pow(T0/Tamb,(gama_cold/(gama_cold-1)));
        double P1 = P0*E_inlet; // inlet losses
        double T1 = T0; // total temperature conserved
        double P2 = P1*1.0; // pressure loss at inlet ducts
        double T2 = T1;
        double sqr_teta2 = sqrt(T2/288.15); // assuming speed lines are scaled REVIEW
THIS !!!

```



```

double speed_crt = speed/sqr_teta2 ;

//{ find compressor speed curve - lower curve no: i0, upper curve no: i1,
interpolation coefficient: tt_

int i0 = 0;
int i1 = 1;
double tt_ = 0.0 ;
for (int i=1;i<Ns_comp;i++) {
    if (speed_crt>=Speed_comp[i-1] && speed_crt<Speed_comp[i]) {
        i0 = i-1 ;
        i1 = i ;
        tt_ = (speed_crt-Speed_comp[i0])/(Speed_comp[i1]-Speed_comp[i0]); // tt_
is the ratio between the speed curves
    }
}

// find turbine speed curve - lower curve no: it0, upper curve no: it1, interpolation
coefficient: ttt

int it0 = 0;
int it1 = 1;
double ttt = 0.0 ;
for (int i=0;i<Ns_turb;i++) {
    if (speed>=Speed_turb[i] && speed<Speed_turb[i+1]) {
        it0 = i ;
        it1 = i+1 ;
        ttt = (speed-Speed_turb[it0])/(Speed_turb[it1]-Speed_turb[it0]); // tt_ is the
ratio between the speed curves
    }
}

//}

/**/**/**

double Rline_try = x[0] ; // try a rline on compressor speed line

double T4_try = 800+x[1]*(T4maxi[0]-800) ; // maximum and minimum
temperature range

/**/**/**

```

```

double A2_ratio = 1.0 ; // 0.8+ x[3]*0.4 ; // 0.874063 ; // A2_ratio - how much
compressor should be bigger? 1.085

double A4_ratio = (0.5+x[2]) ; // 0.649211; // Turbin mapi scale etmek ile
A4_ratio degistirmek ayni etkiyi olusturuyor / denedim gordum

double A4    = A4_0 * A4_ratio ;

double A45_ratio = (0.5+x[3]) ;

double A45    = A45_0*A45_ratio ; // LPT reference area

double A8     = A8_0 *(0.5+x[4]) ; //

/**/**/**

//{ compressor operating point trial

int j0 = (int)floor(Rline_try/0.1) ; // data number on the speed line

int j1 = j0+1 ;

if (j1==Data_comp) j1 = j0 ;

double Mdot0    = Mdot_comp_map[i0][j0]+(Mdot_comp_map[i1][j0]-
Mdot_comp_map[i0][j0])*tt ;

double Mdot1    = Mdot_comp_map[i0][j1]+(Mdot_comp_map[i1][j1]-
Mdot_comp_map[i0][j1])*tt ;

double PR0      = PR_comp_map[i0][j0]+(PR_comp_map[i1][j0]-
PR_comp_map[i0][j0])*tt ;

double EFF0     = Eff_comp_map[i0][j0]+(Eff_comp_map[i1][j0]-
Eff_comp_map[i0][j0])*tt ;

double PR1      = PR_comp_map[i0][j1]+(PR_comp_map[i1][j1]-
PR_comp_map[i0][j1])*tt ;

double EFF1     = Eff_comp_map[i0][j1]+(Eff_comp_map[i1][j1]-
Eff_comp_map[i0][j1])*tt ;

double fraction = Rline_try-j0*0.1 ;

W2_try = Mdot0 + fraction * (Mdot1-Mdot0) ; // W2 trial - corrected mass flow
rate W2_crt

CPR  = PR0 + fraction * (PR1-PR0)    ; //

Ecomp = EFF0 + fraction * (EFF1-EFF0) ; //

Ecomp = Ecomp *comp_eff_scale ;

// cout << " Rline_try = " << Rline_try << " , Mdot = " << W2_try << " , CPR = "
<< CPR << endl ;

//} compressor point defined

W2_try = A2_ratio*W2_try * mass_scale ;

```

```

W2 = W2_try * (P2/101325.0)/sqr_teta2 ; // physical mass flow
CPR = pr_scale*(CPR-1.0)+1.0 ; // scaling due to different impeller diameters
double cp_2 = cp_cold ; // Cp_gas(T2,0.0) ;
double gama_2 = 1.39310 ; // (cp_2)/(cp_2-R_air) ;
double T3_ideal = T2*pow((CPR),((gama_2-1.0)/gama_2)) ;
double T3_real = (T3_ideal-T2)/Ecomp + T2;
double P3 = CPR * P2 ;
double T3 = T3_real ;
double W3 = W2*(1-mb) ;
// double W3_crt = W3 * sqrt(T3)/P3 ;
double FAR3 = 0.0 ;
double cp_3 = cp_cold ; // Cp_gas(T3,FAR3) ;
double comp_work = W2*(T3_real-T2)*cp_3 ; // necessary work for compressor
double turb_work = (comp_work + accessory_work*hp_to_watt)/Emech ;
double P4 = P3 * (1.0-dP4) ; // dP4=7/100 constant
double T4 = T4_try ; // try T4 between Tmin - Tmax
double cp_4 = cp_hot; // Cp_gas(T4,0.0) ;
Wf = W3*(T4*cp_4 - T3_real*cp_3)/(LHV*Ecomb-T4*cp_4);
mf = Wf/W2 ;
if (Wf<=0.001) {
    return 1000000.0 ;
}
double W4 = W2*(1+mf-mb) ; // physical mass flow - not corrected
// double FAR4 = mf ; // approximate value
// cp_4 = Cp_gas(T4,FAR4) ;
double gama_4 = 1.29933 ; //(cp_4)/(cp_4-R_air) ;
double T4_burned = (W3*T3_real*cp_3 + Wf*LHV*Ecomb)/(W4*cp_4) ; // if wf
is burned temperature would be T4_burned
// depending on the T4_burned Wfuel should be updated
Wf = Wf * 0.5*(T4_burned+T4)/(T4);
mf = Wf/W2 ;
W4 = W2*(1+mf-mb) ;

```

```

double T45_real = T4- turb_work/(W4*cp_4) ;
double T45_ideal = T4-(T4-T45_real)/Eturb ;
if (T45_ideal<T2)
    return 1e6 ;
TPR = pow((T4/T45_ideal), gama_4/(gama_4-1.0)) ;
// pressure ratio according to work balance is TPR, if it is different from TPR_try
generate error
if (TPR>TPR_max) {
    TPR = TPR_max ;
}
// calculation according to Turbine Characteristic Curve
double TPRC = pow((1.0+(gama_hot-1.0)/2.0),HD_hot) ;
double TPR_mod = (TPRC-1.)/(HPT_choke-1.) * (TPR-1.0) + 1.0 ; // scale turbine
to choke at PR=4.5
if (TPR_mod>=TPRC)
    TPR_mod = TPRC ;
W4_crt =(1./TPR_mod)*sqrt(2.0*HD_hot*(1.0-pow(1./TPR_mod,(1.0/HD_hot
))))/((1.0-(1.0-pow(1./TPR_mod,(1.0/HD_hot)))))) ;
W4_allowed = W4_crt*(A4)/sqrt(R_air) ; // allowed corrected flow
W4_turb_calculated = W4*sqrt(T4)/P4; // calculated corrected flow
// E R R O R # 1 :
// Turbine capacity difference in percent
Err_01 = abs(W4_turb_calculated-W4_allowed)/W4_turb_calculated*100.0 ; //
should be less than 0.05 %
double P45 = P4/TPR ;
double FAR45 = Wf/W2 ;
double cp_45 = cp_hot ; //Cp_gas(T5_real,FAR5) ;
double gama_45 = 1.33481 ; // (cp_5)/(cp_5-R_air) ;
//HD = gama_5/(gama_5-1.0) ;
double W45 = W2*(1+mf) ;
double T45 = (T45_real*cp_45*W4+T3*cp_3*W2*mb)/(W45*cp_45) ; // cooling
air returns to flowpath and decrease gas temperature
// free-shaft turboshaft can expand from P45 to P0*NPR;

```

```

// (PWR + accessory_work)*hp_to_watt
double T5_real = T45-PWR*hp_to_watt/(W45*cp_hot); // power required by free
turbine
double T5_ideal = T45-(T45-T5_real)/Eturb ;
double P5 = P45*pow((T5_ideal/T45),gama_45/(gama_45-1.0)) ;
double LPTPR = P45/P5 ;
double T5 = T5_real ;
double PWR_calculated = (W45*cp_hot*(T45-T5))/hp_to_watt ;
// pressure ratio according to work balance is TPR, if is it different from TPR_try
generate error
if (LPTPR>TPR_max) {
    LPTPR = TPR_max ;
}
// calculation according to Turbine Characteristic Curve
double LPTPR_mod = (TPRC-1.)/(LPT_choke-1.) * (LPTPR-1.0) + 1.0 ; // scale
turbine to choke at PR=4.5
if (LPTPR_mod>=TPRC)
    LPTPR_mod = TPRC ;

W45_crt = (1./LPTPR_mod)*sqrt(2*HD_hot*(1.0-
pow(1./LPTPR_mod,(1.0/HD_hot))))/((1.0-(1.0-pow(1./LPTPR_mod,(1.0/HD_hot)))));
W45_allowed = W45_crt*(A45)/sqrt(R_air) ; // physical mass flow rate
W45_turb_calculated = W45*sqrt(T45)/P45 ; // calculated
// E R R O R # 2 :
// Turbine capacity difference in percent
Err_02 = abs(W45_turb_calculated-W45_allowed)/W45_turb_calculated*100.0 ;
// should be less than 0.05 %

double T8 = T5 ;

double P8 = P5*Educt ; // simple convergent nozzle - 0.025 loss at jet pipe - fully
expands into atmosphere
double W8 = W45 ; // assuming bleed returned in to the main flow
double P8s=Pamb, T8s=Tamb ;
double A8_star ;
double V8 , M8 ;
double NPR_1 = Pamb/P8 ; // nozzle da gerceklesen basinc oraninin tersi

```

```

if (P8/P0<=1) {
    return 1e6;
}
if (design_mode==1 && (abs(1./NPR_1-1.12)/1.12)>0.02) {
    return 1e6;
}
if (design_mode==0 && (1./NPR_1)<1.01) {
    return 1e6;
}

// A8_star = W8/((P8)/sqrt(T8))/Q_function(1.0, gama_5 , R_air); // 0.32 mach
number is coming from hand calculation
// A8 = A8_try ; // fixed at off-design
if ((1./NPR_1)>=TPRC)
    NPR_1 = 1./TPRC ;

// double mncrt =(D94)*KAREKoK(2*HD*$E$90*(1-D94^(1/HD)))/((1-
$E$90*(1-D94^(1/HD)))) ;

W8_crt =(NPR_1)*sqrt(2.*HD_hot*CD_nozzle*(1.0-
pow(NPR_1,(1.0/HD_hot)))/((1.0-CD_nozzle*(1.0-pow(NPR_1,(1.0/HD_hot)))))) ;

W8_allowed = W8_crt*(A8)/sqrt(R_air) ; // allowable corrected flow according to
curve

W8_exhaust_calculated = W8*sqrt(T8)/(P8) ; // currently applied mass flow rate
// not choked nozzle
P8s = Pamb ;
T8s = T8*pow((P8s/P8),(gama_45-1.0)/gama_45) ;
double ro8 = P8s/(R_air*T8s) ;
//M8 = sqrt(2.0*(T8/T8s-1.0)/(gama_hot-1.0));
V8 = W8/(ro8*A8) ; // sqrt(2.0*CD_nozzle*(T8-T8s)*1000.0); // //
M8*sqrt(gama_hot*R_air*T8s)*CD_nozzle ; // exhaust velocity in m/s --
M8 = V8/sqrt(gama_45*R_air*T8s) ;
Thrust = W8*V8 - W2*V0; // Net thrust
// E R R O R # 3 :
// Turbine capacity difference in percent

Err_03 = (W8_exhaust_calculated-W8_allowed)/W8_exhaust_calculated ; //
should be less than 0.05 %

```

```

    if (Err_03<0) {
        Err_03 = 0; // if calculates is less than allowable no error occurs ; compressor
can easily pass the air flow
    }
    double pwr_penalty=0.0; //
    // -----
    // if (design_mode==1) {
        if ( PWR<PWRi[durum]) {
            pwr_penalty = (1.0-PWR/PWRi[durum])*((MTO+MCR+LOT)*2.0) ;
        } else {
            pwr_penalty=(PWR/PWRi[durum]-1.0)*((MTO+MCR)*0.5+LOT*0.5) ;
        }
        //pwr_penalty = pwr_penalty + abs(comp_work*watt_to_hp/3000.0) ;
    // }
    // -----
    double tc8 = (T5_real+T8s)*0.5 ; // thermocouple sens the mean temperature
    double TSFC = mf/((1+mf)*V8) ;
    //double FuelConsumption = Thrust*TSFC ;
    double Eff_overall = (0.5*(1+mf)*V8*V8)/(mf*LHV*Ecomb) ;
    double specific_thrust = Thrust/W2/9.81 ;
    //
    double hedef0 = max(max(Err_01, Err_02),Err_03) ;
    double hedef2 = Err_02 ; //abs(T4_burned-T4) ; //
    double hedef3 = abs(1./NPR_1-1.12)/1.12*0.01;
    double hedef4 = (1-Ecomp)*100 ; // abs(Wf/lb_to_kg-
Wfi[durum])/Wfi[durum]*1000.0+ abs(CPR-Pri[durum])/Pri[durum]*1000.0 ; //+(1-
Eturb)*100;
    if (design_mode == 1) {
        total_error[durum] = hedef0 + hedef3;
    }
    if (design_mode == 0) {
        total_error[durum] = hedef0 ;
    }

```

```

    }
    if (echo_on > 0) {
        if (design_mode == 0) {
            cout << "#####\n#####" << endl;

            cout << "### OFF - DESIGN MODE ###" << endl;
        } else {
            cout << "::::::::::::::::::::::::::::::::::::::::" << endl;

            cout << " ::: DESIGN MODE ::: " << endl;
        }

        cout << " Free Shaft Calisma noktasi : " << Nokta[durum] << endl;

        cout << "-----\n-----" << endl;

        cout << "Altitude = " << Zi[durum] << " , M0 = " << Mi[durum] << endl;

        cout << " CPR = " << CPR << " , TPR = " << TPR << " , LPTPR = " << LPTPR << " , NPR1 = " << 1/NPR_1 << endl;

        cout << " W2_crt = " << W2_try << " , tt_ = " << tt_ << " , speed = " << speed << " , Speed_crt = " << speed_crt << endl;

        cout << " W4 = " << W4 << " , W4_allowed = " << W4_allowed << " , Nozzle_allowed = " << W8_allowed << " , W8 = " << W8 << endl;

        cout << " W2_try [kg/s] = " << W2_try << " , W2 [kg/s] = " << W2 << " , Wfuel [g/s] = " << Wf*1000.0 << endl;

        cout << "-----\n-----" << endl;

        cout << " A2_ratio = " << A2_ratio << " , A4_ratio = " << A4_ratio << " , A45_ratio = " << A45_ratio << endl;

        cout << "-----\n-----" << endl;

        cout << " P0 = " << P0 << " , P2 = " << P2 << " , P3 = " << P3 << " , Pamb = " << Pamb << endl;

        cout << " P4 = " << P4 << " , P45 = " << P45 << " , P5 = " << P5 << " , P8s = " << P8s << endl;

        cout << "-----\n-----" << endl;
    }
}

```



```

        cout << " T0 = " << T0 << " , T2 = " << T2 << " T3_ideal = " << T3_ideal << " ,
T3_real = " << T3_real << " , Tamb = " << Tamb << endl;

        cout << " T4 = " << T4 << " , T45 = " << T45 << " T5_real = " << T5 << " ,
T5_ideal = " << T5_ideal << " , T8s = " << T8s << endl;

        cout << " -----
-----" << endl ;

        cout << " Comp work = " << comp_work*watt_to_hp << " | Turb work = " <<
turb_work*watt_to_hp << " [hp] " << endl ;

        cout << " -----
-----" << endl ;

        cout << " Required PWR= " << PWRi[durum] << " | calculated PWR = " <<
PWR_calculated << " , bonus thrust = " << Thrust << endl;

        cout << " -----
-----" << endl ;

        cout << " EGT_limit = " << T5i[durum] << " | TC8 = " << tc8 << endl;

        cout << " -----
-----" << endl ;

        cout << "Ecomp = " << Ecomp << " ,Ecomb = " << Ecomb << " , dP4 = " << dP4
<< " , mb = " << mb << " , TurbEff = " << Eturb << endl;

        cout << "Eff_overall = " << Eff_overall << " , TSFC = " << TSFC*1000.0 << " ,
Thrust= " << Thrust <<endl;

        cout << " V8 [m/s] = " << V8 << " , M8 = " << M8 << endl ;

        cout << " hedef0 = " << hedef0 << " , hedef2 = " << hedef2 << " , hedef3 = " <<
hedef3 << " , hedef4 = " << hedef4 << endl;

        cout << " -----
-----" << endl ;

        cout << " x[0] = " << x[0] << " , x[1] = " << x[1] << " , x[2] = " << x[2] << " , x[3]
= " << x[3] << " , x[4] = " << x[4] << endl ;

        cout << " _ _ _ _ _
_ _ _ _ _" << endl ;

        cout << " A4 = " << A4 << " A45 = " << A45 << " ,A8 = " << A8 << endl;

        cout << " rline = " << Rline_try << " fraction = " << fraction << endl;

        cout << " i0 , j0 = " << i0 << " , " << j0 << " , speed = " << Speed_comp[i0] <<
endl;

        cout << " T4_burned = " << T4_burned << " , T4 = " << T4 << endl;

        cout << " TPRC = " << TPRC << " , TPR = " << TPR << " , Tamb = " << Tamb
<< " , W8_crt = " << W8_crt << endl ;

```

```

        cout << " W4_allowed = " << W4_allowed << " , W45_allowed = " <<
W45_allowed << " , W8_allowed = " << W8_allowed << endl;
    }
    result = result + total_error[durum];
}
//} all cases are run and results are summed.
return result;
}
// free shaft turbine

```



EK-6. Turbofan motor fonksiyonu (iki şaftlı)

```
double fTurboFanFreeTurbine(double* x, int array_size){
    int expected_size = 6; // number of input parameters
    // operating: idle maximum cruise
    // TPE331 rpm % -> 100.0 97.0 95.0 -> rpms scaled according to LHTEC-800
compressor
    string Nokta[3]={ "TAKE-OFF" , "CRUISE" , "LOITER"} ;
    double Ni[3] = {90.0, 87.3, 85.5} ; // Physical Engine Speed [%] tasarim noktasi
%90, digerleri ona gore orantili
    double Zi[3] = { 0, 3810.0, 2133.6} ; // Altitude [m]
    double Mi[3] = { 0, 0.3478, 0.294} ; // Flight Mach number
    double PWRi[3]= { 953.0, 690.0, 603.0 } ; // Shaft power Requirement [hp] 953.1,
690.0, 603.0
    double T5i[3]= { 1023.0, 1023.0, 1023.0} ; // maximum EGT temperature[K]
    double Wfi[3] = { 0.1515, 0.1077, 0.097} ; // Wf = [lb/s]
    double Pri[3] = { 10.53, 11.6, 8.6} ; // Wf = [kg/s]
    double Wai[3] = { 3.496, 2.663, 2.461} ; // Wf = [kg/s]
    double T4maxi[3] = { 1391.3, 1302.3, 1248.4} ; // maximum combustor exit
temperature [K]
    double pr_scale = 1.0 ; //(10.2-1.0)/(11.12-1.0) ; //
    double mass_scale = 1.0 ; // compressor map mass flow scale - map design point
206, calculated engine 100
    double comp_eff_scale = 1.0 ; ///0.84168/0.780 ;
    double TPR_max = 10.5 ; //8.6 ;
    double turb_eff_scale = 1.0 ;
    double Thrust_c = 0.0 ; // cold thrust
    double Thrust_h = 0.0 ; // hot thrust
    double HPT_choke = 3.5 ; // HP Turbine choking pressure
    double LPT_choke = 2.5 ; // LP Turbine choking pressure
    double HD_hot = 1.33/(1.33-1.0) ;
    double HD_cold= 1.4/(1.4-1.0) ;
    //{
    double W25; ; // core engine air mass flow rate kg/s
```

```

double BPR ; // Fan by-pass ratio
double FPR ; // Fan pressure ratio
double CPR ; // compressor pressure ratio comes from map
double TPR ; // turbine pressure ratio comes from map
double NPR = 1.12 ; // Nozzle pressure ratio
double Wf ;
// Guess constants
double W25_try ; // trial compressor mass flow for matching
double T4_try ; // trial turbine pressure ratio for matching
//
// Matching constants
// HPT
double W4_crt ; // corrected flow per unit area
double W4_allowed ; // physically possible flow
double W4_turb_calculated ; // mass flow coming from the upstream
double Err_01 ; // difference between the stream value and component allowance
// LPT
double W45_crt ; // corrected flow
double W45_allowed ; // physical flow
double W45_turb_calculated ; // calculated corrected flow
double Err_02 ;
// CORE EXHAUST
double W8_crt ; // corrected flow
double W8_allowed ;
double W8_exhaust_calculated = 1.0;
double Err_03 ;
// FAN EXHAUST
double W18_crt ; // corrected flow
double W18_allowed ;
double W18_calculated ;
double Err_04 ;

```

```

//
double total_error[3] = {1.e9, 1.e9, 1.e9 }; // errors are summed separately for each
condition
double result = 0.0;
//}

// TPE331 baseline component information
double E_inlet = 1.0;    // 0.98 ; // inlet diffuser efficiency
double Efan = 0.86;      // Fan efficiency
double Ecomp = 0.856 ;   // compressor efficiency comes from map
double Eturb = 0.86 ;    // constant
double Ecomb = 0.975 ;   // combustion efficiency in percent
double Educt = (1.0-0.025) ; // 0.07 pressure loss at exhaust duct
double dP4 = (1.0-0.93) ; // PR_comb - combustor pressure drop in percent
double Emech = 0.98 ;    // mechanical efficiency
double CD_nozzle = 0.99 ;
double mb1 = 0.0271 ;    // bleed 1
double mb2 = 0.0374 ;    // bleed 2
double mb = mb1+mb2 ;    // blade cooling + NGV cooling - bleed_air/w2 ratio =
1 percent of main flow is bleed
double mf ;              //fuel/air ratio wf/W2
double accessory_work = 0.0 ; // accessory work and other power take offs
double A2 = 30.1*645.16/1.e6 ; // mm2 A2 compressor inlet M2=0.5
double A25_0 = 12.5*645.16/1.e6 ; // mm2 A25 stage 2 inlet M2=0.4
double A18_0 = 16000.0/1.e6 ; // mm2 A18 fan exit
double A3 = 6.4*645.16/1.e6 ; // mm2 A2 combustor inlet M3= 0.4
double A4_0 = 3519.6/1.e6 ; // mm2 A2 ngv inlet /28.5*645.16/1.e6/ M4 = 0.1
double A45_0 = 12254.9/1.e6 ; // mm2 A45 turbin exit M45= 0.4 -
71.4*645.16/1.e6
double A5 = 71.4*645.16/1.e6 ; // mm2 A5 turbin exit M5= 0.4
double A7 = 73.3*645.16/1.e6 ; // mm2 A7 duct cross section M7=0.4
double A8_0 = 37506.8/1.e6 ; // 55.6*645.16/1.e6 // 37133.0 ; // mm2 A8 Exhaust
area for M8= 0.4143 (turboprop-shaft)
// for design #2

```

```

// A4_0 = 2897.34 ;
// A45_0 = 10587.33 ;
// A8_0 = 52569.77 ;
int MTO = 1 ;
int MCR = 0 ;
int LOT = 0 ;

////////////////////////////////////

int design_mode = 1; // design_mode = 1, operating condition = 0
// design_mode = 0, operating condition = 1
if (design_mode==0) {
    x[2]= 0.5 ; // A4_ratio = 0.5+x[2]
    x[3]= 0.5 ; // A4_ratio = 0.5+x[2]
    x[4]= 0.5 ; // A45_ratio = 0.5+x[3]
}

////////////////////////////////////

// durum specifies engine operating point 0: Take Off, 1: Cruise, 2:Loiter
//

for (int durum=0; durum<1 ; durum++) {
//{ durumlari hesaplama loop u
    switch (durum) {
        case 0:
            MTO = 1; MCR = 0; LOT= 0;
            break;
        case 1:
            MTO = 0; MCR = 1; LOT = 0;
            break ;
        case 2:
            MTO = 0; MCR = 0; LOT = 1;
            break ;
    }
}

// Calculate P0, T0 according altitude standard atmosphere

```

```

double Pamb, Tamb ;
Altitude_T0P0(Zi[durum], &Tamb, &Pamb) ;
double M0 = Mi[durum] ;
double T5_limit = T5i[durum] ;
double speed = Ni[durum] ;

/**/**/**

double Rline_try = x[0] ; // try a rline on compressor speed line

double T4_try = 800.+x[1]*(T4maxi[0]-800.) ; // maximum and minimum
temperature range

/**/**/**

double A25 = A25_0*mass_scale ; //
double A18 = A18_0*1.0 ; //
double A4 = A4_0*(0.5+x[2]) ; // 0.649211; // Turbin mapi scale etmek ile
A4_ratio degistirmek ayni etkiyi olusturuyor / denedim gordum
double A45 = A45_0*(0.5+x[3]) ;
double A8 = A8_0*(0.5+x[4]) ; //
double PWR = PWRi[durum] ; //
FPR = 1.0+ x[5]*0.6 ;
// BPR = 0.1+ x[5]*5.9 ;

/**/**/**

double V0 = M0 * sqrt(gama_cold*R_air*Tamb) ;
double T0 = Tamb * (1.0 + (gama_cold-1.0)*0.5*M0*M0) ;
double P0 = Pamb * pow(T0/Tamb,HD_cold) ;
double P1 = P0*E_inlet; // inlet losses
double T1 = T0; // total temperature conserved
double P2 = P1*1.0; // pressure loss at inlet ducts
double T2 = T1; // temperature before the fan stage
double sqr_teta2 = sqrt(T2/288.15) ;
double T25_ideal = T2*pow((FPR),1.0/HD_cold) ;
double T25_real = T2+ (T25_ideal-T2)/Efan ;

double P25 = FPR * P2 ;

```

```

double T25 = T25_real ;
double sqr_teta25 = sqrt(T25/288.15) ; //
double speed_crt = speed/sqr_teta25 ;
//{ find compressor speed curve - lower curve no: i0, upper curve no: i1,
interpolation coefficient: tt_
    int i0 = 0;
    int i1 = 1;
    double tt_ = 0.0 ;
    for (int i=1;i<Ns_comp;i++) {
        if (speed_crt>=Speed_comp[i-1] && speed_crt<Speed_comp[i]) {
            i0 = i-1 ;
            i1 = i ;
            tt_ = (speed_crt-Speed_comp[i0])/(Speed_comp[i1]-Speed_comp[i0]); // tt_
is the ratio between the speed curves
        }
    }
    // find turbine speed curve - lower curve no: it0, upper curve no: it1, interpolation
coefficient: ttt
    // int it0 = 0;
    // int it1 = 1;
    // double ttt = 0.0 ;
    //
    // for (int i=0;i<Ns_turb;i++) {
    //     if (speed>=Speed_turb[i] && speed<Speed_turb[i+1]) {
    //         it0 = i ;
    //         it1 = i+1 ;
    //         ttt = (speed-Speed_turb[it0])/(Speed_turb[it1]-Speed_turb[it0]); // tt_ is the
ratio between the speed curves
    //     }
    // }
    //}

//{ compressor operating point trial
int j0 = (int)floor(Rline_try/0.1) ; // data number on the speed line

```



```

int j1 = j0+1 ;
if (j1==Data_comp) j1 = j0 ;

double Mdot0 = Mdot_comp_map[i0][j0]+(Mdot_comp_map[i1][j0]-
Mdot_comp_map[i0][j0])*tt_;

double Mdot1 = Mdot_comp_map[i0][j1]+(Mdot_comp_map[i1][j1]-
Mdot_comp_map[i0][j1])*tt_;

double PR0 = PR_comp_map[i0][j0]+(PR_comp_map[i1][j0]-
PR_comp_map[i0][j0])*tt_;

double EFF0 = Eff_comp_map[i0][j0]+(Eff_comp_map[i1][j0]-
Eff_comp_map[i0][j0])*tt_;

double PR1 = PR_comp_map[i0][j1]+(PR_comp_map[i1][j1]-
PR_comp_map[i0][j1])*tt_;

double EFF1 = Eff_comp_map[i0][j1]+(Eff_comp_map[i1][j1]-
Eff_comp_map[i0][j1])*tt_;

double fraction = Rline_try-j0*0.1 ;

W25_try = Mdot0 + fraction * (Mdot1-Mdot0); // W2 trial - corrected mass flow
rate W2_crt

CPR = PR0 + fraction * (PR1-PR0) ; //

Ecomp = EFF0 + fraction * (EFF1-EFF0) ; //

Ecomp = Ecomp *comp_eff_scale ;

// cout << " Rline_try = " << Rline_try << " , Mdot = " << W2_try << " , CPR = "
<< CPR << endl ;

//} compressor point defined

W25_try = W25_try * mass_scale ;

W25 = W25_try * (P25/101325.0)/sqr_teta25 ; // corrected to -> physical mass
flow

CPR = pr_scale*(CPR-1.0)+1.0 ; // scale pressure in addition to the map if
necessary

// for given FPR and PWR find resulting BPR

double fan_work = PWR*hp_to_watt ;

BPR = fan_work/((T25-T2)*cp_cold*W25)-1.0 ;

// if outside the by-pass constraint return big number

//cout << "." << fan_work << "." ;

if (BPR<0.5||BPR>6.0)

return 1e6 ;

```

```

double W18 = W25*BPR; // cold bypass air
double T18 = T25;
double P18 = P25;
double T3_ideal= T25*pow((CPR),1.0/HD_cold) ;
double T3_real = (T3_ideal-T25)/Ecomp + T25;
double P3 = CPR * P25 ;
double T3 = T3_real ;
double W3 = W25*(1-mb) ;
double comp_work = W25*(T3_real-T25)*cp_cold ;    // necessary work for
compressor
double turb_work = (comp_work + accessory_work*hp_to_watt)/Emech ;
double P4 = P3 * (1.0-dP4) ; // dP4=7/100 constant
double T4 = T4_try ; // try T4 between Tmin - Tmax
Wf= W3*(T4*cp_hot - T3_real*cp_cold)/(LHV*Ecomb-T4*cp_hot);
mf = Wf/W25 ;
if (Wf<=0.001) {
    return 1000000.0 ;
}
double W4 = W25*(1+mf-mb) ; // physical mass flow - not corrected
double T4_burned=(W3*T3_real*cp_cold + Wf*LHV*Ecomb)/(W4*cp_hot) ; //
if wf is burned temperature would be T4_burned
// depending on the T4_burned cp changes and Wfuel should be updated
accordingly
Wf= Wf * 0.5*(T4_burned+T4)/(T4);
mf = Wf/W25 ;
W4 = W25*(1+mf-mb) ;
double T45_real = T4- turb_work/(W4*cp_hot) ;
double T45_ideal = T4-(T4-T45_real)/Eturb ;
if (T45_ideal<T25)
    return 1e6 ;
// turbine pressure ratio depending on the temperature ratios
TPR = pow((T4/T45_ideal),HD_hot) ;

```

```

// pressure ratio according to work balance is TPR, if is it different from TPR_try
generate error

if (TPR>TPR_max) {
    TPR = TPR_max ;
}

// calculation according to Turbine Characteristic Curve

// normal curve chokes around 1.89 however real turbines choke at different
pressure ratio

// therefore turbine curve is scaled on the PR axis. corresponding mass flow rates
kept the same

double TPRC = pow((1.0+(gama_hot-1.0)/2.0),HD_hot) ;

double TPR_mod = (TPRC-1.)/(HPT_choke-1.) * (TPR-1.0) + 1.0 ; // scale turbine
to choke at PR=4.5

if (TPR_mod>=TPRC)
    TPR_mod = TPRC ;

// mncrt=(NPR_1)*sqrt(2*HD*CD_nozzle*(1.0-pow(NPR_1,(1.0/HD))))/((1.0-
CD_nozzle*(1.0-pow(NPR_1,(1.0/HD)))));

// W8_allowed = mncrt*(A8)/sqrt(R_air*T8)*P8 ; // allowable nozzle mass flow
rate

// W8_exhaust_curve = W8_allowed *sqrt(T8)/(P8*0.001) ;

W4_crt =(1./TPR_mod)*sqrt(2*HD_hot*(1.0-pow(1./TPR_mod,(1.0/HD_hot
))))/((1.0-(1.0-pow(1./TPR_mod,(1.0/HD_hot)))));

W4_allowed = W4_crt*(A4)/sqrt(R_air) ; // allowed corrected flow

W4_turb_calculated = W4*sqrt(T4)/P4; // calculated corrected flow

// E R R O R # 1 :

// Turbine capacity difference in percent

Err_01 = abs(W4_turb_calculated-W4_allowed)/W4_turb_calculated*100.0 ; //
should be less than 0.05 %

double P45 = (P4/TPR)*Educt ;

// double FAR45 = Wf/W25 ;

// double cp_45 = cp_hot ; //Cp_gas(T5_real,FAR5) ;

// double gama_45 = 1.33481 ; // (cp_5)/(cp_5-R_air) ;

//HD = gama_5/(gama_5-1.0) ;

double W45 = W25*(1+mf) ;

```

```

double T45 = (T45_real*cp_hot*W4+T3*cp_cold*W25*mb)/(W45*cp_hot); //
cooling air returns to flowpath and decrease gas temperature

// free-shaft turboshaft can expand from P45 to P0*NPR;

// (PWR + accessory_work)*hp_to_watt

double T5_real = T45-PWR*hp_to_watt/(W45*cp_hot); // power required by free
turbine

double T5_ideal = T45-(T45-T5_real)/Eturb;

double P5 = P45*pow((T5_ideal/T45),HD_hot);

double LPTPR = P45/P5;

double T5 = T5_real;

double W5 = W45;

// pressure ratio according to work balance is TPR, if is it different from TPR_try
generate error
if (LPTPR>TPR_max) {
    LPTPR = TPR_max;
}

// calculation according to Turbine Characteristic Curve

double LPTPR_mod = (TPRC-1.)/(LPT_choke-1.)* (LPTPR-1.0) + 1.0; // scale
turbine to choke at PR=4.5

if (LPTPR_mod>=TPRC)
    LPTPR_mod = TPRC;

W45_crt = (1./LPTPR_mod)*sqrt(2*HD_hot*(1.0-
pow(1./LPTPR_mod,(1.0/HD_hot)))/((1.0-(1.0-pow(1./LPTPR_mod,(1.0/HD_hot)))));

W45_allowed = W45_crt*(A45)/sqrt(R_air); // allowed corrected flow

W45_turb_calculated = W45*sqrt(T45)/P45; // calculated

// E R R O R # 2 :

// Turbine capacity difference in percent

Err_02 = abs(W45_turb_calculated-W45_allowed)/W45_turb_calculated*100.0;
// should be less than 0.05 %

// HOT Nozzle

// Design mode = 1;

double gama_8 = 1.33;

double T8 = T5;

```

```

double P8 = P5*Educt ; // simple convergent nozzle - 0.025 loss at jet pipe - fully
expands into atmosphere

double W8 = W5 ; // assuming bleed returned in to the main flow

double P8s ;

double NPR = P8/Pamb ; // initial approach

double M8 = sqrt(2.0*(pow((NPR),1.0/HD_hot)-1.0)/(gama_8-1.0)) ;

double V8 ;

double T8s ;

double ro8 ;

if (M8>=1.0){
    M8 = 1.0 ;
    T8s = T8/(1.0+0.5*(gama_8-1.0)) ;
    NPR = pow((1.0+(gama_8-1.0)/2.0),HD_hot) ;
    P8s = P8/NPR;
    V8 = sqrt(gama_8*R_air*T8s);
    ro8 = P8s/(R_air*T8s) ;
    if (design_mode==1) {
        double A8_star = W8/(V8*ro8) ;
        A8 = A8_star ;
        W8_exhaust_calculated = W8*sqrt(T8)/P8 ;
        W8_allowed = W8_exhaust_calculated ;
        // A18 is modified
        Err_03 = 0.0 ;
    }
} else {
    T8s = T8/(1.0+0.5*(gama_8-1.0)*M8*M8) ;
    // NPR is unchanged
    P8s = Pamb ;
    V8 = M8*sqrt(gama_8*R_air*T8s);
    ro8 = P8s/(R_air*T8s) ;
    if (design_mode==1) {
        A8 = W8/(ro8*V8) ;
    }
}

```

```

W8_exhaust_calculated = W8*sqrt(T8)/P8 ;
W8_allowed = W8_exhaust_calculated ;
// A18 is modified
Err_03 = 0.0 ;
}
}
// Design mode = 0;
if (design_mode == 0) {
    // double mncrt =(D94)*KAREKoK(2*HD*$E$90*(1-D94^(1/HD)))/((1-
    $E$90*(1-D94^(1/HD)))) ;
    double NPR_1 = 1.0/NPR ;
    W8_crt =(NPR_1)*sqrt(2*HD_hot*CD_nozzle*(1.0-
    pow(NPR_1,(1.0/HD_hot)))/((1.0-CD_nozzle*(1.0-pow(NPR_1,(1.0/HD_hot)))))) ;
    W8_allowed = W8_crt*(A8)/sqrt(R_air) ; // allowable corrected nozzle mass
    flow rate
    W8_exhaust_calculated = W8*sqrt(T8)/P8 ; //
    // E R R O R # 3 :
    // Turbine capacity difference in percent
    Err_03 = (W8_exhaust_calculated-W8_allowed)/W8_exhaust_calculated ; //
    should be less than 0.05 %
    if (Err_03<0) {
        Err_03 = 0; // if calculates is less than allowable no error occurs ; compressor
        can easily pass the air flow
    }
}
Thrust_h = W8*V8 - W25*V0; // hot thrust
// COLD Nozzle
// Design mode = 1;
double P18s = Pamb ;
double FNPR = P25/P18s ;
double T18s = T18/pow((FNPR),(1./HD_cold)) ;
double M18 = sqrt(2.0*(pow((P25/P18s),1.0/HD_cold)-1.0)/(gama_cold-1.0)) ;
double ro18 = P18s/(R_air*T18s) ;

```

```

double V18 = M18*sqrt(gama_cold*R_air*T18s);
    if (design_mode==1) {
        A18 = W18/(ro18*V18);
        W18_calculated = W18*sqrt(T18)/P18;
        W18_allowed = W18_calculated;
        // A18 is modified
        Err_04 = 0.0;
    }
if (M18>=1.0) {
    M18 = 1.0;
    T18s = T25/(1.0+0.5*(gama_cold-1.0));
    P18s = P25*pow(T18s/T25,HD_cold);
    FNPR = P25/P18s;
    V18 = sqrt(gama_cold*R_air*T18s);
    ro18 = P18s/(R_air*T18s);
    if (design_mode==1) {
        double A18_star = W18/(V18*ro18);
        A18 = A18_star;
        // A18 is modified
        // Err_04 = 0.0;
    }
}
// Design mode = 0;
if (design_mode == 0) {
    double FNPR_1=1.0/FNPR;
    W18_crt = (FNPR_1)*sqrt(2*HD_cold*CD_nozzle*(1.0-
pow(FNPR_1,(1.0/HD_cold))))/((1.0-CD_nozzle*(1.0-pow(FNPR_1,(1.0/HD_cold)))));
    W18_allowed = W18_crt*(A18)/sqrt(R_air); // allowable nozzle mass flow rate
- physical
    W18_calculated = W18*sqrt(T18)/P18;
    // E R R O R # 4 :
    // FAN exhaust capacity difference in percent

```

```

        if (W18_allowed >= W18) {
            Err_04 = 0.0 ;
        } else {
            Err_04 = (W18_calculated - W18_allowed) / W18_calculated * 100.0 ; // should
be less than 0.05 %
        }
    }

    Thrust_c = W18 * (V18 - V0); // cold thrust
    double Thrust = (Thrust_c + Thrust_h) / (0.454 * 9.81) ;
    double tc8 = (T5_real + T8s) * 0.5 ; // thermocouple sens the mean temperature
    double TSFC = Wf / ((Thrust_c + Thrust_h) / 9.81) ;
    // double FuelConsumption = Thrust * TSFC ;
    double Eff_overall = (0.5 * ((1 + mf) * V8 * V8 + BPR * V18 * V18)) / (mf * LHV * Ecomb) ;
    double specific_thrust = (Thrust_c + Thrust_h) / (W25 * (1 + BPR + mf)) / 9.81 ;
    //
    double hedef0 = max(max(Err_01, Err_02), max(Err_03, Err_04)) ;
    double hedef2 = Err_02 ; // abs(T4_burned - T4) ; //
    double hedef3 = W25 * Wf * (1 + BPR) / Thrust ; // 1000. / Thrust ; // specific_thrust ;
    double hedef4 = (1 - Ecomp) * 100 ; // abs(Wf/lb_to_kg-
Wf[durum]) / Wf[durum] * 1000.0 + abs(CPR - Pri[durum]) / Pri[durum] * 1000.0 ; // +(1-
E_turb) * 100;
    if (design_mode == 1) {
        total_error[durum] = hedef0 + hedef3;
    }
    if (design_mode == 0) {
        total_error[durum] = hedef0 ;
    }
    if (echo_on > 0) {
        if (design_mode == 0) {
            cout << "
#####
#####" << endl ;
            cout << " ### OFF - DESIGN MODE ### " << endl;

```



```

        cout << " -----
-----" << endl ;

        cout << " Required PWR= " << PWR << " , Thrust_hot [N]= " << Thrust_h << " ,
Thrust_cold [N]= " << Thrust_c << endl;

        cout << " -----
-----" << endl ;

        cout << " EGT_limit = " << T5i[durum] << " | TC8 = " << tc8 << endl;

        cout << " -----
-----" << endl ;

        cout << "Ecomp = " << Ecomp << " ,Ecomb = " << Ecomb << " , dP4 = " << dP4
<< " , mb = " << mb << " , TurbEff = " << Eturb << endl;

        cout << "Eff_overall = " << Eff_overall << " , TSFC = " << TSFC*1000.0 << " ,
Thrust [lbf] = " << Thrust << endl;

        cout << " V8 [m/s]= " << V8 << " , M8 = " << M8 << " , V18 [m/s]= " << V18 <<
" , M18 = " << M18 << " , V8/V18 = " << V8/V18 << endl ;

        cout << " hedef0 = " << hedef0 << " , hedef2 = " << hedef2 << " , hedef3 = " <<
hedef3 << " , hedef4 = " << hedef4 << endl;

        cout << " -----
-----" << endl ;

        cout << " x[0]= " << x[0] << " , x[1]= " << x[1] << " , x[2]= " << x[2] << " , x[3]
= " << x[3] << " , x[4]= " << x[4] << " , x[5]= " << x[5] << endl ;

        cout << " _ _ _ _ _
_ _ _ _ _" << endl ;

        cout << " rline = " << Rline_try << " fraction = " << fraction << endl;

        cout << " i0 , j0 = " << i0 << " , " << j0 << " , speed = " << Speed_comp[i0] <<
endl;

        cout << " T4_burned = " << T4_burned << " , T4 = " << T4 << " , Specific_Thrust
= " << specific_thrust << endl;

        cout << " W8_exhaust_calculated = " << W8_exhaust_calculated << "
,W8_allowed = " << W8_allowed << " , W18_calculated = " << W18_calculated << "
,W18_allowed = " << W18_allowed << endl ;

        cout << " W4_turb_calculated = " << W4_turb_calculated << " , W4_allowed = "
<< W4_allowed << " , W45_turb_calculated = " << W45_turb_calculated << " ,
W45_allowed = " << W45_allowed << endl;

        cout << " FPR = " << FPR << " , BPR = " << BPR << endl;

        RESULT << num_of_iter << " , " << FPR << " , " << BPR << " , " << TSFC << " , "
<< Thrust << " , " << T4 << endl;

    }

```

```

        result = result + total_error[durum];
    }
    return result;
}
//} Free Shaft to Turbofan – End

```

EK-7. Ana program satırları

```

#include <SFML/Graphics.hpp>
#include <stdio.h>
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
#include <time.h>
#include <random>
#include <math.h>
#include <list>
#include <chrono>
#include <fstream>
#include ".\grafik2.h"
using namespace std;

std::random_device rd; //Will be used to obtain a seed for the random number engine
std::mt19937 mt(rd()); //Standard mersenne_twister_engine seeded with rd()
std::uniform_real_distribution<> dis(0.0, 1.0);
std::ofstream RESULT;
int num_of_iter = 0;
double fTurboShaftFixed(double* x, int boyut); // TurboShaft test function
double fTurboShaftFree(double* x, int boyut); // Free shaft turbine
double fTurboFanFreeTurbine(double* x, int boyut); // Two shaft turbofan
double fTurboFanMapped(double* x, int boyut); // Turbofan with map

```

```

auto (*fTest) = fTurboFanMapped ; // Test function pointer
double Sutherland_dynamic(double Sicaklik);
double Cp_gas (double gas_temp, double FAR);
double Q_function(double MachNumber, double Gama, double R_);
double Altitude_T0P0(double altitude, double* T0, double* P0);
const double olcek = 1.3 ; // pheromone scale
const double bozulma = 0.3 ; // decaying ratio of pheromone
const double tolerans = 0.00001 ; // convergence tolerance for ant iteration
const double pi = 2*acos(0);
const double epsilon = 0.0000000000001 ; // 1e-12 small number
const double cp_cold = 1004.5 ; // kj/kg.K
const double cp_hot = 1150.0 ; //1281.52 ; // kj/kg.K    REf- Walsh and Fletcher
sayfa 312
const double gama_cold = 1.4 ; // gama sabiti hava icin
const double gama_hot = 1.33 ; // gama sabiti yanmis gazlar icin
const double R_air = 287.05 ; // hava icin gaz sabiti
const double LHV = 42800000.0 ; // yakitin isil enerjisi Lower Heating Value
kJ/kg
double E_turb_initial = 0.855 ;
double Q8_max = Q_function(1.0, gama_hot, R_air) ; // max Q value at exhaust
nozzle
double CD_nozzle = 0.99 ;
//double PR_Critical = pow(1-((gama_hot-1)/(gama_hot+1)/CD_nozzle),-
(gama_hot/(gama_hot-1))) ;
//{ conversion constants
double lb_to_kg = 0.45359237 ;
double lbf_to_kgf = 0.45359237 ; // libre force to kg force
double lbf_to_N = 0.45359237*9.81 ; // libre force to kg force
double R_to_K = 5.0/9.0 ; // Rankine to Kelvin
double psi_to_kPa = 7.03543 ; // psi to Pa
double ft_to_meter = 0.3048 ; // feet to meter
double watt_to_hp = 0.00134102209 ; // Watt to horsepower
double hp_to_watt = 745.699871357078 ; // horsepower to Watt

```

```

//} conversion constants

// MAP data is global - necessary speed curves are used by turbojet and fTurboShaft
functions

// 100% curve points for compressor and turbine are selected for matching

// PR should be increasing

// MAP data is generated in excel , ref [] :

const int Ns_comp = 14 ; // number of HPC speed lines

const int Ns_fan = 8 ; // number of HPC speed lines

const int Ns_turb = 9 ; // number of HPT speed lines

int Data_comp = 20 ; // data number in a speed line

int Data_fan = 21 ; // data number in a speed line

int Data_turb = 11 ; // data number in a speed line

const double Rline[20] = {0, 0.052631579, 0.105263158, 0.157894737,
0.210526316, 0.263157895, 0.315789474, 0.368421053, 0.421052632, 0.473684211,
0.526315789, 0.578947368, 0.631578947, 0.684210526, 0.736842105, 0.789473684,
0.842105263, 0.894736842, 0.947368421, 1} ;

const double Fline[21] =
{0,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95,1.0}
;

const double Speed_comp[14] = {60.0, 65.0, 70.0, 75.0, 80.0, 85.0, 90.0, 92.5, 95.0,
97.5, 100.0, 103.0, 107.0, 110.0} ; // % percent speed %100= 41000 TPE-331

const double Speed_fan[8] = {70.0, 75.0, 80.0, 85.0, 90.0, 95.0, 100.0, 105.0} ;

const double Tline[20] = {1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.2, 2.3,
2.4, 2.5, 2.6, 2.7, 2.8, 2.9} ;

const double Speed_turb[9] = {30,40,50,60,70,80,90,100,110} ; // % percent

const double Mdot_fan_map[8][21] = {
};

const double PR_fan_map[8][21] = {
};

const double Eff_fan_map[8][21] = {#};

const double Mdot_comp_map[14][20] = {#};

const double PR_comp_map[14][20] = {#};

const double Eff_comp_map[14][20] = {#};

const double Mdot_turb_map[9][11] = {#};

```

```

const double PR_turb_map[9][11] = {#};
double Genislik = 1024 ; // width of the graphical window
double Yukseklik= 1024 ; // height of the window
int MaxFrame = 10 ;    // keep refresh rate low in order to spend less time on
graphical update
int boyut = 6; // girdilerin parametre boyutu
int echo_on ; // hide outputs from the < function
int max_path = 20 ; // number of possible paths - or selectable targets
double xf_ratio = 1.0 ; // distance versus function value difference
double toplam_feromon;
double yeni_total_feromon;
double low_feromon = 1.0 ;
int swarm_shuffle = 50 ; // change target after 200 iterations if swarm stucked in to a
location
int feromon_update = 100 ;
auto Start_time = std::chrono::system_clock::now();
auto End_time = std::chrono::system_clock::now();
double Sutherland_dynamic(double Sicaklik){
    double dynamic_visc = 1.461*1e-6*pow(Sicaklik,3./2.)/(Sicaklik+110.3);
    return dynamic_visc;
};
double R_gas (double FAR){
    // walsh and fletcher Page 115 "Properties and Charts for Dry Air, Combustion
    Products and other Working Fluids"
    // F3.22 Gas constant for products of combustion in dry air
    double R_tmp = 287.05 - 0.00990*FAR + 1e-7*FAR*FAR ;
    return R_tmp ;
}
double Cp_gas (double gas_temp, double FAR){
    double A0 = 0.992313 ; double A1 = 0.236688 ; double A2 = -1.852148 ; double A3
= 6.083152 ;
    double A4 = -8.893933 ; double A5 = 7.097112 ; double A6 = -3.234725 ; double A7
= 0.794571 ;

```

```

double A8 = -0.081873 ; double A9 = 0.422178 ; double A10 = 0.001053 ;

double B0= -0.718874 ; double B1=8.747481 ; double B2= -15.863157 ; double
B3=17.254096 ; double B4= -10.233795 ;

double B5= 3.081778 ; double B6= -0.361112 ; double B7= -0.003919 ; double
B8=0.0555930 ; double B9= -0.0016079 ;

double TZ = gas_temp*0.001 ;

double TZ_2 = TZ*TZ ;

double TZ_3 = TZ_2*TZ ;

double TZ_4 = TZ_3*TZ ;

double TZ_5 = TZ_4*TZ ;

double TZ_6 = TZ_5*TZ ;

double TZ_7 = TZ_6*TZ ;

double TZ_8 = TZ_7*TZ ;

double CP_A = A0 + A1 * TZ + A2 * TZ_2 + A3 * TZ_3 + A4 * TZ_4 + A5 * TZ_5
+ A6 * TZ_6 + A7 * TZ_7 + A8 * TZ_8 ;

double CP_B = FAR/(1.0 + FAR) * (B0 + B1 * TZ + B2 * TZ_2 + B3 * TZ_3 + B4 *
TZ_4 + B5 * TZ_5 + B6 * TZ_6 + B7 * TZ_7) ;

double CP = 1000.0*(CP_A+CP_B) ;

return CP;

}

double Altitude_T0P0(double altitude, double* T0, double* P0){
    // altitude comes as meters
    altitude = altitude * 0.001 ; // convert to kilometers

    //          0      1      2      3      4      5      6      7      8
    double zi[9] = { -5.00 , 0.00 , 11.00 , 20.00 , 32.00 , 47.00, 51.00, 71.00,
84.50 } ; // kilometers

    double Li[9] = { -6.50 , -6.50 , 0.00 , 1.00, 2.80 , 0.00, -2.80, -2.00, 0.00 }
; // slope coef

    double Ti[9] = {320.676 , 288.15 , 216.65 , 216.65, 228.65, 270.65, 270.65,
214.65, 189.867 } ; // Kelvin

    double Pi[9] = {177760.0, 101325.0, 22632.0, 5474.8, 868.01, 110.9, 66.938,
3.9564, 0.4866} ; // Pascals

    double W0 = 28.9644 ; // kg/kmol

    double Rstar = 8.31432 ; // J/kmol-K

```

```

double g0 = 9.80665 ; // m/s2

// -500 m to 0 m
if (altitude>zi[0] && altitude<zi[1]) {
    *T0 = Ti[0] + Li[0]*(altitude - zi[0]) ;
    *P0 = Pi[0] * pow((Ti[0] / *T0),(g0*W0/(Rstar*Li[0])))) ;
}

// 0 m to 11000 m
if (altitude>=zi[1] && altitude<=zi[2]) {
    *T0 = Ti[1] + Li[1]*(altitude - zi[1]) ;
    *P0 = Pi[1] * pow((Ti[1] / *T0),(g0*W0/(Rstar*Li[1])))) ;
}

// 11000 m to 20000 m
if (altitude>zi[2] && altitude<=zi[3]) {
    *T0 = Ti[2] ;
    *P0 = Pi[2] * pow(10.0, (-g0*W0*(altitude-zi[2])/(Rstar*Ti[2])));
}

// 20000 m to 32000 m
if (altitude>zi[3] && altitude<=zi[4]) {
    *T0 = Ti[3] + Li[3]*(altitude - zi[3]) ;
    *P0 = Pi[3] * pow((Ti[3] / *T0),(g0*W0/(Rstar*Li[3])))) ;
}

// 32000 m to 47000 m
if (altitude>zi[4] && altitude<=zi[5]) {
    *T0 = Ti[4] + Li[4]*(altitude - zi[4]) ;
    *P0 = Pi[4] * pow((Ti[4] / *T0),(g0*W0/(Rstar*Li[4])))) ;
}

// 47000 m to 51000 m
if (altitude>zi[5] && altitude<=zi[6]) {
    *T0 = Ti[5] ;
    *P0 = Pi[5] * pow(10.0, (-g0*W0*(altitude-zi[5])/(Rstar*Ti[5])));
}

```



```

// 51000 m to 71000 m
if (altitude>zi[6] && altitude<=zi[7]) {
    *T0 = Ti[6] + Li[6]*(altitude - zi[6]) ;
    *P0 = Pi[6] * pow((Ti[6] / *T0),(g0*W0/(Rstar*Li[6]))) ;
}
// 71000 m to 85000 m
if (altitude>zi[7] && altitude<=zi[8]) {
    *T0 = Ti[7] + Li[7]*(altitude - zi[7]) ;
    *P0 = Pi[7] * pow((Ti[7] / *T0),(g0*W0/(Rstar*Li[7]))) ;
}
return 1.0;
};

double Q_function(double MachNumber, double Gama, double R_) {
    double tr_ = 1.0+0.5*(Gama-1.0)*pow(MachNumber,2.0) ; // sicaklik orani
T8t/T8s nozzle cikisinda
    double pr_ = pow(tr_,(Gama/(Gama-1.0))) ; // basinc orani P8t/P8s nozzle
cikisinda
    return pow(Gama/R_,0.5)*MachNumber*pow(tr_, -(Gama+1.0)/(2.0*(Gama-
1.0)));
}

//*****

int main()
{
//  std::mt19937 mt(rd()); //Standard mersenne_twister_engine seeded with rd()
//  std::uniform_real_distribution<> dis(0.0, 1.0);
    auto nSeed = time(0);
    mt.seed(nSeed);
    echo_on = 0;
    cout.precision(16);
    RESULT.open ("00-result-01.txt");
    RESULT.precision(16);

```

```

// RESULT << "num_of_iter" << ", " << "CPR" << ", " << "W2_crt" << ", " <<
"TPR" << ", " << "NPR" << ", " << "T4" << ", " << "P4" << ", " << "W4_crt" << ", " <<
"A4" << ", " << "W4_allowed" << endl;

RESULT << num_of_iter << ", " << "FPR" << ", " << "BPR" << ", " << "TSFC"
<< ", " << "Thrust" << ", " << "T4" << endl;

// define variable swarm and define the size of swarm
//

swarm suru1;

suru1.setSize(1000);

// Create the main graphical window
sf::RenderWindow app(sf::VideoMode(Genislik, Yukseklik), "SFML window");
app.setFramerateLimit(MaxFrame);
sf::RectangleShape dortgen;
dortgen.setPosition(-500,-500);
dortgen.setSize(sf::Vector2f(1000,1000));
dortgen.setFillColor(sf::Color::Black);
dortgen.setOutlineThickness(30);
dortgen.setOutlineColor(sf::Color::Blue);
sf::View view(sf::FloatRect(0,0,1024,1024));
view.setCenter(0,0);

// Sonar systems tutorials are used for sfml graphics
// uncomment this section for plotting compressor map
// open a new graphical window for displaying the function
// optimization begins after closing this windows
// Grafik2 gr1;
// gr1.BASLIK = " Compressor Map" ;
// gr1.x_axis_title = "Corrected Mass Flowrate [kg/s]";
// //gr1.y_axis_title = "x2 axis";
// gr1.contour_max = 1.0 ; //max value
// gr1.contour_min = 0.0 ; //min value
// gr1.x_0 = 0.0; // x axis minimum value on scale
// gr1.x_1 = 7.0; // x axis maximum value on scale

```

```

// gr1.y_0 = 0.0; // y axis minimum value on scale
// gr1.y_1 = 24.0; // y axis maximum value on scale
//// int n = 20 ; // data
// int m = 14 ; // speed line
// point3d dp[n*m]; // dummy point
// cell2d du; // dummy ugen
//// std::vector<cell2d> mesh;
// std::vector<point3d> speed_line;
// std::vector<point3d> max_eff_pts;
//// // nodes preparation -> get data from compressor map
////
// double maximum_value = -9999.0;
// double minimum_value = 9999.0;
// double x[3];
// for (int j=1; j<=n; j++)
// {
//     for (int i=1; i<=m; i++)
//     {
//         dp[i+(j-1)*m].x = Mdot_comp_map[i-1][j-1];
//         dp[i+(j-1)*m].y = PR_comp_map[i-1][j-1];
//         dp[i+(j-1)*m].z = Eff_comp_map[i-1][j-1];
//         x[0]=dp[i+(j-1)*m].x;
//         x[1]=dp[i+(j-1)*m].y;
//         x[2]=dp[i+(j-1)*m].z;
//
//         if(dp[i+(j-1)*m].z>maximum_value)
//             maximum_value = dp[i+(j-1)*m].z;
//         if(dp[i+(j-1)*m].z<minimum_value)
//             minimum_value =dp[i+(j-1)*m].z;
//     }
// }

```

```

// gr1.contour_max = maximum_value ; //max value
// gr1.contour_min = minimum_value ; //min value
// // mesh vector preparation
// for (int j=1; j<n; j++)
// {
//     for (int i=1; i<m; i++)
//     {
//         int n1 = i+(j-1)*m;
//         int n2 = i+j*m;
//         int n3 = (i+1)+j*m;
//         int n4 = (i+1)+(j-1)*m;
//         if (n1<=m*n && n2<=m*n && n3<=m*n && n4<=m*n )
//         {
//             du.p[0]= dp[n1]; du.p[1]= dp[n2] ;du.p[2]= dp[n4]; du.alan = 0.0f;
//             mesh.push_back(du);
//             du.p[0]= dp[n2]; du.p[1]= dp[n3] ;du.p[2]= dp[n4]; du.alan = 0.0f;
//             mesh.push_back(du);
//         }
//     }
// }
// double max_eff ;
// int gcc_index ;
// gr1.marker = '.' ;
// gr1.contour(mesh);
// //gr1.axis();
//
// for (int i=1; i<=m; i++)
// {
//     speed_line.clear();
//     max_eff = 0;
//     for (int j=1; j<=n; j++)

```

```

//      {
//          speed_line.push_back(dp[i+(j-1)*m]);
//          if (dp[i+(j-1)*m].z>max_eff)
//          {
//              max_eff = dp[i+(j-1)*m].z;
//              gcc_index = i+(j-1)*m ;
//          }
//      }
//      max_eff_pts.push_back(dp[gcc_index]);
//      gr1.polyline(speed_line);
//  }
//  gr1.marker = 'o' ;
//  gr1.polyline(max_eff_pts);
//  gr1.yuvarlak(3.49,10.55);
//  gr1.axis();
//  gr1.display();
// function display part is finished here
// uncomment above section for plotting compressor map
//
// Start the loop
while (app.isOpen())
{
    // Process events
    sf::Event event;
    while (app.pollEvent(event))
    {
        // Close window : exit
        switch (event.type){
            case sf::Event::Closed:
                app.close();
                break;;
        }
    }
}

```

```

case sf::Event::MouseButtonReleased:
    //suru1.target.x[0] = event.mouseButton.x-512;
    //suru1.target.x[1] = event.mouseButton.y-512;
    break;
case sf::Event::KeyPressed:
    switch (event.key.code)
    {
        case sf::Keyboard::W:
            view.zoom(2);
            break;
        case sf::Keyboard::S:
            view.zoom(0.5);
            break;
        case sf::Keyboard::Right:
            view.move(40,0);
            break;
        case sf::Keyboard::Left:
            view.move(-40,0);
            break;
        case sf::Keyboard::Up:
            view.move(0,-40);
            break;
        case sf::Keyboard::Down:
            view.move(0,40);
            break;
        case sf::Keyboard::Escape:
            app.close();
            break;
    }
}
}

```

```
// Clear screen
app.clear();
app.setView(view);
app.draw(dortgen);
suru1.draw(app);
app.display();
suru1.update();
}
RESULT.close();
return EXIT_SUCCESS;
}
```

ÖZGEÇMİŞ

Adı Soyadı : Altuğ PİŞKİN

Yabancı Dil : İngilizce

E-posta : altug.piskin@ogtm.gov.tr

Eğitim ve Mesleki Geçmişi

Lisans : 1991 İstanbul Teknik Üniversitesi Uçak ve Uzay Bilimleri Fakültesi Uçak Mühendisliği

Y.Lisans : 2002 Eskişehir Osmangazi Üniversitesi Fen Bilimleri Fakültesi Makine Mühendisliği

Doktora : 2022 Eskişehir Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Uçak Gövde Motor Bakım Anabilim Dalı

İnci Hava İkmal ve Bakım Merkezi K.lığı'nda 1992-1997 yılları arasında Turboprop, Turboşaft ve GTC motorları teknik sistem sorumlu mühendisi olarak;

Arçelik kompresör işletmesi Eskişehir de 1997-2002 yılları arasında ürün geliştirme bölümünde prototip ve cad-cam mühendisi olarak;

TUSAŞ Motor Sanayi TEI Eskişehir'de 2002-2022 yılları arasında ARGE bölümünde ısı transferi ağırlıklı olmak üzere çeşitli proje ve görevlerde arge tasarım mühendisi olarak çalışmıştır.

Tez sırasında yapılan yayınlar

- Piskin, A. & Baklacioglu, T. & Turan, O. & Aydin, H.. (2019), “Modeling of Energy Efficiency of a Turboprop Engine using Ant Colony Optimisation”, The Aeronautical Journal. 124. 1-20. 10.1017/aer.2019.134.
- Piskin, Altug & Aktas, Himmet & Topal, Ahmet & Turan, Onder & Baklacioglu, Tolga. (2017). “Rotor Balancing with Turbine Blade Assembly Using Ant Colony Optimization for Aero-Engine Applications”, International Journal of Turbo & Jet-Engines. 38. 10.1515/tjj-2017-0060.

- Piskin Altug & Baklacioglu Tolga & Turan Onder (2021), “A hybrid ant colony–particle swarm optimization method (ACOPSO) for aerospace propulsion systems”, Aircraft Engineering and Aerospace Technology ISSN: 0002-2667, Article publication date: 7 December 2021, DOI: <https://doi.org/10.1108/AEAT-08-2021-0249>.
- Piskin, Altug & Baklacioglu, Tolga & Turan, Onder. (2022). Optimization and off-design calculations of a turbojet engine using the hybrid ant colony – particle swarm optimization method. Aircraft Engineering and Aerospace Technology. ahead-of-print. 10.1108/AEAT-08-2021-0247.

