

HAFİF SULU ÜRETKEN REAKTÖR İÇİN NÖTRONİK ANALİZ

NEUTRONIC ANALYSIS FOR LIGHT WATER BREEDER REACTOR

MUHAMMED ALİM KIRIŞIK

DR. EROL ÇUBUKÇU

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Nükleer Enerji Mühendisliği Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

ÖZET

HAFİF SULU ÜRETKEK REAKTÖR İÇİN NÖTRONİK ANALİZ

Muhammed Alim KIRIŞIK

Yüksek Lisans, Nükleer Enerji Mühendisliği

Tez Danışmanı: Dr. Erol ÇUBUKÇU

Ocak 2023, 107 sayfa

Bu çalışmada, Toryum yakıt çevrimi kullanan ve 1977 ile 1982 yılları arasında Amerika Birleşik Devletleri Shippingport Atomik Güç İstasyonu'nda işletilmiş Hafif Sulu Üretken Reaktör'ün nötronik analizlerinin gerçekleştirilmesi amaçlanmıştır. Anılan modelleme Serpent Monte Carlo kodu ile yapılmıştır. Kor geometrisinin hassas bir üç boyutlu modellemesi için Serpent koduna veriler, bilgisayar destekli bir çizim programı olan FreeCAD kullanılarak üretilmiştir. Ardından, yanma hesaplamaları gerçekleştirilerek fisil izotop üretkenlik parametreleri belirlenmiştir. Reaktörün, reaktivite kontrolünün besleyici yakıt demetlerinin eksenel yöndeki hareketleri ile sağlandığından, her bir yanma adımından sonra anılan besleyicilerin eksenel konumunun yeniden belirlenmesi gerekmiştir. Bu amaçla geliştirilen bir Python yazılımı ile Serpent kodunun gerekli yinelemeleri yapması sağlanmıştır. Reaktör korunun karmaşık geometrisi, doğruya yakın bir sonuç elde edilebilmesi için birçok farklı yanma adımı aralığı ve girdi seçeneklerinin denenmesini gerektirmiştir.

Bu çalışma sonucunda, önerilen reaktörün birin üzerinde bir dönüşüm oranına sahip olduğu, dolayısıyla hafif sulu reaktörler ile üretkenliğe ulaşılabilirdi doğrulanmıştır. Ayrıca, yeterli hesaplama gücü sağlandığında, Hafif Sulu Üretken Reaktör gibi karmaşık geometrilere sahip reaktörlerin Serpent Monte Carlo kodu ile benzeşimi yapılabileceği görülmüştür.

Anahtar Kelimeler: Üretken Reaktör, Monte Carlo Yöntemi, Toryum Yakıt Çevrimi

ABSTRACT

NEUTRONIC ANALYSIS FOR LIGHT WATER BREEDER REACTOR

Muhammed Alim KIRIŞIK

Master of Science, Department of Nuclear Engineering

Supervisor: Dr. Erol ÇUBUKÇU

January 2023, 107 pages

The main objective of this study is to perform the neutronic analysis of the Light Water Breeder Reactor, which operated between 1977 and 1982 at the Shippingport Atomic Power Station, while employing a Thorium fuel cycle. Modelling has been realized using the Serpent Monte Carlo code. To realize an accurate 3-D modelling of the reactor, a computer aided design software FreeCAD has been used together with Serpent. Upon performing the burnup calculations, fissile isotope breeding parameters have been evaluated. Because the reactivity control mechanism of the reactor relies on the axial relocation of seed regions, the appropriate locations need to be determined at the end of each burnup step. To this end, a Python based code has been developed to perform the necessary iterations in Serpent. The complexity of the core geometry necessitated the trial of various input forms and burnup steps, to obtain an accurate performance prediction.

It has been verified that the proposed reactor core can reach a conversion ratio larger than unity; hence, breeding is a possibility with the use light water reactors. Furthermore, it has been demonstrated that when necessary computational power can be provided, Serpent Monte Carlo code may be employed for simulations involving very complex geometries, as in the case of the Light Water Breeder Reactor.

Keywords: Breeder Reactor, Monte Carlo Method, Thorium Fuel Cycle.

TEŞEKKÜR

Lisans ve yüksek lisans eğitimim süresince akademi başta olmak üzere her alanda bana bilgi, tecrübe ve desteğini her zaman sağlayan tez danışman hocam Sayın Dr. Erol Çubukçu'ya,

Tez savunma jürimde yer alan değerli hocalarım, Sayın Prof. Dr. Mehmet Tombakoğlu'na, Sayın Prof. Dr. Ayhan Yılmaz'ın, Sayın Prof. Dr. Şule Ergün'e ve Sayın Prof. Dr. Serhat Yeşilyurt'a,

Tez çalışmam sırasında defalarca yardımlarına başvurduğum çalışma arkadaşlarım Sayın Araş. Gör. Gökhan Pediz ve Sayın Araş. Gör. Uğur Karakurt'a,

Büyük Pedro St. Petersburg Politeknik Üniversitesi Nükleer Mühendislik bölümü yüksek lisans öğrencisi Sayın Emirhan Avcı ve FH-Aachen Uygulamalı Bilimler Üniversitesi Nükleer Mühendislik bölümü yüksek lisans öğrencisi Sayın Sinan Kesik'e,

Tez çalışmam, eğitim sürem ve çalışma hayatım boyunca her türlü manevi desteklerini benden esirgemeyen Hacettepe Üniversitesi Nükleer Enerji Mühendisliği Ailesi'ne sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

ÖZET.....	i
ABSTRACT	ii
TEŞEKKÜR	iii
İÇİNDEKİLER.....	iv
ŞEKİLLER DİZİNİ.....	vi
ÇİZELGELER DİZİNİ	viii
SİMGELER VE KISALTMALAR.....	ix
1. GİRİŞ	1
2. HAFİF SULU ÜRETKEN REAKTÖR	4
2.1. Genel Bilgiler	4
2.1.1. Shippingport Atomik Güç İstasyonu.....	4
2.1.2. LWBR Teknik Özellikleri.....	4
2.2. LWBR Kor Geometrisi	10
2.2.1. Yakıt Demetleri	11
2.2.1.1. Tip I (Type I) Yakıt Modülleri.....	12
2.2.1.2. Tip II ve Tip III (Type II & III) Yakıt Modülleri.....	16
2.2.1.3. Tip IV ve Tip V (Type IV & Type V) Yakıt Modülleri.....	21
2.2.2 Destek Izgaraları	22
2.3.Reaktivite Kontrolü	27
2.4. Üretkenlik Parametreleri	30
3. NÖTRONİK VE YANMA HESAPLAMALARI.....	36
3.1. Nötron Transport Hesaplamaları Kodları.....	36
3.1.1.Monte Carlo Transport Hesaplamaları	36
3.1.1.1.Yüzey İzleme Yöntemi.....	38
3.1.1.2.Delta İzleme Yöntemi	39
3.1.2. Serpent Kodu Transport Algoritması	41
3.2. Yanma Hesaplamaları	43
3.2.2. Bağlantı Şemaları	45

3.2.2.1 Sabit Ekstrapolasyon.....	45
3.2.2.2. Doğrusal Ekstrapolasyon	46
3.2.2.3. Sabit Ekstrapolasyon/Doğrusal İnterpolasyon.....	47
3.2.2.4. Doğrusal Ekstrapolasyon ve Doğrusal İnterpolasyon.....	48
3.2.2.5. Doğrusal Ekstrapolasyon ve İkinci Derece İnterpolasyon.....	49
3.2.2.6. Stokastik Örtük Euler Yöntemi.....	50
4. GİRDİLERİN HAZIRLANMASI VE BENZEŞİM.....	52
4.1 Serpent Kodunda Geometrinin Oluşturulması.....	52
4.2 Python Kodu	59
4.3 Benzeşim.....	62
5. SONUÇLAR VE DEĞERLENDİRME.....	65
5.1 Hesaplamalar	65
5.1.1. Tutarsız Hesaplama Denemeleri	66
5.1.2. Hesaplama 1 ve 2	67
5.1.3. Hesaplama 3 ve 4.....	72
5.1.4. Hesaplama 5 ve 6.....	77
6. TARTIŞMA VE YORUM.....	85
7. DEVAM ÇALIŞMALARI İÇİN ÖNERİLER	87
8. KAYNAKLAR	89
EKLER DİZİNİ	92
EK A. AWBA Reaktörleri	92
EK.A.1 AWBA Prebreeder Reaktörleri.....	92
EK.A.2 AWBA Üretken Reaktör Konseptleri.....	93
EK.A.2.1 Ticari Ölçekli LWBR Konsepti	93
EK.A.2.2 Hareketli ThO ₂ Çubuğu Kontrollü Üretken Reaktör	95
EK.A.2.3 Gelişmiş Hareketli Yakıtlı Üretken Reaktörü	98
EK.A.2.4 Besleyici-Örtü Prebreeder/Üretken Reaktörü.....	99
EK B Python Kodu	99
ÖZGEÇMİŞ	107

ŞEKİLLER DİZİNİ

Şekil 1.1.	Dünya kabuğunda Uranyum ve Toryum bolluklarının karşılaştırması.....	2
Şekil 1.2.	Su soğutmalı reaktörler için U_3O_8 ihtiyacının zamana göre değişimi	3
Şekil 2.1.	LWBR üretkenlik parametrelerinin zamana göre değişimi	5
Şekil 2.2.	Th-232 izotopunun nötron soğurarak ve bozunum yolu ile U-233 izotopuna dönüşümü.	7
Şekil 2.3	LWBR korunun kesit görüntüsü ve modül dizilimi.....	12
Şekil 2.4	Besleyici yakıt demeti kesit görüntüsü ve yakıt çubuğu dizilimi.	13
Şekil 2.5	Tip I yakıt modülü için örtü yakıt demeti kesit görüntüsü ve yakıt çubuğu dizilimi	14
Şekil 2.6	Tip I yakıt modülü için eksenel kesit.	15
Şekil 2.7	Tip I yakıt modülü kesit görüntüsü ve boyutları.....	16
Şekil 2.8	Tip III yakıt modülü için eksenel kesit görüntüsü ve yakıt çubuğu dizilimi.	17
Şekil 2.9	Tip II yakıt modülü için eksenel kesit görüntüsü ve yakıt çubuğu dizilimi... ..	18
Şekil 2.10.	Tip II ve III yakıt modülü için eksenel kesitleri.....	19
Şekil 2.11.	Tip II yakıt modülü kesit görüntüsü ve boyutları.	20
Şekil 2.12.	Tip III yakıt modülü kesit görüntüsü ve boyutları	21
Şekil 2.13.	Tip IV ve V yansıtıcı modülleri için kesit görüntüsü ve boyutları	22
Şekil 2.14.	LWBR Yansıtıcı Demeti Destek Izgaraları	24
Şekil 2.15.	LWBR Örtü Yakıt Demeti Destek Izgaraları.....	25
Şekil 2.16.	LWBR Besleyici Yakıt Demeti Destek Izgaraları	26
Şekil 2.17.	LWBR reaktivite kontrol mekanizması	28
Şekil 2.18	Farklı kor durumları için kritikliği sağlayan besleyici yakıt demeti konumlarının işletme süresine göre değişimi.....	29
Şekil 2.19	Besleyici yakıt demeti reaktivite değerinin tam güç ile işletim süresince değişimi.....	30
Şekil 2.20	U-233, U-235 ve Pu-239 için η değerlerinin soğurulan nötron enerjisine karşı değişimleri.....	31

Şekil 2.21	Parazitik nötron kayıplarının olmadığı varsayılan durum için fisil envanter oranının başlangıç fisil envanter değerlerine göre değişimi.....	32
Şekil 2.22	Parazitik nötron kayıplarının olmadığı varsayılan durum için fisil izotop üretim verimliliğinin başlangıç fisil envanter değerlerine göre değişimi..	33
Şekil 4.1	Besleyici yakıt demeti Serpent modellemesi kesit alanı görünümü	54
Şekil 4.2	Tip I yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü.....	55
Şekil 4.3	Tip II yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü.....	56
Şekil 4.4	Tip III yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü.....	57
Şekil 4.5	LWBR koru Serpent Modellemesi.	58
Şekil 4.6	Besleyici yakıt demetlerinin 24 inç aşağıda olduğu durum için LWBR koru Serpent modellemesi dikey kesit görünümü.....	59
Şekil 4.7.	Python kodu algoritması.	61
Şekil 5.1	1. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	69
Şekil 5.2	2. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	71
Şekil 5.3	3. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	74
Şekil 5.4	4. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	76
Şekil 5.5	5. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	78
Şekil 5.6	6. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları.	80
Şekil 5.7	Hesaplamalar için Serpent çözüm sürelerinin benzeşim zamanına göre değişimi	82
Şekil 5.8	Hesaplama 6 için yakıt bölgelerindeki fisil izotop miktarının zamana göre değişimi.	84
Şekil A.1	Ticari Ölçekte LWBR kor konsepti kesit alan görüntüsü ve modül dizilimi	94
Şekil A.2	Hareketli ThO ₂ Çubuğu Kontroll Üreken Reaktör konsepti için kesit alan görüntüsü ve modül dizilimi.....	96
Şekil A.3	Hareketli ThO ₂ Çubuğu Kontrollü Üreken Reaktör konsepti için zehir çubuklu yakıt demetlerinin kesit alanı ve yakıt dizilimi.....	97
Şekil A.4	Hareketli ThO ₂ Çubuğu Kontrollü Üreken Reaktör konsepti için ThO ₂ kontrol çubuklu yakıt demetlerinin kesit alanı ve yakıt dizilimi	98

ÇİZELGELER DİZİNİ

Çizelge 2.2	LWBR yakıt modülü verileri	10
Çizelge 2.3	AM350 Paslanmaz Çeliği Kompozisyonu	23
Çizelge 5.1	Hesaplama 1 için Serpent girdi seçenekleri	68
Çizelge 5.2	Hesaplama 2 için Serpent girdi seçenekleri ve hesaplama süreleri	70
Çizelge 5.3	Hesaplama 3 için Serpent girdi seçenekleri ve hesaplama süreleri	73
Çizelge 5.4	Hesaplama 4 için Serpent girdi seçenekleri ve hesaplama süreleri	75
Çizelge 5.5	Hesaplama 5 için Serpent girdi seçenekleri ve hesaplama süreleri	77
Çizelge 5.6	Hesaplama 6 için Serpent girdi seçenekleri ve hesaplama süreleri	79
Çizelge 5.7	Hesaplama adımları arasında yapılan değişikliklerin özeti.....	81

SİMGELER VE KISALTMALAR

Simgeler

$\Sigma_{top,N}$	: N bölgesi için toplam makroskobik tesir kesiti
σ_i	: i çekirdeği için mikroskobik tesir kesiti
ϕ	: Nötron akısı
λ_i	: i çekirdeği için bozunum sabiti.

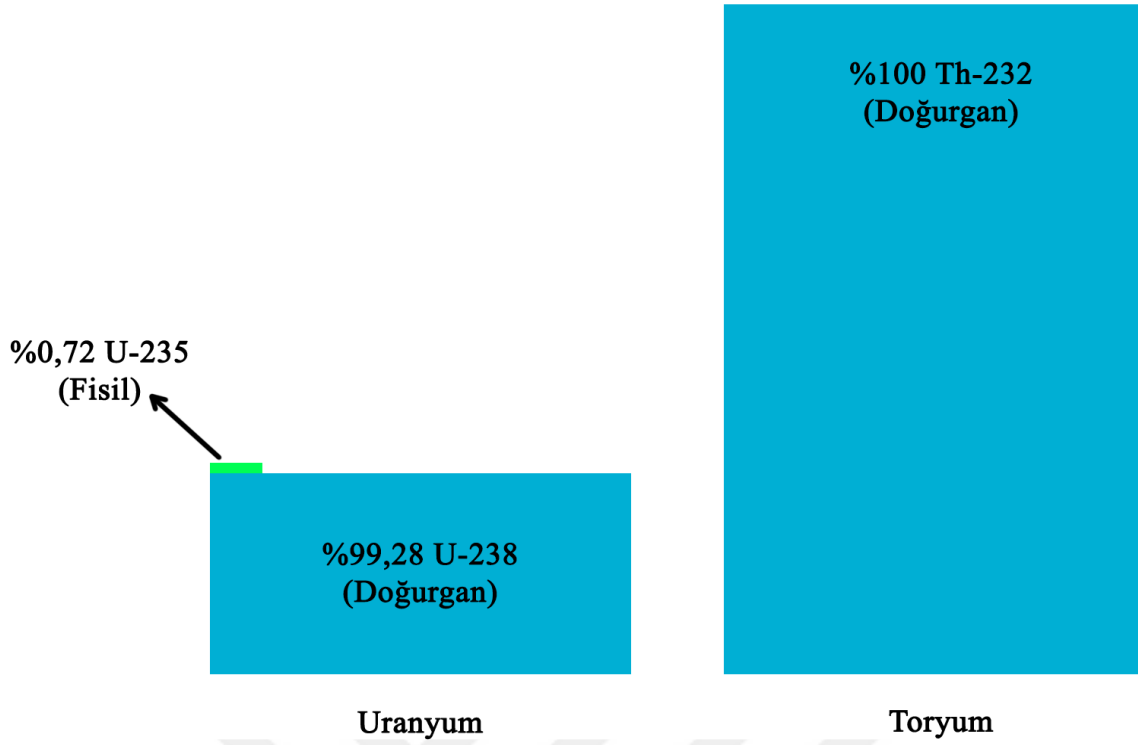
Kısaltmalar

AWBA	: Gelişmiş Hafif Sulu Üretken Reaktör Uygulamaları (Advanced Water Breeder Applications)
BAPL	: Bettis Atomik Güç Laboratuvarı (Bettis Atomic Power Laboratory)
CRAM	: Chebyshev Rasyonel Yaklaşım Metodu (Chebyshev Rational Approximation Method)
DO	: Dönüşüm Oranı (Conversion Ratio)
ETGS	: Efektif Tam Güç Saati (Effective Full Power Hour)
FEO	: Fisil Envanter Oranı (Fissile Inventory Ratio)
LWBR	: Hafif Sulu Üretken Reaktör (Light Water Breeder Reactor)
LWR	: Hafif Sulu Reaktör (Light Water Reactor)
ÖYA	: Önceki Yanma Adımı
PWR	: Basıncılı Su Reaktörü (Pressurized Water Reactor)
TTA	: Dönüşüm Takip Analizi (Transmutation Trajectory Analysis)
YAB	: Yanma Adımı Başlangıcı
YAS	: Yanma Adımı Sonu

1. GİRİŞ

Dünya üzerinde işletimde olan nükleer güç santrallerinin büyük çoğunluğunu Uranyum 235 (U-235) izotopuna zenginleştirilmiş UO_2 veya UO_2 - PuO_2 karışımı MOX yakıtları ile enerji üreten hafif sulu reaktörler (Light Water Reactor – LWR) oluşturmaktadır. Plütonyum elementi doğada bulunmayan bir element olmakla birlikte U-235 izotopu da doğal Uranyum madenlerinde sadece %0,72 civarı bir bollukta bulunur. Uranyum maden yataklarının dağılım haritaları incelendiğinde de bu yatakların dünya üzerinde belli coğrafyalarda yoğunlaştığı görülebilir [1]. Öte yandan Toryum hem dünya üzerinde daha bol bulunan bir elementtir hem de maden yatakları dünya üzerinde daha eşit bir şekilde dağılmıştır [2]. Bu sebeple toryum elementinin nükleer enerji üretimine kazandırılması uzun yıllardır çalışılan bir konu olmuştur.

1960'lı yıllarda Amerika Birleşik Devletleri Enerji Bakanlığı, o zaman için bilinen Uranyum maden rezervlerinin sınırlılığı sebebi ile nükleer enerji alanındaki tasarım odaklarını nükleer güç santrallerinde Toryum elementi kullanımına ve böylece enerji kaynaklarından faydalanma miktarını arttırmaya yöneltmişlerdir [3]. Bu alanda yapılan çalışmaların sonucunda soğutucu ve moderatör olarak basınçlandırılmış hafif su kullanan, Toryum ve Uranyum 233 (U-233) yakıtlarına sahip üretken bir reaktör olan Hafif Sulu Üretken Reaktör (Light Water Breeder Reactor – LWBR) tasarlanmıştır. 1977 yılının sonbaharında Amerika Birleşik Devletleri'nin Pennsylvania eyaletinde bulunan Shippingport Atomik Güç İstasyonu'nda işletimine başlayan LWBR, 1982 yılına kadar güç üretmeye devam etmiştir. 60 MW elektrik gücünde olan reaktörün, 5 yıllık işletme sürecinin sonunda başlangıçtaki fisil izotop miktarından yüzde 1,39 daha fazla fisil izotop miktarına sahip olduğu gözlemlenmiştir [4].

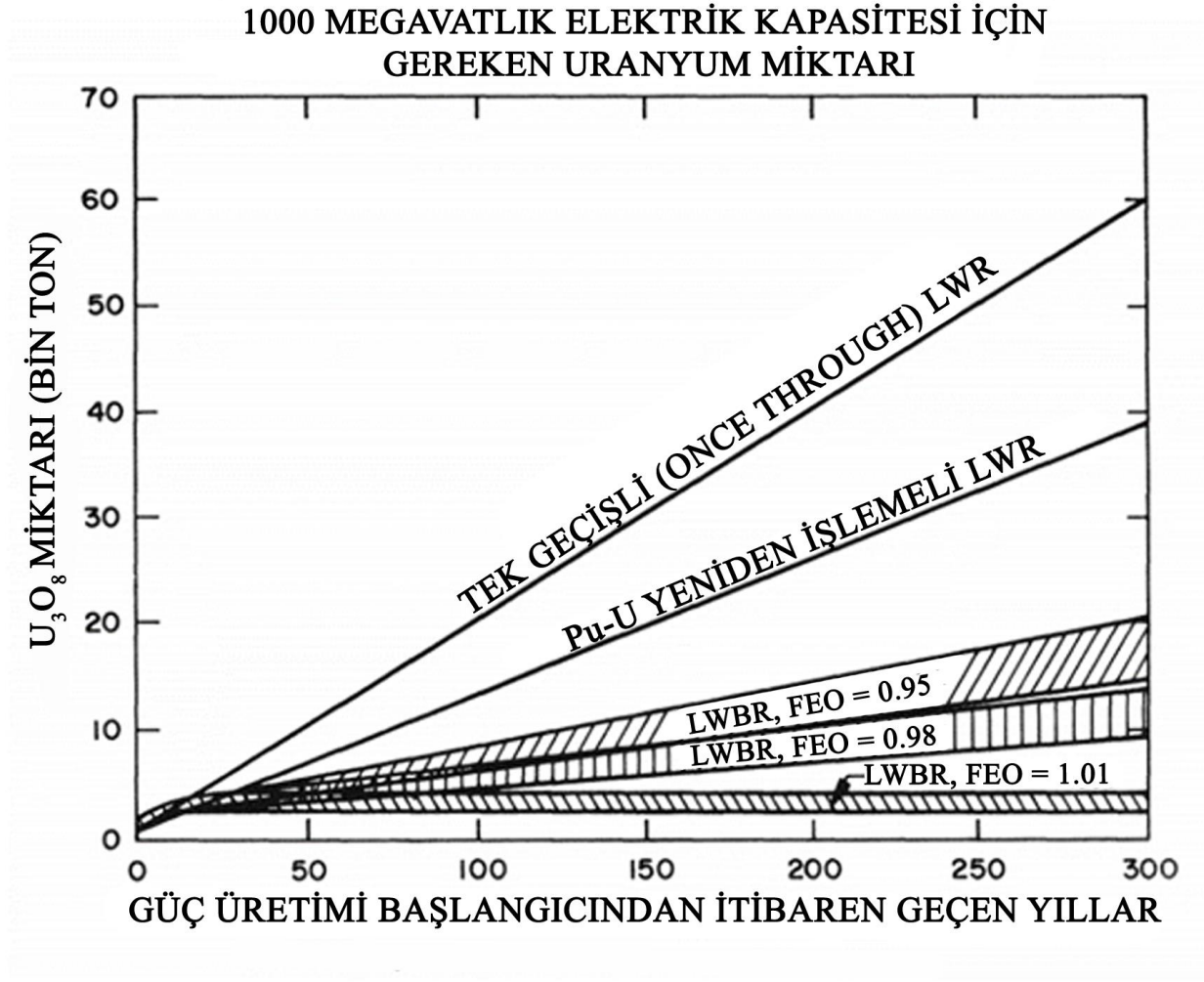


Şekil 1.1. Dünya kabuğunda Uranyum ve Toryum bolluklarının karşılaştırması [1][2]

Son yıllarda uranyum rezervlerinin sınırlılığı yeniden gündeme gelen bir konudur. Hem ülkemizde hem de dünyanın geri kalanında, nükleer mühendislik alanında en çok konuşulan ve tartışılan konulardan biri Ergimiş Tuz Reaktörleri ve bu reaktörlerden dolayı toryum yakıt çevrimi olmuştur. Birçok ülke tarafından ilgili birçok çalışma ve proje olmasına rağmen Ergimiş Tuz Reaktörlerinin henüz kanıtlanmış teknolojiler oldukları söylenemez.

Toryum kullanımı ile gündeme gelen bir başka reaktör türü ise Basınçlı Ağır Su Reaktörü olan CANDU reaktörleridir. Fakat her ne kadar uzun bir süre bu reaktörlerin Toryum ile yeterli üretkenlikte çalışabileceği öne sürülmüş olsa da yakın zamanda yapılan birçok çalışma, yeniden işleme (reprocessing) kayıpları hesaba katılınca bu reaktörlerde Toryum ile net üretkenliğin elde edilemeyeceği yönünde sonuç vermiştir [5].

Toryum elementinden enerji üretme çalışmaları günümüzde devam etmekte ve gelecekte bu çalışmaların yoğunluk kazanması öngörülmektedir. Bu çalışmanın amacı, Toryum yakıt çevrimi (Th-U233) ile çalışmış ve üretkenlik konusundaki başarısı kanıtlanmış olan Hafif Sulu Üretken Reaktör için Serpent Monte Carlo kodu [6] ile üretkenlik performansının benzeşiminin gerçekleştirilmesidir.



Şekil 1.2. Su soğutmalı reaktörler için U_3O_8 ihtiyacının zamana göre değişimi [7]

2. HAFİF SULU ÜRETKEN REAKTÖR

2.1. Genel Bilgiler

2.1.1. Shippingport Atomik Güç İstasyonu

Nükleer enerji ile elektrik üretmek amacıyla kurulan ticari ölçekli güç istasyonlarının ilki olan Shippingport Atomik Güç İstasyonu (AGİ), Amerika Birleşik Devletleri'nin Pensilvanya Eyaletinde bulunmaktadır. Shippingport AGİ'de ilk kurulan reaktör, kritikliğe 1957 yılında ulaşmıştır. Bu reaktör, bir basınçlı hafif su reaktörü olmakla birlikte besleyici ("seed") ve örtü ("blanket") yakıt demetlerinden oluşan koru, uçak gemilerinde kullanılabilecek şekilde tasarlanmıştır. Tasarımın besleyici bölgelerinde %93 zenginlikte U-235 izotopu olan Uranyum yakıtı kullanılırken, örtü bölgelerinde doğal zenginlikte Uranyum yakıt kullanılmıştır [3].

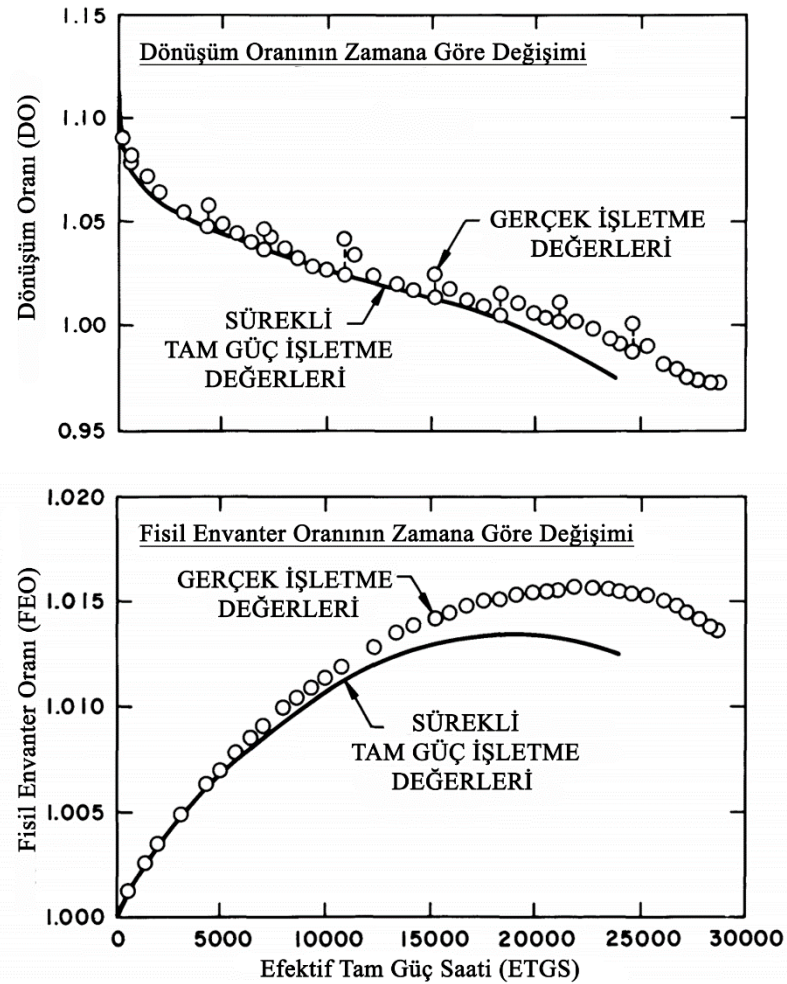
Shippingport AGİ'ye kurulan ikinci kor da ilki gibi besleyici ve örtü yakıt bölgelerinden oluşan bir basınçlı hafif su reaktörüdür fakat bu reaktör ilkine kıyasla daha yüksek enerji elde edilmesi ve daha uzun işletim ömrü hedeflenerek tasarlanmıştır [8]. Bu reaktörün işletim ömrü 1965 yılında başlamış olup 1974 yılında sona ermiştir. 1977 yılına gelindiğinde de Shippingport AGİ'nin üçüncü ve son reaktörü, Hafif Sulu Üretken Reaktör ("Light Water Breeder Reactor") işletmeye geçmiştir

2.1.2. LWBR Teknik Özellikleri

LWBR için araştırma ve geliştirme çalışmalarına 1965 yılında, Amerika Birleşik Devletleri Atom Enerjisi Komisyonu'nun nükleer maden kaynaklarından yararlanma oranını arttırma yönündeki taleplerinin sonucunda başlanmıştır. Bu bahsedilen talebin doğmasındaki en etkili bileşen, o yıllarda mevcut varlığı bilinen Uranyum kaynaklarının nükleer enerjiyi uzun yıllar idare edemeyeceğine dair öngörülerdi (Daha sonraki yıllarda keşfedilen yeni Uranyum rezervleri ve yeni madencilik teknikleri bu görüşün değişmesinde etkili olmuştur) [9]. Bu sebeple o yıllarda Amerika Birleşik Devletleri'nin nükleer enerji alanındaki odağı Toryum yakıt çevrimi olmuştur. Yani LWBR projesinin ana amacı, Toryum ve U-233 yakıt çevriminin ticari ölçekte elektrik enerjisi üretiminde kullanılabileceğini ve bu elektrik üretimi üretkenlik sınırının üstünde sürdürerek yeni fisil izotopların oluşumunun da sağlanacağını ispatlanmasıydı. LWBR için araştırma ve

geliştirme safhası 1976 yılında son bulmuş ve reaktör Shippingport AGİ'ne kurularak 1977 ve 1982 yılları arasında beş yıl işletmede bulunmuştur.

Tam güçte olduğu zaman reaktörün üretimi 60 MWe'dır. Reaktörün tasarımında 18.000 Efektif Tam Güç Saati (ETGS) kadar enerji üretimine denk gelen bir ömür öngörülmüş olsa da reaktör işletilmeye devam edilip 29.000 ETGS'ye yakın bir enerji üretimi elde edildikten sonra reaktörün işletimi son bulmuştur. Beş yıllık işletim sürecinin sonunda LWBR'ün Fisil Envanter Oranı (FEO) 1,0139 olarak ölçülmüştür, yani LWBR koru, işletmesi sona erdiğinde işletmeye başlamadan önceki ilk durumuna göre %1,39 daha fazla fisil izotop içerdiği görülmüştür [4].



Şekil 2.1. LWBR üretkenlik parametrelerinin zamana göre değişimi [9]

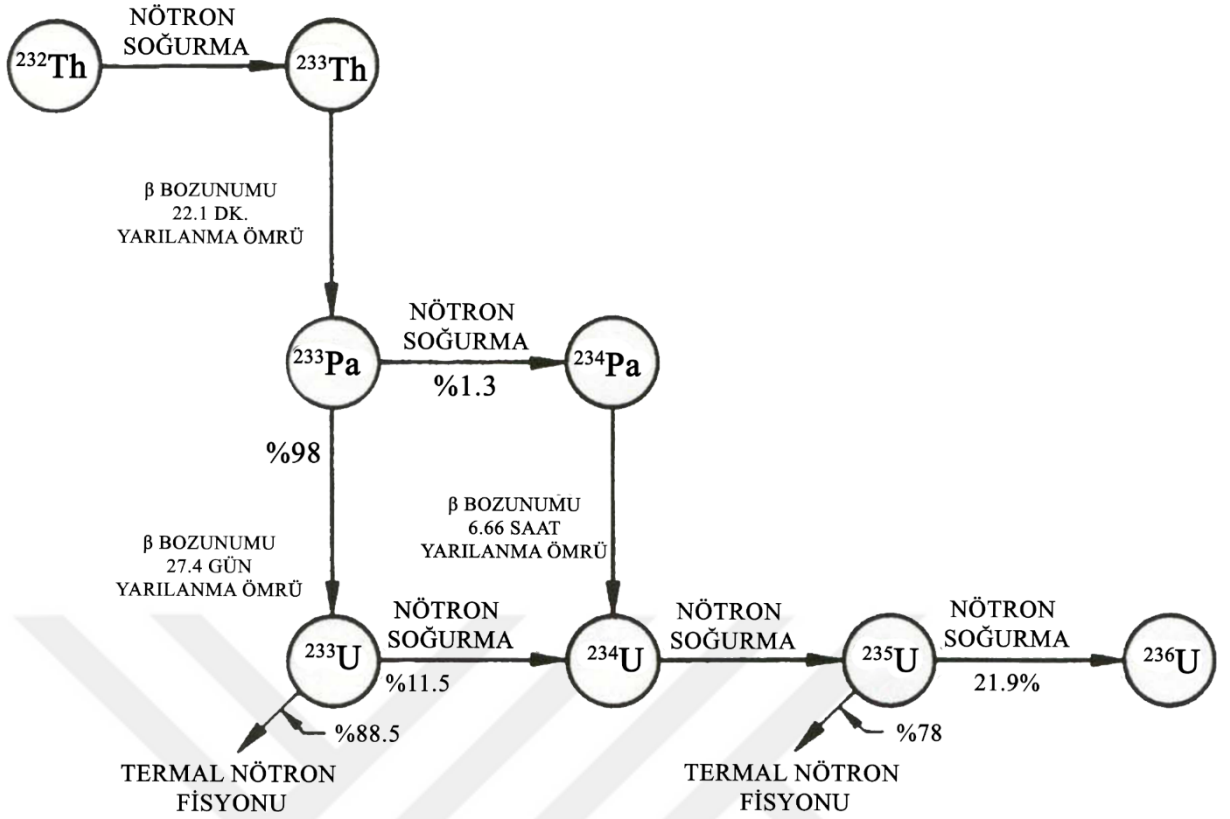
LWBR reaktörü için Şekil 2.1.'de görüleceği üzere sürekli tam güçte çalışma durumu için hesaplanan üretkenlik parametreleri ile işletme geçmişi için ölçülen değerler farklılık gösterir. İşletim geçmişinde reaktörün tam gücün altında işletildiği veya sıfır güçte

kalınan sürelerde dönüşüm oranı Xe-135 ve Pa-233 gibi nötron soğurumu için toryum ile rekabet halinde olan ve parazitik nötron kayıplarına sebep olan izotopların azalması ile bir miktar artar. LWBR işletim geçmişinde FEO, 22.300 ETGS civarında en büyük değeri olan 1,015'e ulaşmıştır [9].

Shippingport AGİ'deki ilk iki reaktör koruna benzer şekilde LWBR de besleyici ve örtü yakıt bölgelerinden oluşur. Besleyici yakıt bölgelerindeki yakıt çubuklarının fisil izotop zenginlikleri, yakıt çubuğu yarıçapları ve yakıt merkez mesafeleri, örtü bölgesine yeterli verimlilikte nötron akısı sağlayacak şekilde tasarlanmıştır.

U-233 doğada bulunan bir izotop olmadığı için LWBR'ü için gereken yakıt çevrimi iki aşamalı olarak düşünülebilir. Bu yakıt çevriminin ilk aşaması üretkenlik öncesi, ("pre-breeder") olarak adlandırılır ve bu aşamada U-235 gibi fisil izotopların yakıt olarak kullanıldığı toryum içeren bir reaktörde U-233 izotopunun üretilmesidir. Daha sonra ikinci aşama olan üretkenlik aşamasında ("breeder phase") önceki aşamada üretilen U-233 izotopları Toryum ile birlikte yakıt olarak kullanılır ve harcanan fisil izotopları telafi edecek şekilde yeni fisil izotopların üretilmesi sağlanır [7].

Üretken reaktörlerde fisil izotop üretimi, doğurgan ("fertile") izotopların nötron soğurması ve takip eden bozunumlar sonrası fisil izotoplara dönüşmesi ile sağlanır. LWBR için bu durum, Toryum 232 (Th-232) izotopu ile gerçekleşir. Nötron soğurması sonrası Th-232 izotopu, Th-233 izotopuna dönüşür ve bu izotop 22,1 dakika yarılanma ömrü ile Beta ışıması yaparak Protaktinyum 233 (Pa-233) izotopuna dönüşür. Daha sonra 27,4 günlük yarılanma ömrü ile bu izotop da Beta ışıması yapar ve fisil U-233 izotopuna dönüşür. Böylece doğurgan Th-232 izotopundan fisil U-233 izotopu elde edilmiş olur.



Şekil 2.2. Th-232 izotopunun nötron soğurarak ve bozunum yolu ile U-233 izotopuna dönüşümü. [9]

LWBR projesindeki esas odak bu üretkenliğin sağlanması olduğu için nötron ekonomisinin en az kayıpla sağlanması, reaktörün tasarımında dikkat edilmesi gereken önemli bir etkeni. Bu amaçla geliştirilen bazı tasarımsal farklılıklar, LWBR'ü normal bir PWR ile kıyaslandığında özgün kılmaktadır. Bu farklılıklar:

- PWR için yaygın olan kare dizilimi yerine LWBR'de yakıt çubukları altıgen geometrili yakıt demetlerinde, üçgen dizilimindedir. PWR ile karşılaştırıldığında adım-çap oranı (pitch to diameter ratio) da farklıdır, üretkenliği arttırmak için bu oran LWBR'lerde daha düşük tutulmuştur.
- Reaktördeki reaktivite kontrolü çözünebilir ve yanabilir zehirlerle (soluble & burnable poison) veya nötron soğurucu kontrol çubuklarıyla değil, eksenel yönde yukarı ve aşağı hareket edebilen besleyici yakıt demetleriyle sağlanmaktadır, böylece parazitik nötron kayıpları minimize edilmiştir.
- Destek ızgaralarında (Support grids) gereken hacmi azaltmak için Zirkonyum Alaşımı yerine AM-350 paslanmaz çeliği kullanılmıştır.

- Tm yakıt demetlerinin zirkonyum alařımından yapılmıř kılıflar bulunmaktadır. Besleyici yakıt demetleri iin bu kılıf yakıt demetlerinin ve yansıtıcı demetlerin etrafında bulunurken rt demetlerinde bu kabuk yapısı yakıt demetinin i tarafındaki altıgen blgenin etrafında, hareketli besleyici blge ile sabit rt blgeleri arasında kalan kanal iin kılavuz tp grevinde kullanılmaktadır.



Çizelge 2.1 LWBR koru genel özellikleri [10]

Reaktör Tipi	Besleyici ve örtü bölgelerinden oluşan, basınçlı hafif su moderatör ve soğutuculu üretken reaktör.
Reaktivite kontrolü	Değişken yakıt geometrisi
Termal Güç (MW)	236,6
Elektriksel Güç (MW)	60
Kor Yüksekliği (inç)	104
Kor Çapı (inç)	90
Soğutucu Kor Giriş Sıcaklığı (°C)	271,11
Soğutucu Kor Çıkış Sıcaklığı (°C)	282,78
Soğutucu Çalışma Basıncı	11,135 MPa
Toplam Modül Sayısı	27
Yakıt Malzemeleri:	UO ₂ -ThO ₂ , ThO ₂
Zarf Malzemesi	Zirkaloy-4
Destek Izgarası Malzemesi	AM350

Çizelge 2.2 LWBR yakıt modülü verileri [10]

	Besleyici Yakıt Demetleri	Standart Örtü Demetleri	Güç Düzleştirici Örtü Demetleri	Yansıtıcı Demetler
Yakıt Dış Çapı (inç)	0,306	0,5715	0,5275	0,832
Pelet Dış Çapı	0,2520	0,5105	0,4695	0,7415
Zarf Kalınlığı (inç)	0,0220	0,02775	0,02625	0,042
Izgara Kalınlığı (inç)	0,0135	0,014	0,014	0,018
Izgara Sayısı	9	8	8	6
Aktif Yakıt Bölgesindeki Izgara Sayısı	7,5	6,5	6,5	6
Toplam Çubuk Sayısı	7428	3234	3581	3047

2.2. LWBR Kor Geometrisi

Hafif Sulu Üretken Reaktör (LWBR), soğutucu ve moderatör olarak basınçlandırılmış hafif su kullanılan, kritikliğini termal veya epitermal enerjilerdeki nötronların fisyon tepkimeleri ile sağlayan ve fisil izotopları kullanarak enerji ürettiği sırada yeni fisil izotopların üretimini de gerçekleştiren (yani Üretken veya “Breeder”) bir reaktördür. LWBR reaktör korunda 12 adet yakıt modülü ve bu yakıt modüllerini çevreleyen 15 adet yansıtıcı modül bulunur. Yakıt modülleri düzgün veya düzgün olmayan altıgen geometride olup hareketli ve hareketsiz yakıt demetlerinden oluşur. Yakıt modüllerinin merkezinde düzgün altıgen geometride besleyici olarak adlandırılan, eksenel yönde hareket edebilen yakıt demeti bölgesi bulunur. Besleyici yakıt demetinin etrafında ise yakıt modülünün kor içinde bulunduğu bölgeye bağlı olarak bir veya iki farklı, hareketsiz örtü yakıt demeti bölgesi bulunur. Reaktörün kritikliği, hareketli besleyici yakıt demetinin eksenel yöndeki hareketi ile sağlanır.

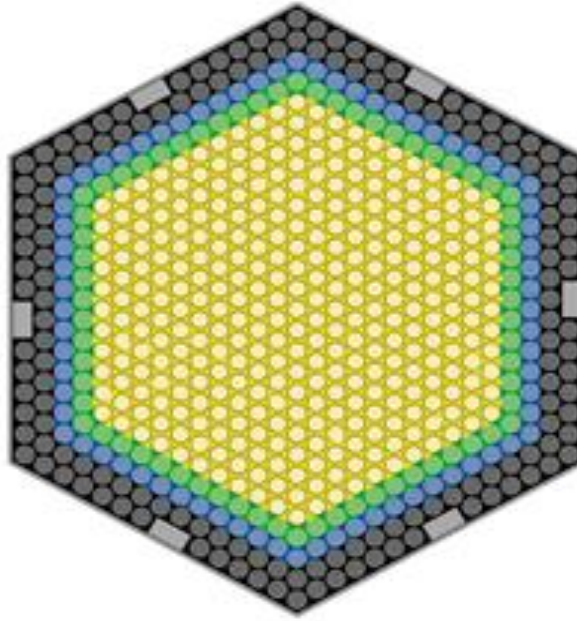
LWBR korundaki yakıt çubuklarını ThO_2 peletler veya farklı fisil zenginlikteki UO_2 - ThO_2 peletlerinin yığıtları (“stack”) oluşturmaktadır. Tezin geri kalanında Uranyum- ThO_2 içeren peletlerin yakıt çubuğu içerisinde üst üste oluşturdıkları yükseklik için ikili yığıt olarak bahsedilecektir. İkili yığıtların uzunlukları ve fisil izotop zenginlikleri, yakıt çubuklarının bulundukları yakıt demeti tiplerine ve yakıt demeti içerisindeki konumlarına göre değişiklik göstermektedir. Bunun yanında bu peletlerde bulunan Uranyum elementleri %98 zenginlikte U-233 izotopu içermektedir [10]. Yakıt çubuklarındaki aktif bölgeleri toplam 104 inç yüksekliğe sahiptir. Bu yüksekliğin en alttaki ve en üstteki 10 inçlik kısımlarında bütün yakıt çubuklarında ThO_2 peletler bulunmaktadır. Bu iki bölgenin arasında kalan 84 inçlik yükseklikte ise yakıt çubuklarının yakıt demeti içinde bulundukları konumlara göre farklılık gösteren oranlarda ikili veya tekli (ThO_2) yığıtlar bulunmaktadır. Besleyici yakıt demetinin iç kısımlarındaki yakıt çubuklarında 84 inçlik kısmın tamamı ikili yığıtlardan oluşurken dış taraflardaki yakıt çubuklarında ikili yığıt yükseklikleri kademeli olarak azalarak tekli ThO_2 peletleri ile birlikte yer alırlar.

2.2.1. Yakıt Demetleri

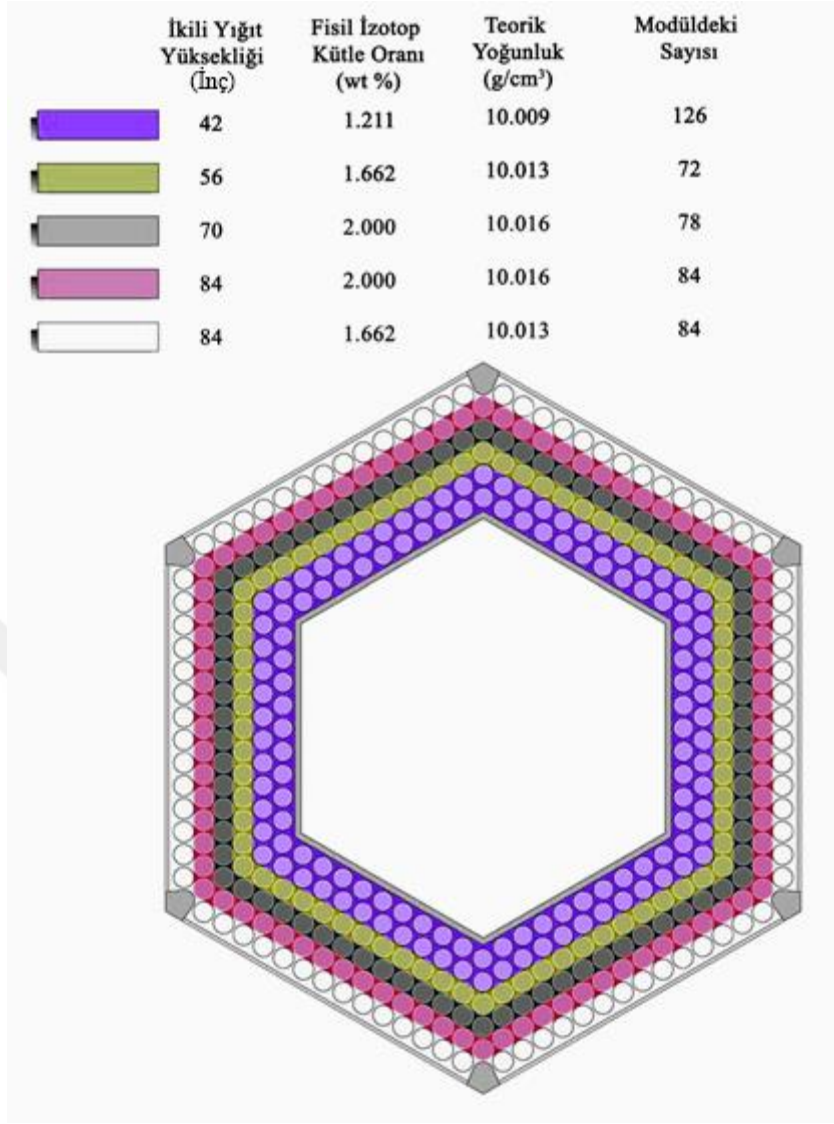
LWBR korunda bulunan 12 yakıt modülü, 15 adet yansıtıcı modül ile çevrelenmiştir. Bahsedilen 12 yakıt modülünün her biri, merkezlerinde bulunan ve eksenel yönde hareket edebilen besleyici yakıt demeti bölgesi ile bu besleyici yakıt bölgesini çevreleyen hareketsiz örtü yakıt demeti bölgelerinden oluşur. Besleyici yakıt demeti bölgeleri 12 yakıt modülü için de aynı düzgün altıgen geometriye sahiptir, bununla beraber örtü bölgelerinin geometrileri, bulundukları modüllerin kor içerisindeki konumlarına göre üç farklı tipte değişiklik gösterir. Korun en merkezinde bulunan üç modül (1. Tip modüller) düzgün altıgen geometriye sahipken bu üç modülü çevreleyen diğer 9 modül (2. Ve 3. Tip modüller) iki farklı düzgün olmayan altıgen geometrisinden birine sahiptir. Yakıt modüllerini çevreleyen yansıtıcı modüller ise düzgün olmayan altıgen veya beşgen geometrilerine sahiptirler. Korda bulunan toplam 27 modül, bahsedilen bu 5 farklı geometri tipine göre kategorize edilmiştir.

1. Tip yakıt modülleri için farklı başlangıç fisil izotop zenginliği ve ikili yığıt yüksekliği değerleri şekil 2.3, 2.4 ve 2.5'te gösterilmiştir.

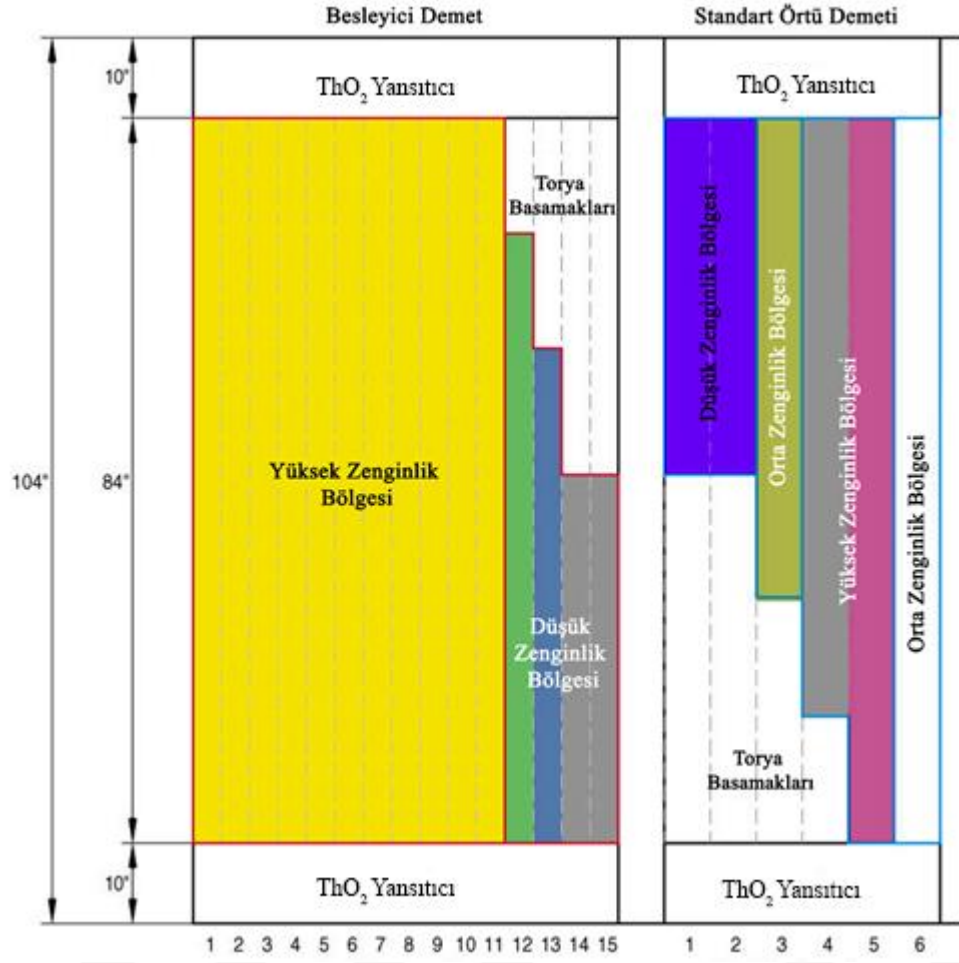
	İkili Yığıt Yüksekliği (inç)	Fisil İzotop Kütle Oranı (wt %)	Teorik Yoğunluk (g/cm ³)	Modüldeki Sayısı
	84	5.195	10.042	331
	70	4.327	10.035	66
	56	4.327	10.035	72
	42	4.327	10.035	150



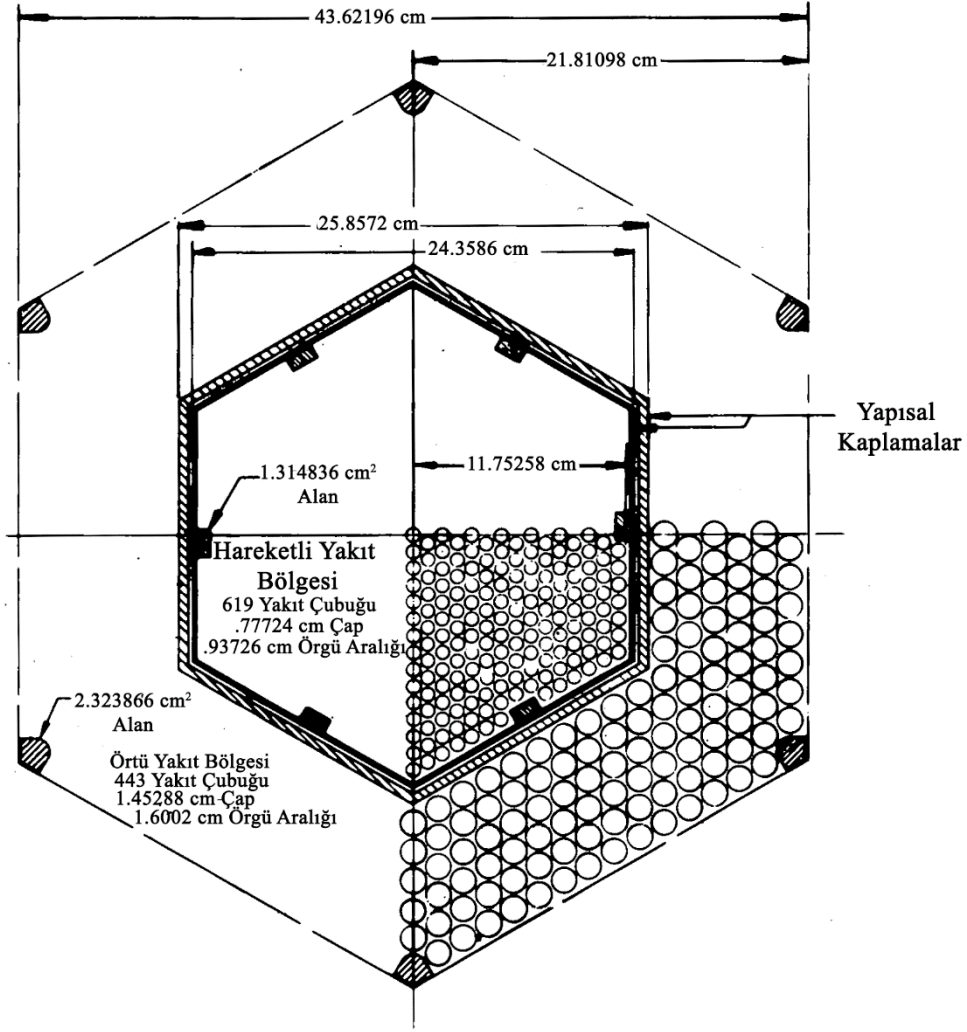
Şekil 2.4 Besleyici yakıt demeti kesit görüntüsü ve yakıt çubuğu dizilimi [10].



Şekil 2.5 Tip I yakıt modülü için örtü yakıt demeti kesit görüntüsü ve yakıt çubuğu dizilimi [10].



Şekil 2.6 Tip I yakıt modülü için aksenal kesit [10].



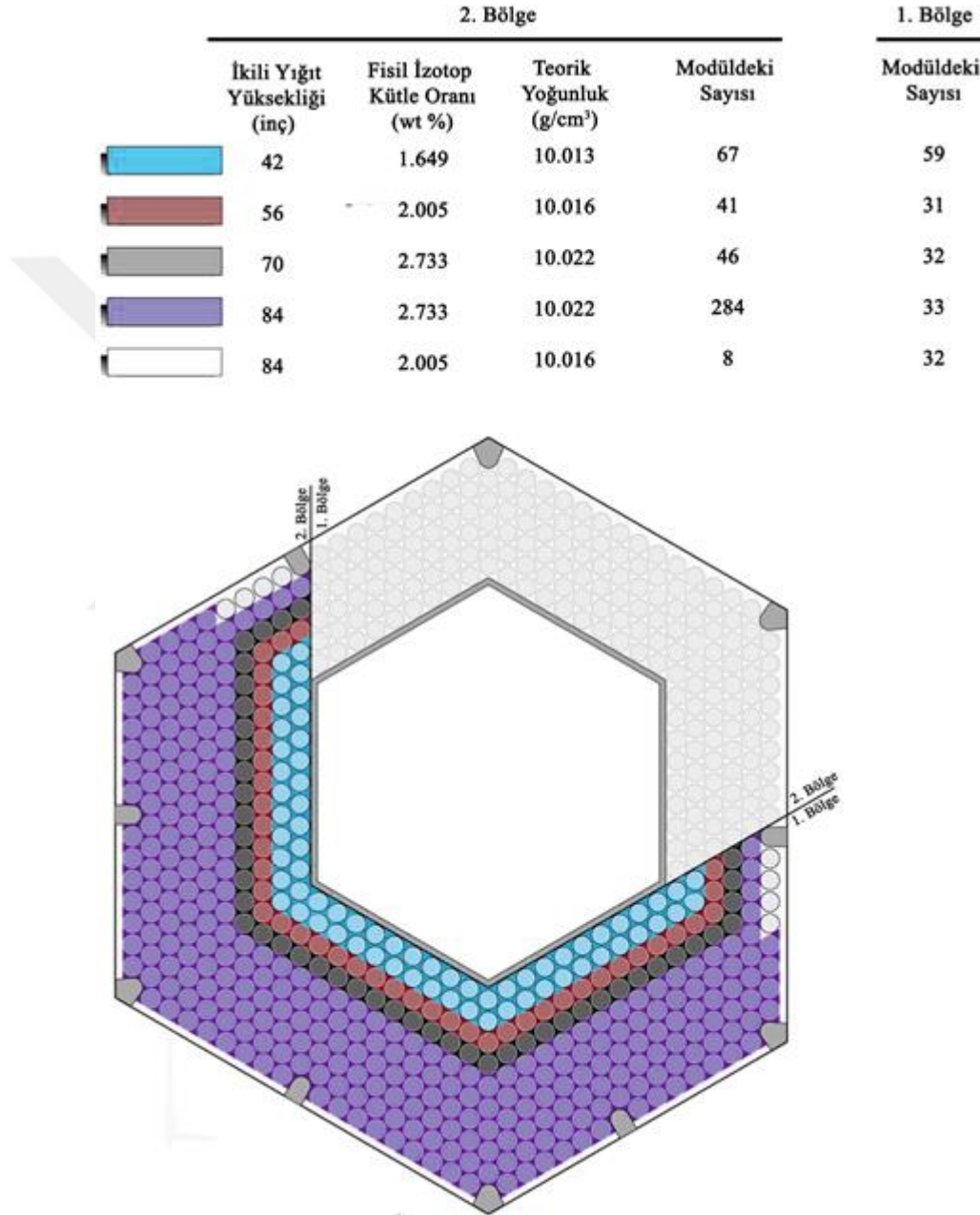
Şekil 2.7 Tip I yakıt modülü kesit görüntüsü ve boyutları [11].

2.2.1.2. Tip II ve Tip III (Type II & III) Yakıt Modülleri

2. ve 3. Tip yakıt modülleri, 1. Tip yakıt modüllerinin etrafında bulunan yakıt modülleridir. 2. Ve 3. Tip yakıt modüllerinde besleyici yakıt demetleri 1. Tip modüllerdeki besleyici yakıt demetleri ile aynıdır. Bu yakıt modüllerinde 1. Tip yakıt modülünden farklı olarak örtü yakıt demeti iki farklı bölgeden oluşur. Örtü bölgesinin diğer yakıt modülleri ile komşu olan bölgeleri standart örtü bölgesi iken yansıtıcı modüller ile komşu olan dış bölgeleri “Güç Düzleştirici” örtü bölgeleridir. Güç düzleştirici örtü bölgelerinde ikili yığıt yükseklik dizilimleri standart örtü bölgesininki ile aynı olması ile birlikte başlangıç fisil izotop zenginliği ve yakıt çubuğu çapları standart örtü bölgesininkinden farklıdır. Güç düzleştirici örtü bölgesi, standart örtü bölgesindeki

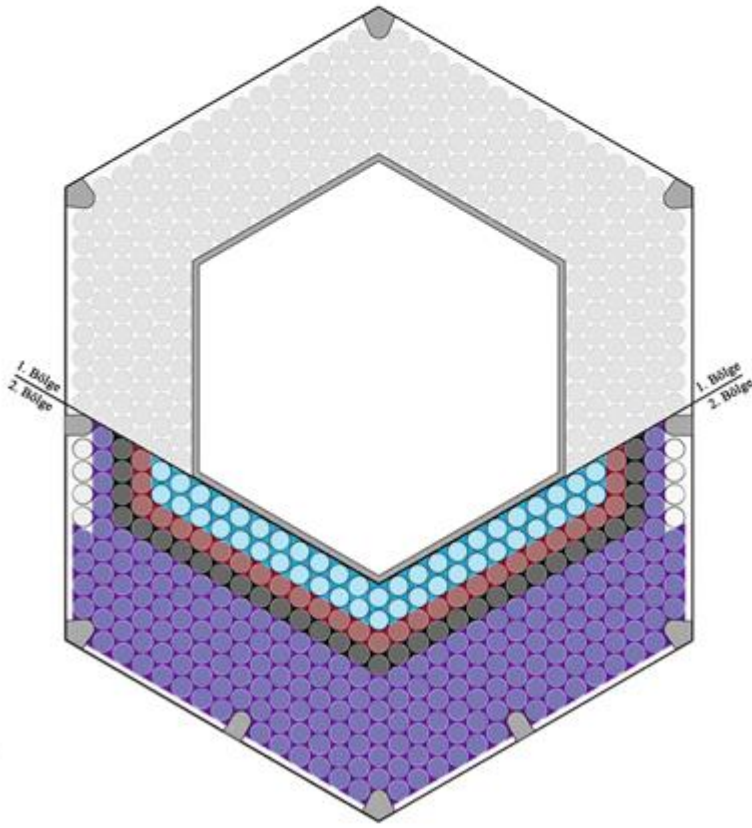
yakıt çubuklarına kıyasla daha fazla fisil izotop zenginliğine sahiptir, böylece kor içinde daha tekdüze bir güç dağılımı sağlanır.

2. ve 3. Tip yakıt modülleri arasındaki fark ise, 3. Tip yakıt modüllerinin yansıtıcı modüller ile komşu olan kenar sayısının daha fazla olması, yani güç düzleştirici bölgelerinin yakıt modülü içerisinde 2. Tip yakıt modüllerine göre daha fazla alan kapsamasıdır.

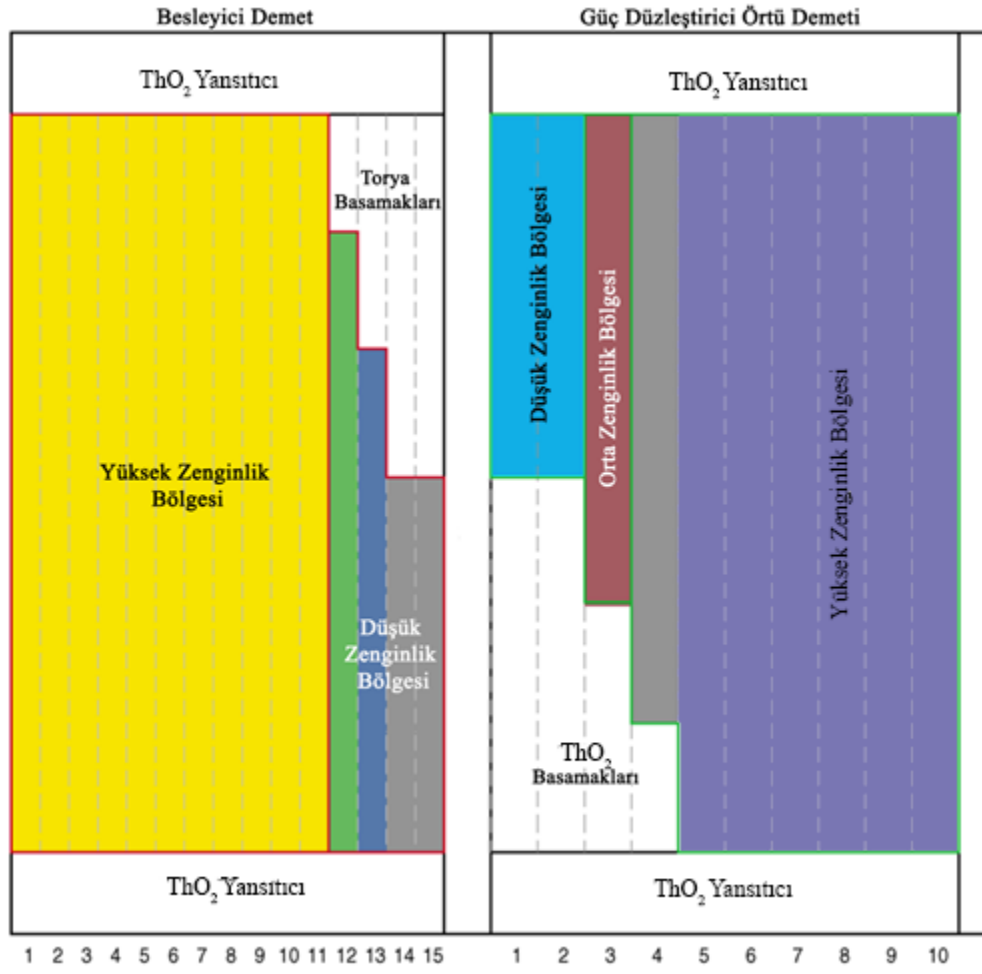


Şekil 2.8 Tip III yakıt modülü için aksenal kesit görüntüsü ve yakıt çubuğu dizilimi [10].

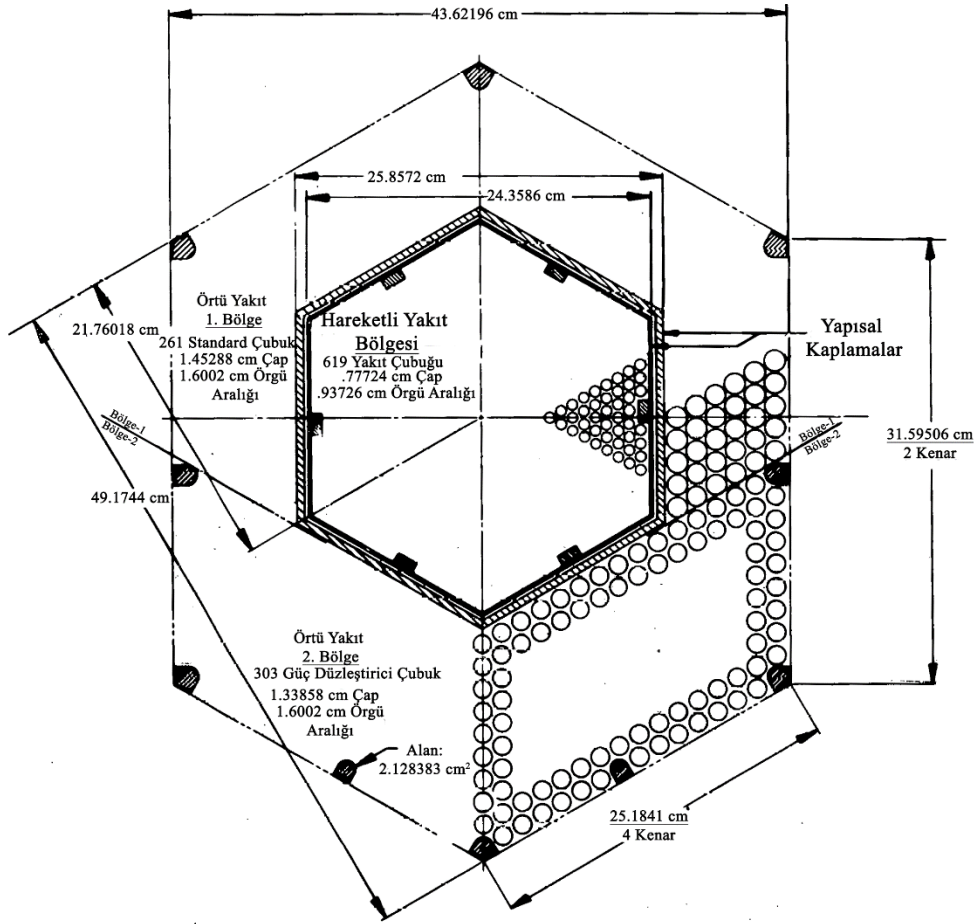
	2. Bölge				1. Bölge
	İkili Yığıt Yüksekliği (inç)	Fisil İzotop Kütle Oranı (wt %)	Teorik Yoğunluk (g/cm ³)	Modüldeki Sayısı	Modüldeki Sayısı
	42	1.649	10.013	46	80
	56	2.005	10.016	29	43
	70	2.733	10.022	33	45
	84	2.733	10.022	187	47
	84	2.005	10.016	8	46



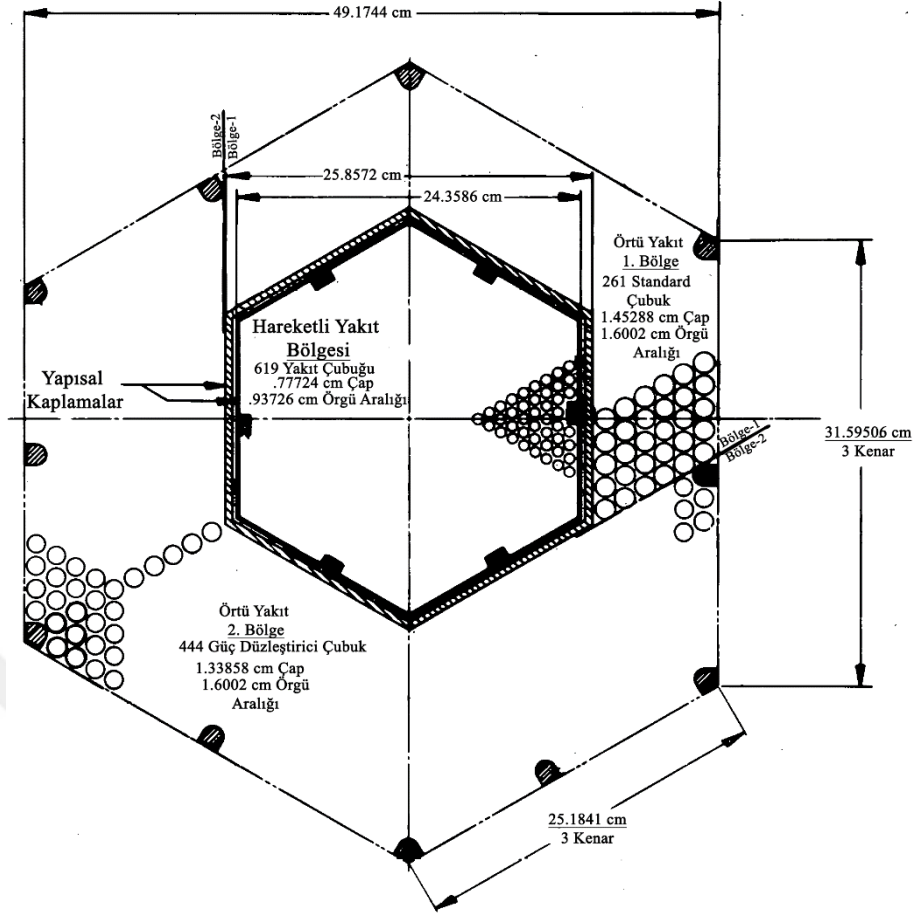
Şekil 2.9 Tip II yakıt modülü için eksenel kesit görüntüsü ve yakıt çubuğu dizilimi [10].



Şekil 2.10. Tip II ve III yakıt modülü için aksenal kesitleri [10].



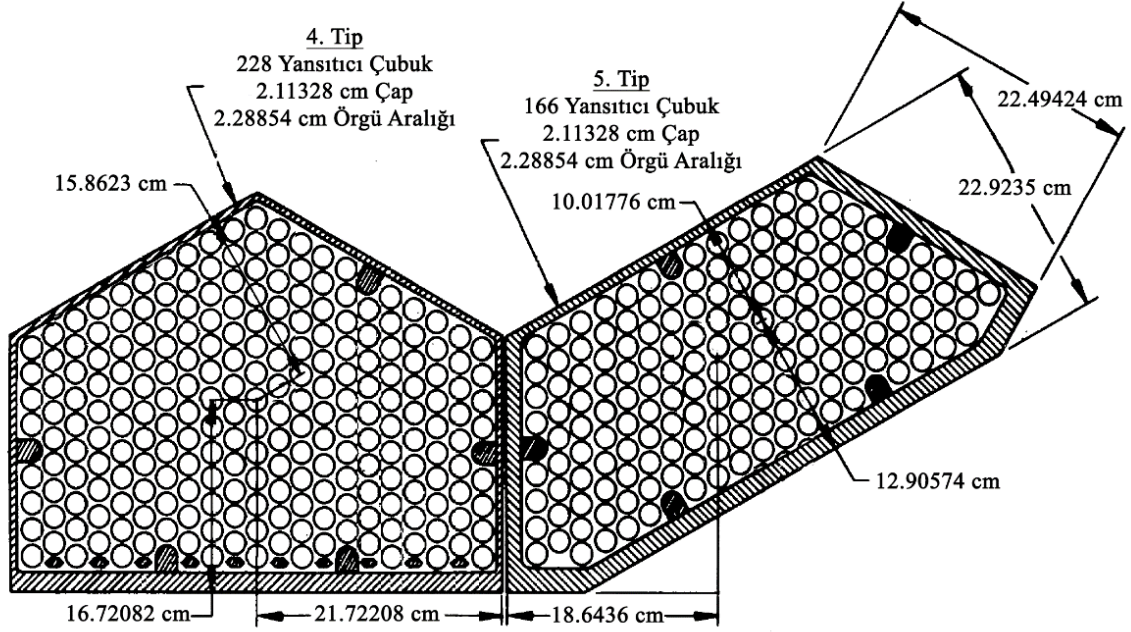
Şekil 2.11. Tip II yakıt modülü kesit görüntüsü ve boyutları [11].



Şekil 2.12. Tip III yakıt modülü kesit görüntüsü ve boyutları [11].

2.2.1.3. Tip IV ve Tip V (Type IV & Type V) Yakıt Modülleri

Bu modülleri 1,2 ve 3. Tip yakıt modüllerinin etrafında bulunan yansıtıcı modülleri oluşturur. Yansıtıcı modülleri başlangıçta fisil izotop içermez ve tamamen ThO_2 içeren peletlerden oluşur. Bu modüllerin korda bulunma amacı kordan kaçan nötron kayıplarını toplam nötronların % 0,8'i civarına düşürmektir[4]. Bu yüzdenin daha düşük olacağı daha büyük LWBR kor tasarımları, yani Shippingport LWBR'ünden daha yüksek üretkenliğe sahip reaktörler de mümkündür[4].



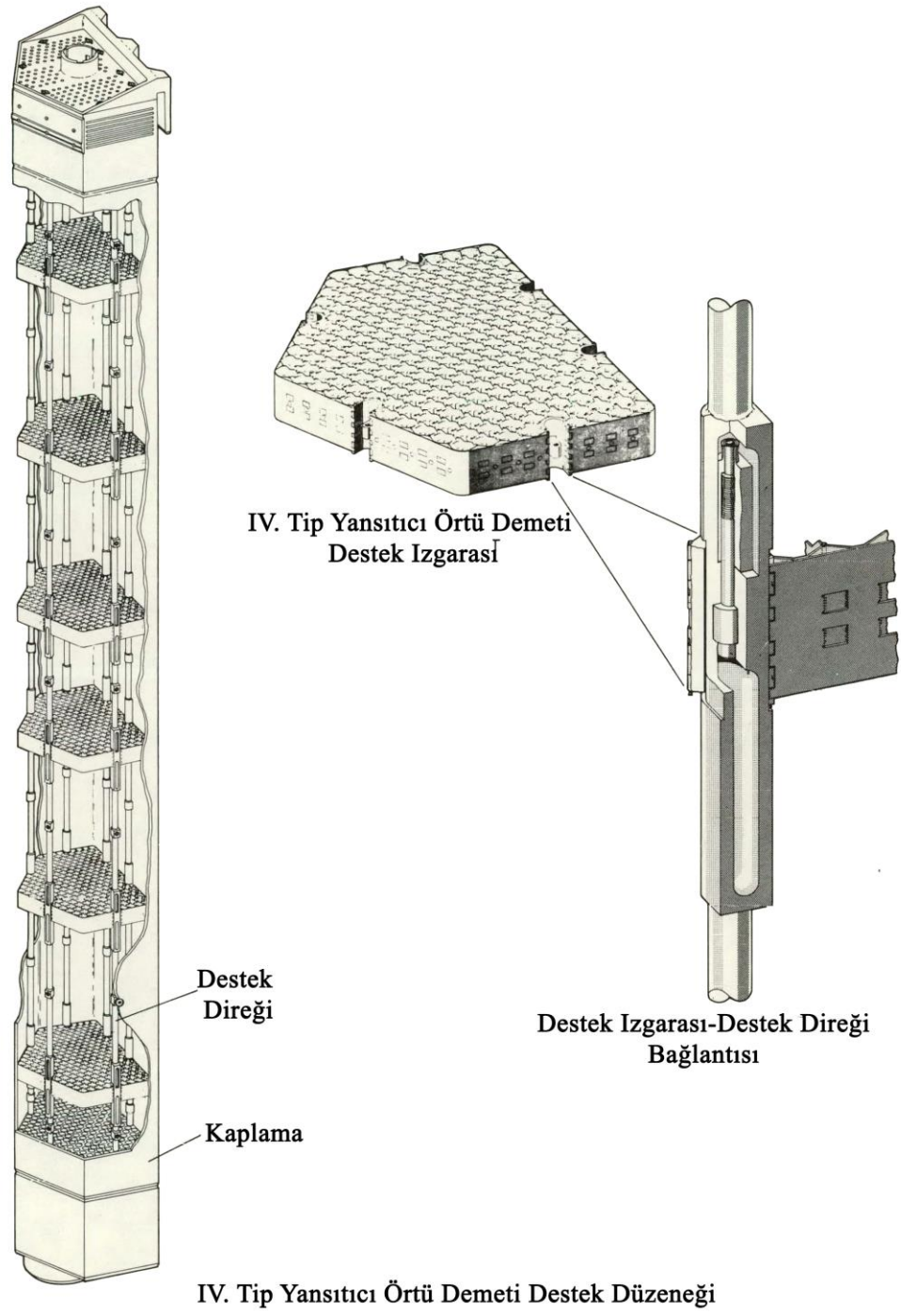
Şekil 2.13. Tip IV ve V yansıtıcı modülleri için kesit görüntüsü ve boyutları [11].

2.2.2 Destek Izgaraları

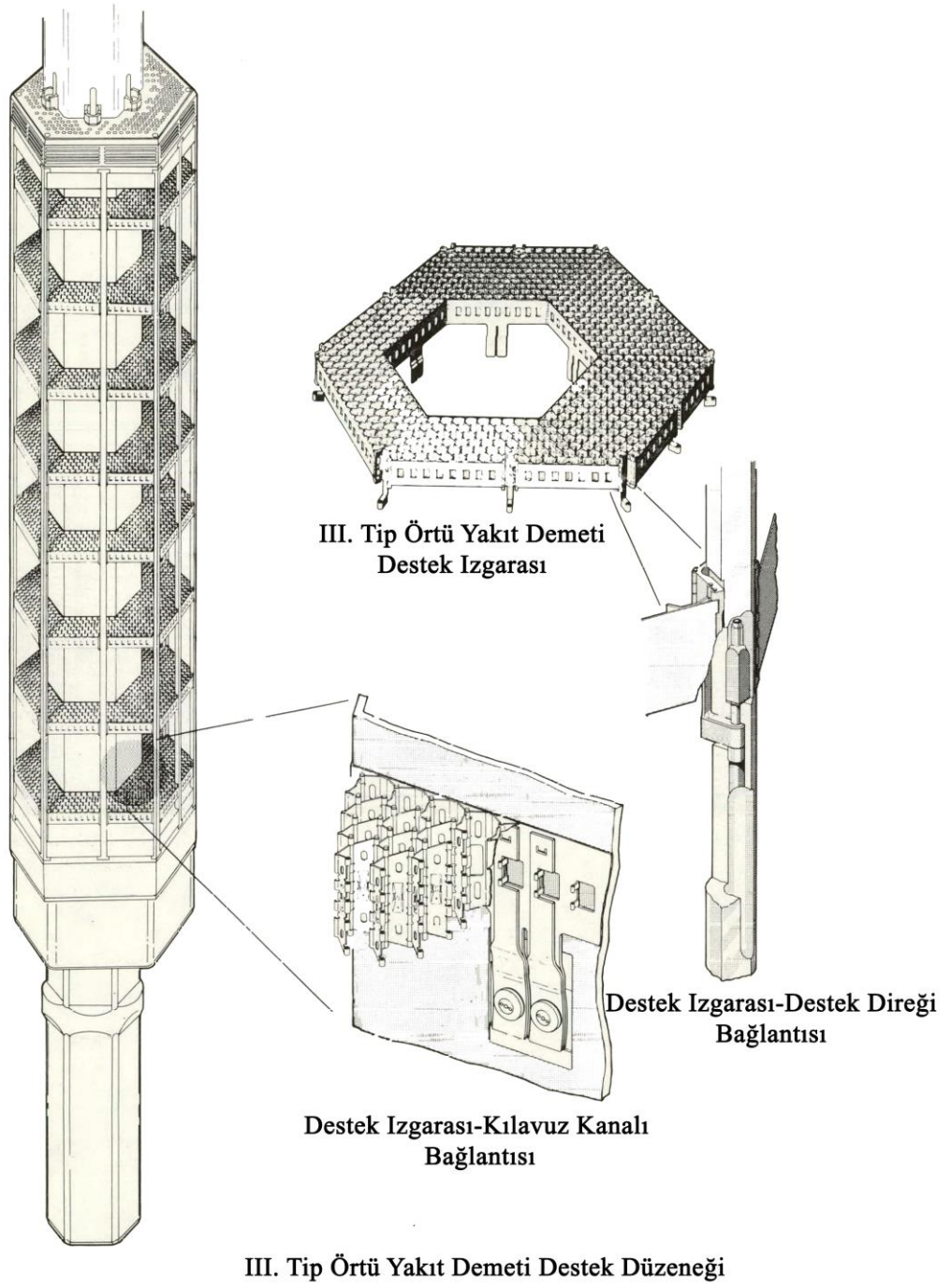
Nükleer reaktörlerdeki yakıt çubuklarının boyutlarında işletme süresi boyunca çeşitli değişimler meydana gelir. Bu değişimler fisyon ürünlerinin etkisiyle yakıt peletlerinin şişmesi ile veya zarf malzemesinin radyasyon ve basınç gibi etkenlerle kalınlığının değişmesi ile oluşur ve zaman içinde yakıt çubuklarının eğilmesine sebep olur. Yakıt çubuklarında gözlenebilecek eğilmeler ve akış sebebi ile oluşabilecek titreşimler reaktör işletimi sırasında performans ve güvenlik açılarından sorun oluşturabilecek faktörlerdir. Yakıt demetlerinde yaygın reaktörlere kıyasla daha sık yakıt çubuğu dizilimine sahip LWBR için de destek ızgaraları tasarım sürecinde önem arz eden yapılar olmuştur. Hafif sulu reaktörlerde yaygın olarak kullanılan zirkonyum alaşımlarının daha düşük nötron soğurma tesir kesitlerine sahip olmalarına rağmen mekanik özellikleri sebebi ile LWBR destek ızgaralarında AM350 paslanmaz çeliği kullanılması tercih edilmiştir [12].

Çizelge 2.3 AM350 Paslanmaz Çeliği Kompozisyonu [12]

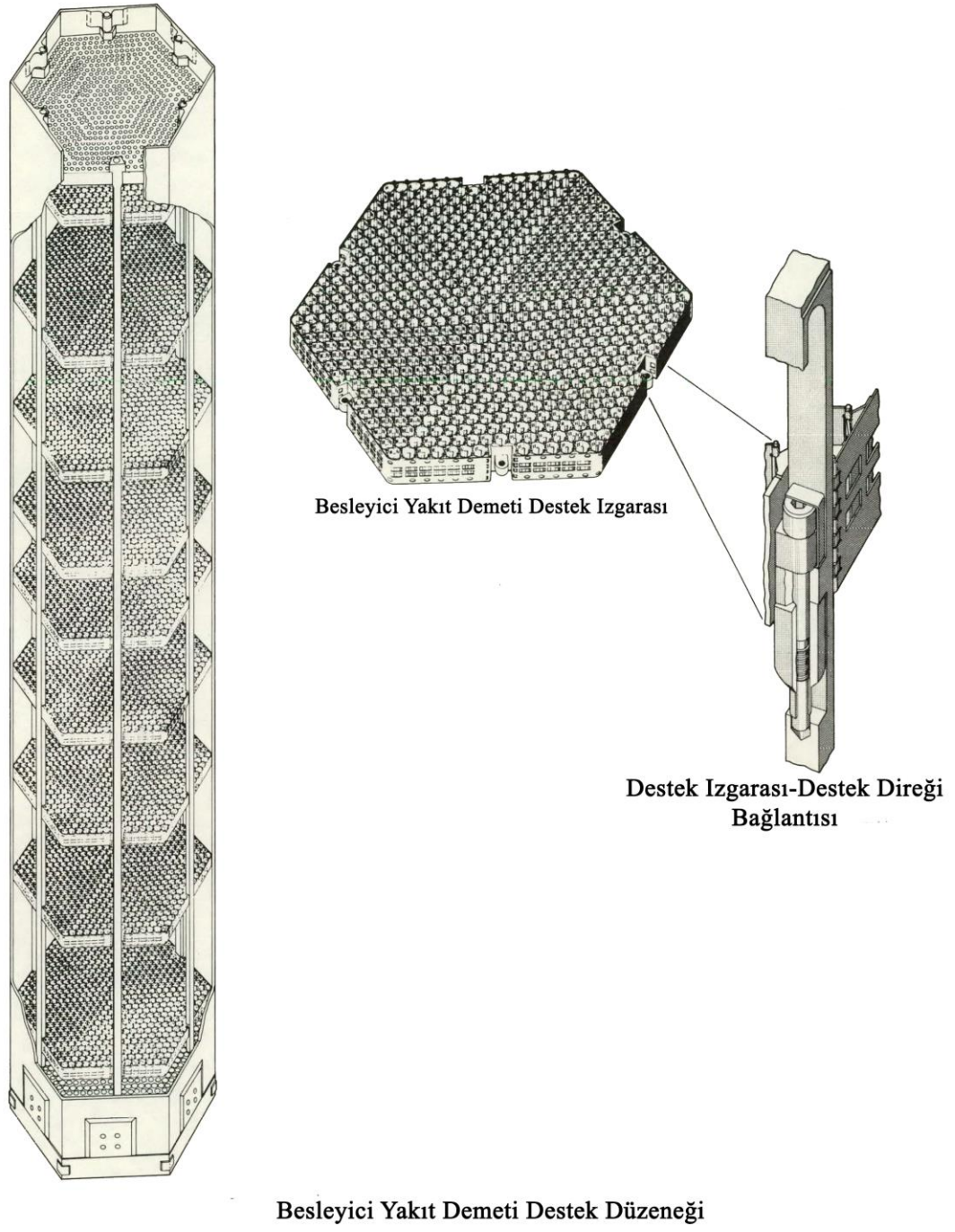
Element	Kütlece Yüzdesi (%)
Karbon	0,08 – 0,12
Krom	16,00 – 17,00
Manganez	0,50 – 1,25
Molibden	2,50 – 3,25
Nikel	4,00 – 5,00
Azot	0,07 – 0,13
Fosfor	0,04 (en çok)
Silikon	0,50 (en çok)
Sülfür	0,30 (en çok)
Demir	Dengeleyici



Şekil 2.14. LWBR Yansıtıcı Demeti Destek Izgaraları [12].



Şekil 2.15. LWBR Örtü Yakıt Demeti Destek Izgaraları [12].



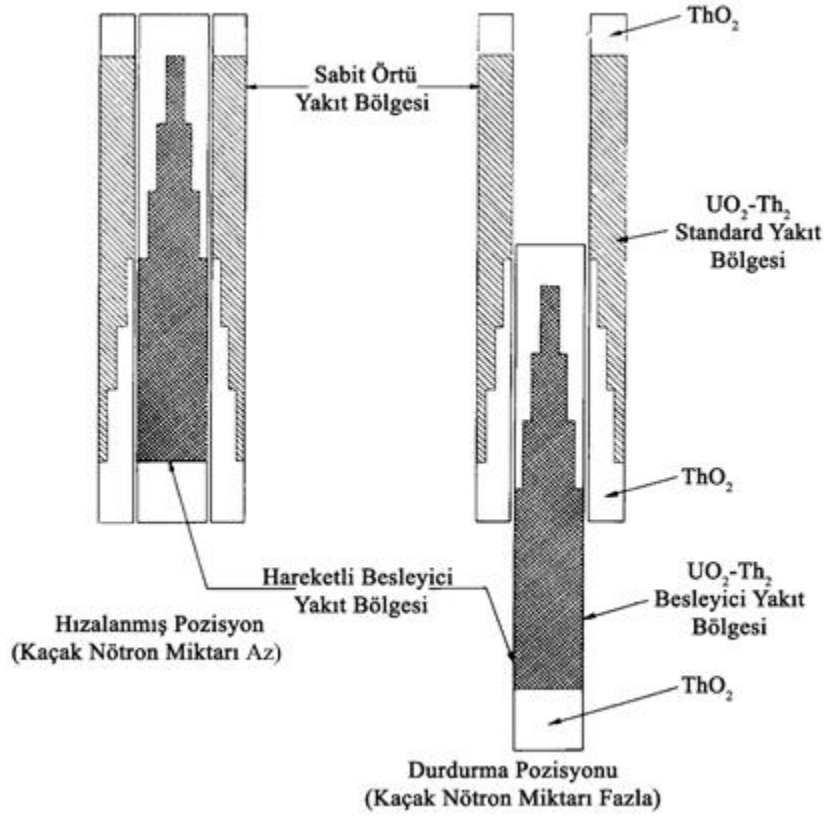
47

Şekil 2.16. LWBR Besleyici Yakıt Demeti Destek Izgaraları [12].

2.3.Reaktivite Kontrolü

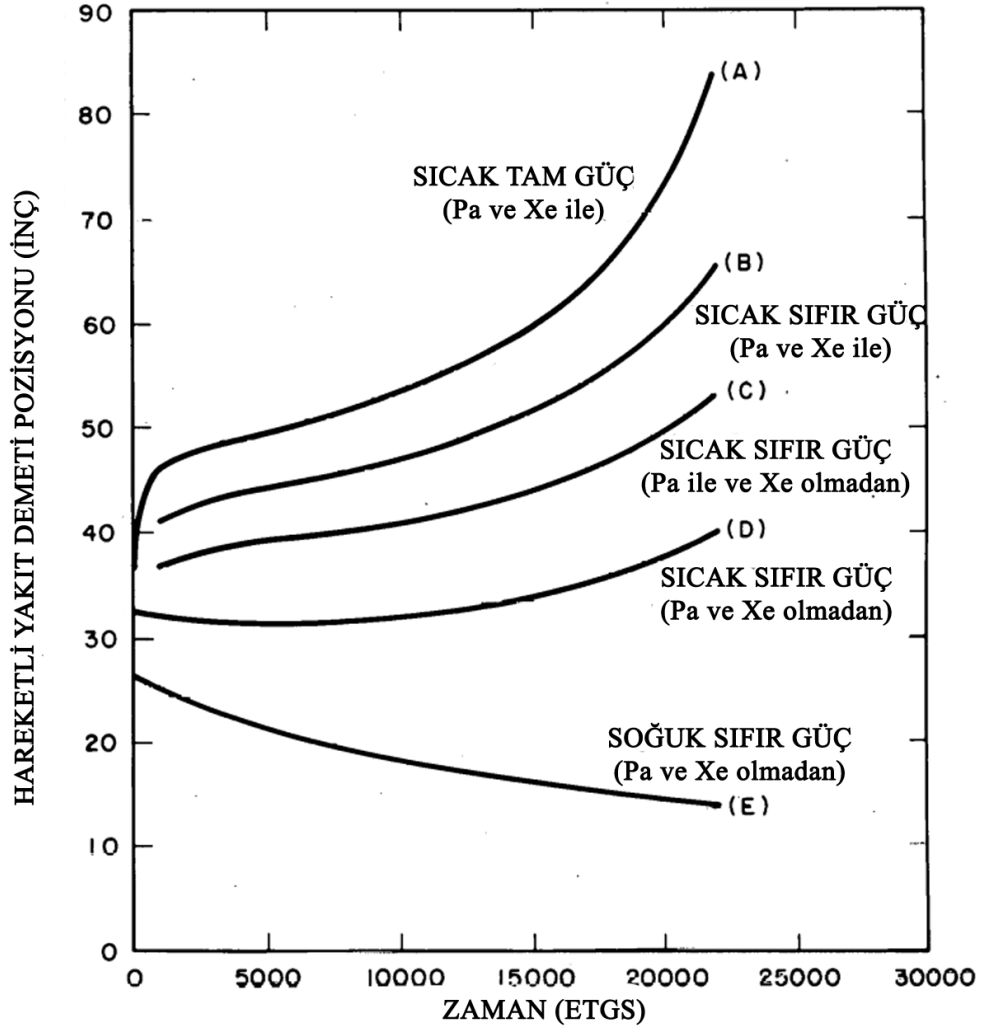
LWBR’de işletim sırasında oluşan reaktivitenin kontrolü, diğer hafif sulu reaktörlerdeki nötron soğurucu kontrol çubukları veya kimyasal kontrol yöntemleri yerine besleyici yakıt bölgelerinin dikey ekseninde hareket ettirilmesi ile sağlanır.

LWBR besleyici demetlerindeki yakıt çubukları, kordaki örtü demetlerindeki yakıt çubuklarına kıyasla daha fazla fisil izotop zenginliğine sahiptirler. Bu sebeple bu yakıt demetleri için nötron çoğaltma faktörünün değeri, yani k_{∞} birden fazladır. Benzer şekilde daha az zenginliğe sahip örtü yakıt demetlerinde k_{∞} birden azdır [13]. Besleyici yakıt bölgesinin hareketi yüksek k_{∞} bölgesinden düşük k_{∞} bölgesine kaçan nötron miktarını değiştireceği için böylece kordaki reaktivite kontrolü sağlanmış olur. Hareketli besleyici yakıt bölgeleri ile sabit duran örtü bölgelerinin birbirleri ile kesiştikleri sınır kısımları, Şekil 2.6, 2.10 ve 2.17’de gösterilen ThO₂ basamakları denen dizilimi oluşturacak şekilde ThO₂ peletlerinden oluşur. ThO₂ basamakları, fisil izotoplara sahip bölgeleri birbirinden ayırdığı gibi aynı zamanda fazla nötronların doğurgan Th-232 izotopları tarafından soğurularak korda yeni fisil izotopların üretimini sağlar. Böylece reaktörde üretkenlik sağlanmış olur. LWBR için de reaktivitenin kontrolü diğer hafif sulu reaktörlerde olduğu gibi nötron soğurucu kontrol çubukları veya kimyasal kontrol yöntemleri ile sağlanmaya çalışılırsa bu malzemeler nötronların bir kısmını soğurup nötron ekonomisinde parazit kayıplara sebep olacak, böylece reaktörde üretkenlik sağlanamayacaktır [13].



Şekil 2.17. LWBR reaktivite kontrol mekanizması [13].

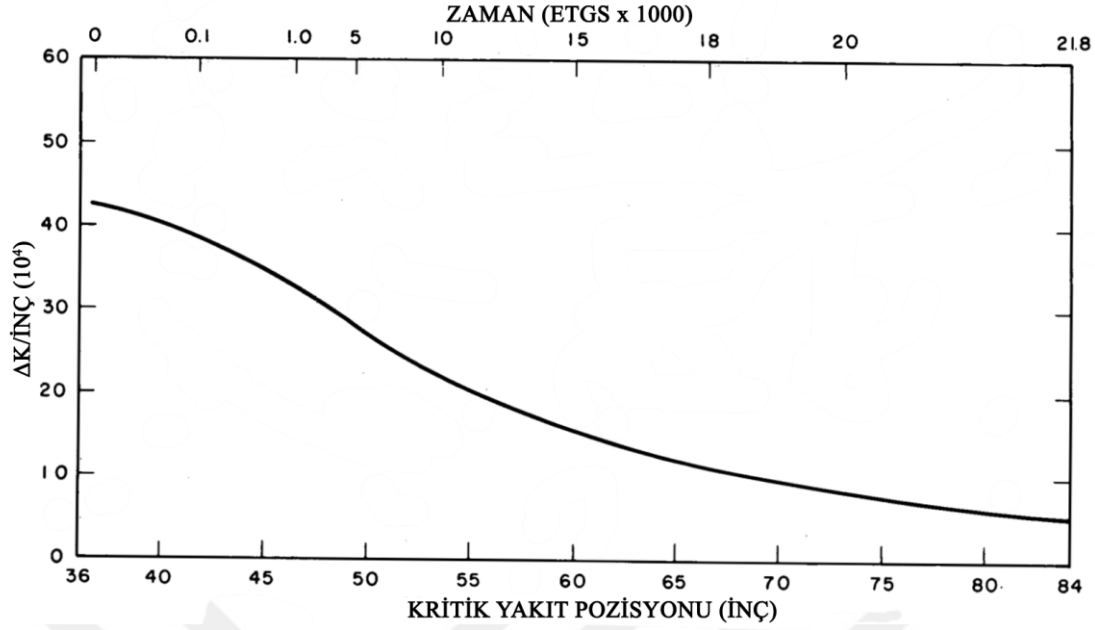
LWBR normal işletimi sırasında reaktör korunda bulunan 12 adet hareketli besleyici yakıt demetinin hepsi dikey ekseninde aynı konumda hareket eder. Böylece hem reaktördeki reaktivite kontrolünün daha basitleşmesi hem de kor boyunca tekdüze bir güç dağılımı elde edilmesi sağlanır. Reaktör ömrünün başlangıcında besleyici yakıt demeti, örtü yakıt demetleri ile aynı hizada bulunduğu noktadan yaklaşık 24.5 inç aşağıda bulunur ve reaktör işletimde olduğu süre boyunca yukarı doğru hareket ederek fisil izotop dağılımı değişen reaktörün kritikliğini sağlar. Besleyici yakıt demetlerinin tam güç çalışan korda farklı senaryolar için kritiklik konumları şekil 2.18’de verilmiştir.



Şekil 2.18 Farklı kor durumları için kritikliği sağlayan besleyici yakıt demeti konumlarının işletme süresine göre değişimi [13].

ThO₂ basamaklarının Şekil 2.17’de görüldüğü üzere farklı uzunluklarda yapılmasının bir etkisi de besleyici yakıt demetlerinin reaktör ömrünün başlarındaki reaktivite değerlerini (“reactivity worth”) arttırmaktır.

Reaktör işletildiği süre boyunca zamanla ThO₂ basamaklarındaki Th-232 izotoplarının çoğu U-233 izotoplarına dönüşecektir, bu sebeple besleyici yakıt demetinin reaktivite değeri işletme ömrü boyunca azalış gösterir. Besleyici yakıt demetlerinin reaktivite değerleri, reaktörün gerçek zamanlı işletimi için şekil 2.19’da verilmiştir.



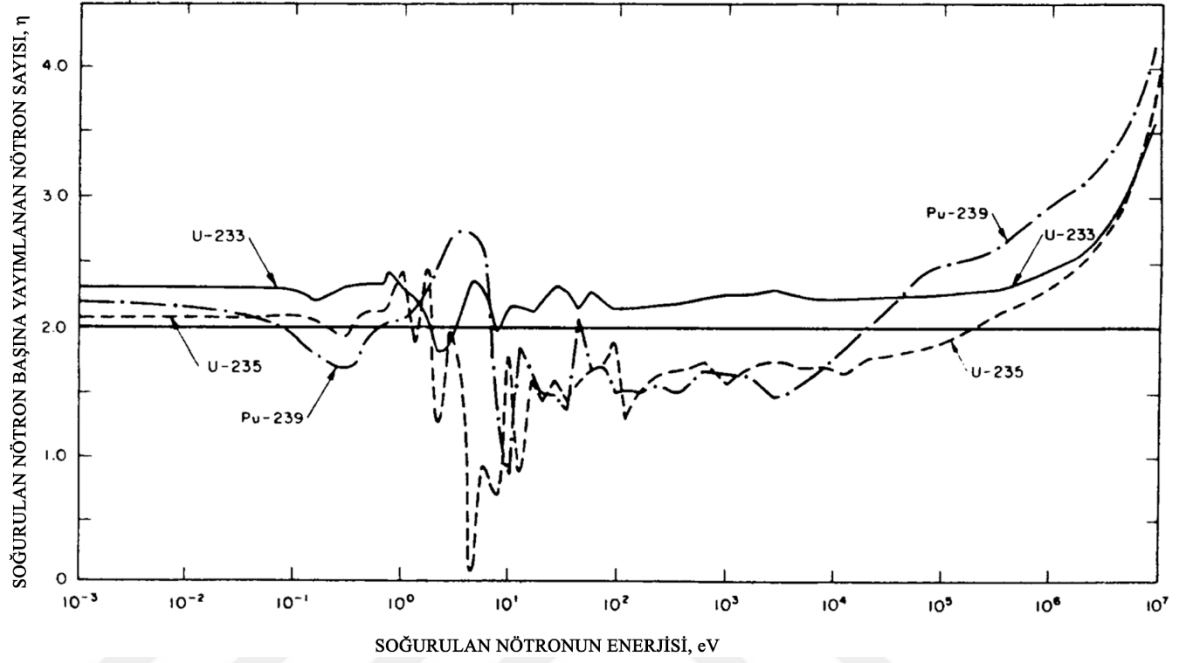
Şekil 2.19 Besleyici yakıt demeti reaktivite değerinin tam güç ile işletim süresince değişimi [14].

2.4. Üretkenlik Parametreleri

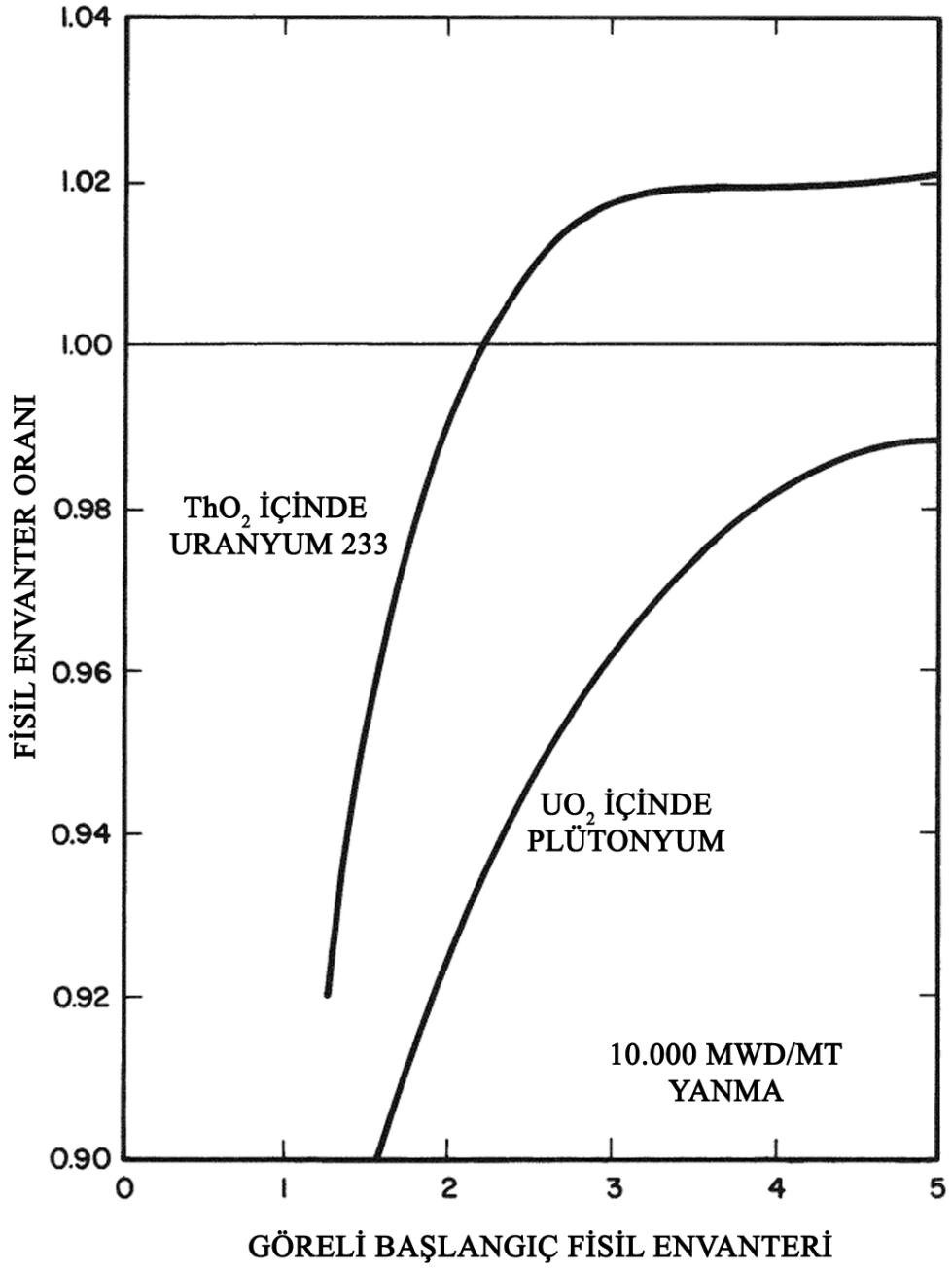
Fisil izotop tarafından soğurulan nötron başına fisyon sonrası açığa çıkan ortalama nötron sayısının oranı “ η ” sembolü ile gösterilir. Reaktördeki zincir fisyon tepkimelerinin devam etmesi için fisyon sonrası açığa çıkan nötronlardan bir tanesinin fisil izotoplar tarafından soğurulması gerekir. Üretkenliğin devamı için ise bir diğer nötronun doğurgan izotoplar tarafından soğurulması gerekmektedir. Fisyon sonrası açığa çıkan nötronlardan bir kısmının reaktörden kaçacağı veya yakıt dışındaki malzemeler tarafından soğurulacağı da hesaba katıldığında bir reaktörde dönüşüm oranının 1’den büyük olması için η değerinin de 2’den büyük olması gerektiği görülür.

U-235 ve Plütonyum 239 (Pu-239) izotopları ile termal enerjideki nötronların fisyon tepkimelerinde η değeri 2’nin çok az üzerindedir. Epitermal enerji bölgesindeki nötronlar için ise bu iki izotopun η değerleri 2’nin altına düşmektedir. Yani U-235 ve Pu-239 izotoplarının termal ve epitermal enerji bölgelerindeki nötronlarla üretkenlik için kullanılması mümkün değildir. Şekil 2.20’de de görülebileceği üzere U-233 izotopu için η değeri termal enerji bölgesinde 2.3 civarı bir değere sahipken epitermal enerji bölgesinde η daha aşağı değerlere düşmektedir. Öte yandan bir reaktör için nötron enerji spektrumunun epitermal bölgeye doğru kayması parazitik soğurulma tepkimelerinde

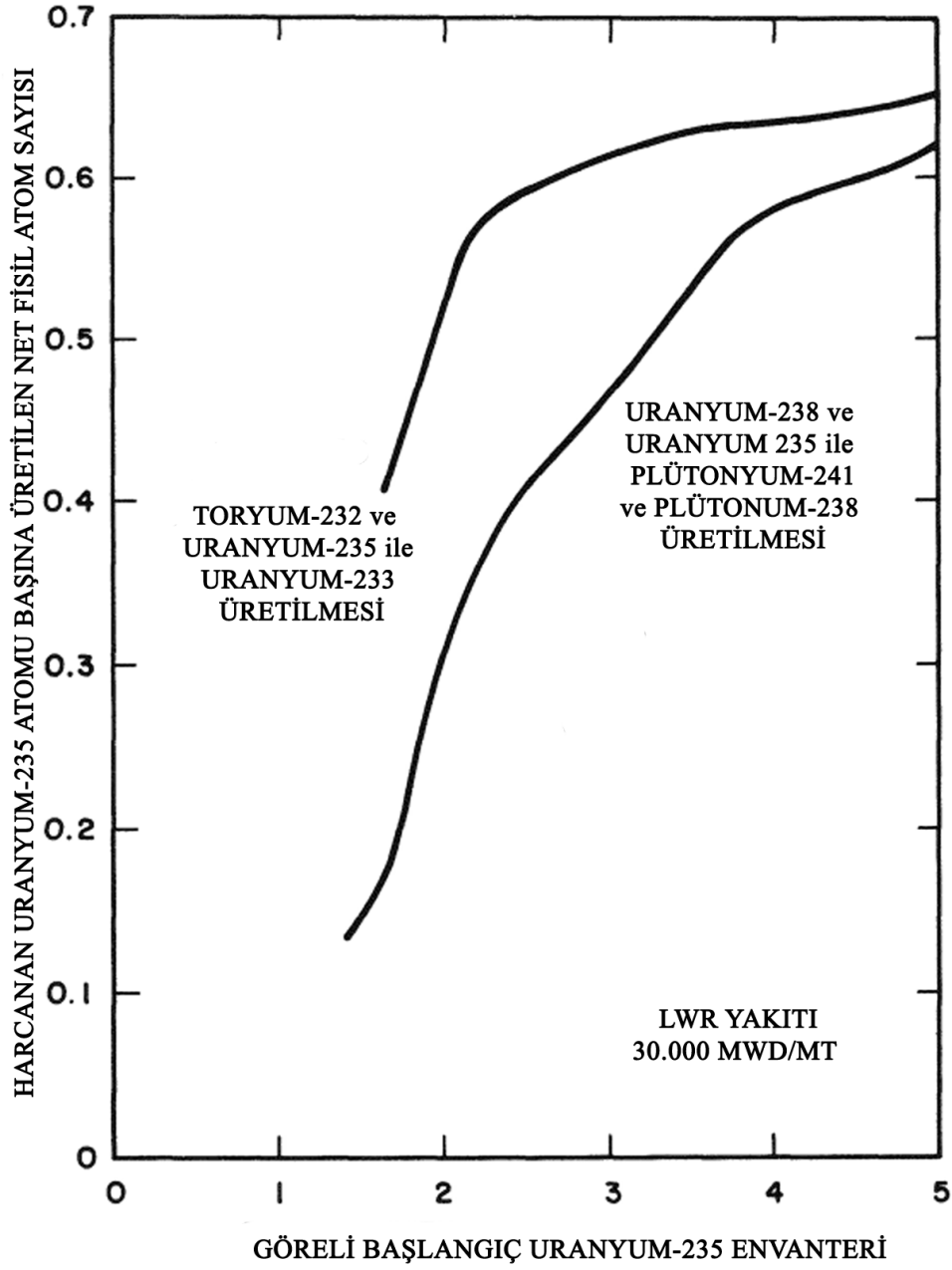
azalmaya ve eğer reaktör toryum içeriyorsa toryum elementinin fisyon ve (n,2n) tepkimelerinin artmasına yol açacak, böylece η için efektif değerin ve nötron ekonomisinin artmasına sebep olacaktır [13].



Şekil 2.20 U-233, U-235 ve Pu-239 için η değerlerinin soğurulan nötron enerjisine karşı değişimleri [7].



Şekil 2.21 Parazitik nötron kayıplarının olmadığı varsayılan durum için fisil envanter oranının başlangıç fisil envanter değerlerine göre değişimi [7].



Şekil 2.22 Parazitik nötron kayıplarının olmadığı varsayılan durum için fisil izotop üretim verimliliğinin başlangıç fisil envanter değerlerine göre değişimi [7].

Üretkenlik performansının belirlenmesinde yararlanılan iki önemli parametre “dönüşüm oranı” (DO) ve “fisil envanter oranı” (FEO) değerleridir.

Dönüşüm oranı veya üretkenlik oranı, bir “t” anındaki üretilen fisil izotop sayısının harcanan fisil izotop sayısına oranı olarak tanımlanır. Yani dönüşüm oranı, doğurgan izotoplar ve fisil izotopların nötron yakalama hızlarının oranı olarak da tanımlanabilir.

Doğurgan izotopların nötron yakalamasıyla oluşan fisil izotoplar da zaman içinde harcanacaklardır. Bu sebeple nükleer reaktörlerde üretkenlik söz konusu olması için yani reaktörde hem enerji üretilip hem de reaktörde tüketilenden daha fazla fisil izotop üretilmesi için bu dönüşüm oranının 1 değerinin üstünde olması gerekmektedir.

Bir “t” anındaki Dönüşüm Oranı aşağıdaki şekilde yazılabilir;

$$DO(t) = \frac{t \text{ anındaki fisil izotopların üretim hızı}}{t \text{ anındaki fisil izotopların harcanma hızı}} \quad (2.1)$$

Tezdeki hesaplamalarda referans olarak kullanılan çalışmalarda dönüşüm faktörü Şekil 2.2.’deki gibi hesaplanmıştır [15]:

$$DO(t) = \frac{C_{Th-233}(t) + C_{U-232}(t) + C_{U-234}(t) + A_{Pa-231}(t) \left(\frac{F_{U-232}(t)}{A_{U-232}(t)} \right) - A_{Pa-233}(t)}{F_{U-232}(t) + A_{U-233}(t) + A_{U-235}(t)} \quad (2.2)$$

Burada; C(t) t anındaki yakalama hızını, A(t) t anındaki soğurma hızını, F(t) ise t anındaki fisyon hızını gösterir. Referans olarak kullanılan çalışmada yapılan bu hesaplama göre Th-232, U-232, Pa-231 ve U-234 izotopları doğurgan kabul edilmiştir [14]. Bu dönüşüm oranı hesabı Serpent kodunda yapılan dönüşüm oranı hesabından farklılık gösterir.

Fisil envanter oranı, reaktörün bir “t” anında sahip olduğu fisil atom miktarının reaktör işletmeye başladığı andaki başlangıç fisil atom miktarına oranı olarak tanımlanabilir.

$$FEO(t) = \frac{t \text{ anındaki fisil atom miktarı}}{\text{Başlangıç Fisil Atom Miktarı}} \quad (2.3)$$

Pa-233 izotopu yaklaşık 27 günlük yarılanma ömrü ile beta ışıınımı yaparak U-233 izotopuna dönüşmektedir. LWBR ile ilgili yapılan geçmiş çalışmalarda reaktör kapanması sonrası Pa-233 izotoplarının hepsinin U-233 izotopuna dönüşeceği

varsayıldığından FEO hesaplanırken Pa-233 izotop miktarı da fisil atomlar arasında kabul edilmiştir [9][15].

$$FEO(t) = \frac{\gamma N_{U-232}(t) + N_{U-233}(t) + N_{U-235}(t) + N_{Pa-233}(t)}{N_{U-232}(0) + N_{U-233}(0) + N_{U-235}(0)} \quad (2.4)$$

Bu eşitlikte;

N = Nüklid konsantrasyonu

γ = ortalama fisyon – soğurma oranı

Bu tez çalışmasında da FEO, her yanma adımı için Eşitlik 2.4 kullanılarak hesaplanmıştır. DO için Serpent kodunun çıktı dosyalarında bulunan değerler Eşitlik 2.2 ile farklılık gösterse de bu fark ihmal edilebilir seviyelerde olduğu gözlemlenmiştir. Bu sebeple tezin 5. başlığındaki hesaplamalarda Serpent kodunun çıktı dosyalarında bulunan değerler kullanılmıştır.

3. NÖTRONİK VE YANMA HESAPLAMALARI

3.1. Nötron Transport Hesaplamaları Kodları

3.1.1. Monte Carlo Transport Hesaplamaları

Nötron transport hesaplamaları için Serpent ve diğer Monte Carlo kodları; etkileşimlerin oluşması, parçacıkların benzeşim yapılan geometri modelindeki hareketleri ve istenilen bir benzeşim sonuçlarının takibi için hesaplamalar yapan kısımlardan oluşur [16]. Benzeşimde parçacıkların hareketlerinin izlenmesi için kullanılan rutinlerde parçacıkların birbirini takip eden etkileşimleri arasında geometri içinde izleyecekleri yolların benzeşimi rastgele bir şekilde yapılır.

Nötron etkileşimleri için kullanılan makroskobik tesir kesiti yani Σ_{top} ifadesi, nötronun ortamda ilerlediği mesafe başına etkileşimde bulunma ihtimalini gösterir. Bu durumda makroskobik tesir kesitinin sabit olduğu sonsuz ve homojen bir ortam için nötronun dl mesafesinde hareket ederken etkileşimde bulunma ihtimali:

$$dP = \Sigma_{top} dl \quad (3.1)$$

İfadesi ile gösterilebilir. Eğer $P_0(l)$ nötronun bir l mesafesini etkileşimde bulunmadan katetmesi ihtimaline eşitse nötronun dl kadar daha mesafe katettiğinde $P_0(l)$ fonksiyonundaki azalma nötronun bu aralıkta etkileşimde bulunma ihtimaline eşit olacaktır. Böylece

$$dP_0 = -P_0(l)dP \quad (3.2)$$

Yani:

$$dP_0 = -P_0(l)\Sigma_{top}dl \quad (3.3)$$

İfadesi P_0 için çözülerek etkileşim gerçekleşmemesi ihtimali için;

$$P_0(l) = e^{-l\Sigma_{top}} \quad (3.4)$$

Eşitliği yazılabilir. Nötronun bir l mesafesini etkileşimde bulunmadan katedip dl mesafesinde etkileşimde bulunması ihtimali böylece:

$$P_0(l) = P_0(l) \Sigma_{top} dl = \Sigma_{top} e^{-l \Sigma_{top}} dl \quad (3.5)$$

İfadesine eşit olur. Bu durum için Olasılık Yoğunluk Fonksiyonu (OYF):

$$p(l) = \Sigma_{top} e^{-l \Sigma_{top}} \quad (3.6)$$

$$dp(l) = \Sigma_{top} dl \quad (3.7)$$

Şeklinde yazılabilir. Buradan görüldüğü gibi parçacıkların birbirini takip eden etkileşimleri arasında izleyecekleri yol uzunlukları sonsuz homojen bir ortam için üstel bir dağılım gösterir. Σ_{top} ifadesi parçacığın hareket ettiği yol boyunca sabit bir değere sahip olduğu için bu OYF kullanılarak parçacığın kat edeceği yol uzunluğu aşağıdaki ifade ile yazılabilir.

$$l = -\frac{\log \xi}{\Sigma_{top}} \quad (3.8)$$

ξ ifadesi $[0,1]$ aralığında tekdüze dağılımlı rastgele bir değişkendir.

Σ_{top} ifadesinin parçacığın izlediği yol boyunca sabit kalması için parçacığın farklı malzeme içeren yüzeylere veya aynı malzeme içinde farklı makroskobik tesir kesitine sahip bölgelere geçmemesi gerekir. Benzeşimi yapılan eğer farklı malzemelerden oluşan heterojen bir geometriye her bölgede farklı bir Σ_{top} değeri olacağı için geometrideki serbest yol uzunluğu için OYF parçalı sürekli bir fonksiyon olacaktır. Bu durumda parçacığın katedeceği yolun hesaplanmasında uygulanabilecek iki yöntem vardır. Bunlar: “Yüzey İzleme” (Surface Tracking veya Ray-Tracking) yöntemi ve “Delta İzleme” (veya “Delta Saçılımı” (Delta Tracking veya Delta Scattering) yöntemleridir [17].

3.1.1.1.Yüzey İzleme Yöntemi

Yüzey izleme yöntemi en yaygın kullanılan parçacık izleme algoritmasıdır. Nötronun ortamda kat ettiği mesafe için yukarıda açıklanan örnekleme formüllerinden de anlaşılacağı üzere nötronun hareketi sırasında etkileşim noktasına kadar devam edeceği yol uzunluğu, nötronun ortamda kat ettiği mesafeden bağımsızdır. Örneğin bir nötron bir malzeme boyunca etkileşimde bulunmadan ilerleyip sonraki malzeme bölgesine geçtiyse bu yeni malzeme bölgesinde kat edeceği mesafe nötronun bu noktaya kadar kat ettiği mesafeden bağımsızdır. Yani nötronun yeni bölgeye geçtiği noktayı yeni bir başlangıç noktası olarak kabul edip işlemleri bu yeni bölgeye göre tekrarlamak mümkündür. Parçacık farklı makroskobik tesir kesitlerine sabit bölgelerin sınırlarını geçtikçe yeni bölgenin tesir kesiti değerleri kullanılarak yeni bir yol hesaplanır. Parçacığın bölge sınırlarından geçişinin belirlenmesi için de en yakın sınırlara olan mesafelerin hesaplanması gerekmektedir. Yüzey izleme yönteminin böylece iki en büyük dezavantajı vardır; parçacıkların sınırdan geçtiği her durumda yeni mesafelerinin hesaplanması için parçacık durdurulur ve parçacıkların hareketleri hesaplanırken en yakın yüzeylere olan mesafelerin de hesaplanması gerekmektedir [18]. Parçacıkların sınır geçişi durumlarında durdurulmalarının gerekmesi, benzeşim geometrisinin parçacıkların ortalama serbest yol değerlerinden daha küçük olduğu benzeşimlerde hesaplama gücü gereksinimini arttıracaktır. En yakın yüzeylere olan mesafelerin hesaplanmasının gerekmesi de birçok yüzey barındıran geometrilerin olduğu benzeşimlerde hesaplama gücü gereksinimini arttıracaktır.

Algoritma 1: Yüzey İzleme Algoritması [19]

- 1: Döngüyü başlat
- 2: Parçacığın bulunduğu konumdaki bölge için toplam makroskobik tesir kesiti değerini bul (Σ_{top})
- 3: Parçacığın bulunduğu konumdan en yakın yüzeye olan mesafeyi hesapla (*En yakın yüzey mesafesi*, δ)
- 4: Parçacığın bulunduğu konumdan etkileşimde bulunacağı konuma olan mesafeyi hesapla ($l = -\log \xi / \Sigma_{top}$)
- 5: Eğer $l < \delta$ ise parçacığı etkileşim konumuna taşı ve döngüyü sonlandır
Eğer $l > \delta$ ise parçacığı yüzey sınırına taşı ve döngüyü devam ettir

3.1.1.2.Delta İzleme Yöntemi

Önceki başlıkta bahsedildiği üzere yüzey izleme yöntemi, farklı makroskobik tesir kesitlerine sahip bölgelerden oluşan heterojen geometriler için benzeşim yapılırken hesaplama gücü gereksinimini arttıran sınır geçiş hesaplama rutinlerine sahiptir. Delta izleme yönteminde geometrideki farklı makroskobik tesir kesitleri bir üst sınır tesir kesiti (“Majorant cross section”) tanımlanarak homojenleştirilir [17][20]. Üst sınır tesir kesitinin elde edilmesi, hayali bir saçılma olayının tanımlanması ile mümkündür. Bahsedilen bu hayali saçılma olayı, parçacığın ne enerjisini ne de doğrultusunu değiştirmeyen bir saçılma olayıdır. Parçacığın enerjisi ve doğrultusu hayali saçılma olayında korunduğu için parçacık izlediği yol boyunca bu hayali saçılma olayı ne kadar gerçekleşirse gerçekleşsin benzeşimde istatistiksel olarak bir sapma olmayacaktır.

Benzeşimde bulunan N farklı bölge için toplam fiziksel makroskobik tesir kesitleri $\Sigma_{top,1}$, $\Sigma_{top,2}$, ... $\Sigma_{top,N}$ olsun. Bu N farklı bölge için hayali saçılma olayının gerçekleşme ihtimali Σ_{san} sanal tesir kesiti ile tanımlanırsa hesaplamalarda kullanılacak toplam tesir kesitleri her bir bölge için:

$$\begin{aligned}\Sigma'_{top,1}(E) &= \Sigma_{top,1}(E) + \Sigma_{san,1}(E) \\ \Sigma'_{top,2}(E) &= \Sigma_{top,2}(E) + \Sigma_{san,2}(E) \\ &\cdot \\ &\cdot \\ &\cdot \\ \Sigma'_{top,N}(E) &= \Sigma_{top,N}(E) + \Sigma_{san,N}(E)\end{aligned}\tag{3.10}$$

Şeklinde yazılabilir. Sanal tesir kesitlerini istediğimiz değerlerde atanabileceği için benzeşimdeki geometrinin tamamında geçerli olacak bir Üst Sınır tesir kesiti tanımlanabilir. Geometrideki tüm bölgeler arasında en yüksek değerdeki toplam tesir kesiti $\Sigma_{top,maks}$ ifadesi ile Üst Sınır tesir kesiti de $\Sigma_{üst}$ ifadesi ile gösterilecek olursa $\Sigma_{üst} \geq \Sigma_{top,maks}$ olmak üzere:

$$\Sigma_{üst}(E) = \Sigma'_{top,1}(E) = \Sigma'_{top,2}(E) = \dots = \Sigma'_{top,N}(E) \quad (3.11)$$

Olacak şekilde $\Sigma_{üst}$ tesir kesiti tanımlanarak geometri homojen hale getirilebilir. Böylece parçacığın hareketini parçacık farklı bölgelere geçtiği durumda durdurmak gerekmeyecektir. Bunun yerine yüzey izleme yönteminden farklı ek bir adım olarak geri çevirme örnekleme (“rejection sampling”) yer alacaktır. P_N , N’inci bölgede etkileşimde bulunan parçacık için etkileşim noktasının kabul edilme olasılığı ise bu olasılık:

$$P_N(E) = \frac{\Sigma_{top,N}(E)}{\Sigma_{üst}(E)} \quad (3.12)$$

İfadesine eşit olacaktır. ξ ifadesi $[0,1]$ aralığında tekdüze dağılımlı rastgele bir değişken olmak üzere ξ değişkeni için:

$$\xi < \frac{\Sigma_{top,N}(E)}{\Sigma_{üst}(E)} \quad (3.13)$$

İfadesi sağlanamazsa etkileşim noktası geri çevrilir ve buradaki etkileşim sanal bir etkileşim olarak kabul edilir, enerjisi ve yönü aynı kalan parçacığın kat edeceği mesafe tekrar hesaplanır.

Delta izleme yöntemi, yüzey izleme yöntemine kıyasla heterojen geometrili veya ortalama serbest yol değerlerinin geometrinin boyutlarına kıyasla daha büyük olduğu yani parçacıkların hareketleri sırasında yüksek olasılıkla bölge sınırlarından geçtikleri benzeşimlerde yüzey izleme yöntemine göre hesaplama gücü gereksinimi açısından avantajlıdır. Fakat delta izleme yönteminin de bazı dezavantajları vardır [18][21]. Örneğin geometrilerinin bütününe kıyasla küçük hacimlerde yoğunlaşmış fakat toplam tesir kesiti değerleri yüksek olan malzemeler içeren benzeşimlerde geri çevirme örneklemesinin döngüleri verimsizleşir. Hacimce küçük olan bu malzemelerin yüksek toplam tesir kesiti değerleri yüzünden üst sınır tesir kesiti değeri de parçacığın geçeceği bölgelerin toplam tesir kesiti değerlerine kıyasla büyük bir değere sahip olacaktır. Bu

durumda benzeşinde geri çevirme örnekleme yapılırken etkileşimlerin çoğu yapay etkileşim olacak ve bu yöntemin performansı düşecektir.

Bir diğer dezavantaj ise bu yöntemde parçacık akısı için yol uzunluğu kestirici (“track length estimator”) yerine daha az verimli etkileşim kestiricisinin (“collision estimator”) kullanılmasının gerekmesidir.

Algoritma 2: Delta İzleme Algoritması [19]
1: Üst Sınır tesir kesitini hesapla ($\Sigma_{üst}$) 2: Döngüyü başlat 3: Parçacığın bulunduğu konumdan etkileşimde bulunacağı konuma olan mesafeyi hesapla $(l = -\log \xi / \Sigma_{üst})$ 4: Parçacığın etkileşimde bulunacağı konumdaki toplam fiziksel tesir kesiti değerini bul ($\Sigma_{top,N}$) 5: Etkileşim noktası için geri çevirme örneklemini gerçekleştir. Etkileşim noktası kabul edilirse döngüyü sonlandır. $(\xi < \Sigma_{top,N}(E) / \Sigma_{üst}(E))$ 5: Etkileşim noktası geri çevrilirse döngüyü devam ettir $(\xi > \Sigma_{top,N}(E) / \Sigma_{üst}(E))$

3.1.2. Serpent Kodu Transport Algoritması

Serpent kodunda varsayılan Monte Carlo transport yöntemi delta izleme yönteminin değiştirilmiş bir halidir. Daha önce bahsedildiği gibi delta izleme yönteminin dezavantajlarından biri hacimce küçük ama toplam makroskobik tesir kesiti büyük olan malzemelerin olduğu geometrilerde verimsizleşmektedir. Bu sorunu aşmak için Serpent’in delta izleme algoritmasında iki farklı üst sınır tesir kesiti tanımlanmaktadır. Bunlardan ilki yüksek değere sahip olan ve normal delta izleme yöntemindeki ile aynı olan üst sınır tesir kesiti $\Sigma_{üst,1}(E)$ ’dir. Bu tesir kesiti tüm bölgeler arasından maksimum toplam makroskobik tesir kesitine sahip bölgenin tesir kesiti değerine sahiptir. İkinci olan düşük üst sınır tesir kesiti $\Sigma_{üst,2}$ ise ağır soğurucu olarak tanımlanan izotopların toplam tesir kesitleri ihmal edilerek hesaplanır. Daha sonra bu iki üst sınır tesir kesitinin değerleri karşılaştırılır. Eğer iki tesir kesiti değeri arasında kayda değer bir fark varsa parçacığın etkileşime kadar kat edeceği yol uzunluğu ilk olarak düşük üst sınır tesir kesiti $\Sigma_{üst,2}$

kullanılarak hesaplanır. Bulunan bu mesafe, yüksek soğurucu bölgelere olan en kısa mesafeler ile karşılaştırılır. Eğer parçacık yüksek soğurucu bölgenin sınırını geçmemiş ise parçacığın örneklenmiş etkileşme noktası kabul edilir. Öte yandan eğer parçacık yüksek soğurucu bölgenin sınırını geçmiş ise bu sefer parçacık yüksek soğurucu bölge sınırında durdurulup $\Sigma_{üst,1}$ kullanılarak parçacığın gideceği mesafe tekrar örneklenir.

Algoritma 3: Serpent Genişletilmiş Delta İzleme Algoritması [19]

1: Yüksek ve Düşük Üst Sınır tesir kesitlerini hesapla

$$(\Sigma_{üst,1}) \quad (\Sigma_{üst,2})$$

2: Döngüyü başlat, etkin üst sınır tesir kesitini düşük üst sınır tesir kesitine eşitle
 $(\Sigma_{üst,e} = \Sigma_{üst,2})$

3: Parçacığın bulunduğu konumdan en yakın yüksek soğurucu bölge yüzeyine olan mesafeyi hesapla

$$(En \text{ yakın yüzey mesafesi}, \delta)$$

4: Parçacığın bulunduğu konumdan etkileşimde bulunacağı konuma olan mesafeyi hesapla

$$(l = -\log \xi / \Sigma_{üst,e})$$

5: Eğer parçacığın kat edeceği mesafe en yakın yüksek soğurucu bölge yüzeyine olan mesafeden büyükse etkin üst sınır tesir kesitini yüksek üst sınır tesir kesitine eşitle, en yakın yüksek soğurucu bölge yüzeyine olan mesafeyi sonsuz kabul edip 4. adıma dön

$$(\Sigma_{üst,e} = \Sigma_{üst,2}), (\delta = \infty)$$

6: Eğer parçacığın kat edeceği mesafe en yakın yüksek soğurucu bölge yüzeyine olan mesafeden küçükse parçacığı bu noktaya taşı, bu konumdaki toplam fiziksel tesir kesiti değerini bul

$$(\Sigma_{top,N})$$

7: Etkileşim noktası için geri çevirme örneklemini gerçekleştir. Etkileşim noktası kabul edilirse döngüyü sonlandırıp etkileşimi örneklemdir.

$$(\xi < \Sigma_{top,N}(E) / \Sigma_{üst}(E))$$

8: Etkileşim noktası geri çevrilirse döngüyü devam ettir. Yüksek soğurucu bölge yüzeyine olan en kısa mesafeden parçacığın kat ettiği mesafeyi eksiltip 4. adıma dön

$$.(\xi > \Sigma_{top,N}(E) / \Sigma_{üst}(E)), (\delta = \delta - l)$$

Tez çalışmasında modellenen reaktörde kontrol çubukları gibi tesir kesitleri yüksek fakat hacimce az yer kaplayan geometri elemanları olmadığı için delta izleme yöntemi ile hesaplama yapmanın hesaplama süresi açısından sunduğu verimlilik de Serpent kodunun seçilmesinde etkili olan sebeplerden biri olmuştur.

3.2. Yanma Hesaplamaları

Nükleer reaktörlerdeki yakıt malzemelerinin kompozisyonları; nötron tepkimeleri ve radyoaktif bozunumlar sonucu değişime uğrar. Nötron tepkimelerinin oluşum hızları malzemelerin bulundukları ortamdaki nötron akısına bağlıdır. Yakıtlardaki fisil izotop kompozisyonu ise bu akı dağılımına etki eder. Fakat hem yakıt kompozisyonunu oluşturan çekirdeklerin yoğunluk dağılımının, hem de nötron akı dağılımının zamana bağlı çözümlerinin aynı anda hesaplanması olanaklı değildir. Bunun yerine, yanma kodlarında çekirdek yoğunlukları ve nötron yoğunluklarındaki değişiklikler birbirlerinin takip eden adımlarda çözülür. İlk adımda malzeme kompozisyonunun sabit olduğu varsayımı yapıp bu kompozisyon için nötron transport denklemleri çözülerek nötron yoğunluk dağılımları hesaplanır. Bu hesaplanan dağılım sayesinde nötron tepkimelerinin oluşma hızları hesaplanır. İkinci adımda ise tepkime hızlarının sabit kaldığı varsayımı yapıp yanma denklemleri çözülerek malzeme kompozisyonları hesaplanır.

Serpent Monte Carlo kodunda yanma hesaplamaları da önceki paragrafta bahsedilen adımlara dayanarak üç farklı farklı kısımdan oluşur. Bu kısımlar; nötron akısını malzeme kompozisyonuna göre ve sabit hal için (“steady state”) çözen kısım, belli bir zaman aralığında malzeme kompozisyonunda mikroskobik tepkimeler yüzünden meydana gelen değişimleri hesaplayan kısım ve son olarak bu iki hesaplama kısmını bağlaştırıp yanma hesaplamaları yapılmasını sağlayan kısımlardır [22].

Yanma hesaplarında nötron tepkimeleri ve radyoaktif bozunum sebebi ile malzemelerin kompozisyonlarında meydana gelen değişimler “Bateman Tükenme Denklemleri” çözülerek hesaplanır [23].

Bateman Tükenme Denklemleri “n” adet farklı çekirdek için aşağıdaki gibi yazılabilir:

.

$$\frac{dN_i}{dt} = \text{Üretim Hızı} - \text{Tüketim Hızı} - \text{Bozunma Hızı} \quad (3.14)$$

Veya;

$$\frac{dN_i}{dt} = \sum_j^n S_{j \rightarrow i} - \sigma_i N_i \phi - \lambda_i N_i \quad (3.15)$$

$$(i = 1, \dots, n \text{ için})$$

Burada:

$S_{j \rightarrow i}$, dönüşüm, bozunum veya fisyon sonucu i çekirdeğinin üretim hızı,

$\lambda_i N_i$, i çekirdeğinin bozunum hızı

$\sigma_i N_i \phi$, i çekirdeğinin dönüşüme uğrayarak azalma hızıdır.

Bateman tükenme denklemlerinin çözümü için ele alınan her çekirdek, tepkime ve kontrol hacmi için nötron akısı, tesir kesitleri ve tepkime hızlarının bilinmesi gerekmektedir. Serpent Monte Carlo kodunun nötron transport benzeşimi aşamasında mikroskobik tesir kesit değerlerinin akı ve hacim ağırlıklı ortalamaları Bateman tükenme denklemlerin çözüldüğü zaman aralığı boyunca sabit varsayılarak tek grup fisyon ve değişim (transmutasyon) tesir kesitleri hesaplanır [22]. Radyoaktif bozunum ve fisyon ürünü verileri ENDF veri dosyalarından okunur [24][25]. Serpent Bateman denklemlerinin çözümü için iki seçeneğe sahiptir. Bunlardan ilki, Dönüşüm Takip Analizi (Transmutation Trajectory Analysis) ikincisi ise Chebyshev Rasyonel Yaklaşım Metodu (Chebyshev Rational Approximation Method) metodudur [26].

Dönüşüm Takip Analizi, karmaşık bozunum ağlarındaki tüm olası bozunum yollarını doğrusal bir set bozunum zincirine çeviren bir yöntemdir. Nükleer reaktör gibi birçok dönüşümün gerçekleştiği uygulamalarda bu yöntem ile tüm bozunum zincirlerini yazmak karmaşıktır.

CRAM metodu, yanma kodlarında yaygın olarak kullanılan Dönüşüm Takip Analizi'nden daha hızlı ve daha yakın sonuçlar veren bir algoritma olduğu için Serpent kodunun yanma denklemi çözümlerinde kullanılan varsayılan algoritmasıdır [24][26].

3.2.2. Bağlantı Şemaları

Bahsedildiği gibi yanma hesabı sırasında Bateman yanma denklemleri; nötron tepkime hızları ve tesir kesitlerinin yanma adımı boyunca sabit kabul edilmesi yaklaşımı ile hesaplanır. Yanma hesaplamaları bu sebeple birkaç adıma bölünerek yapılmalıdır. Öte yandan bu hesaplamalar için gereken sürenin uzunluğu sebebi ile yanma adımı sürelerinin mümkün olduğunca uzun tutulması istenir. Her yanma adımı malzeme kompozisyonlarını değiştireceği için benzeşim hesaplamalarının transport ve yanma kısımlarının bağlantı şemaları benzeşimin isabetli sonuçlar verebilmesi için önemlidir.

Serpent kodunun nötron transport ile yanma kısımlarındaki çözümlerin bağlaşımı kısmı için sunduğu birkaç farklı algoritma seçeneği vardır [27]. Tezin bu kısmında bu algoritmalar özetlenecektir.

3.2.2.1 Sabit Ekstrapolasyon

“Açık Euler Metodu” veya “Euler Metodu” olarak da bilinen bu algoritma, Monte Carlo kodlarında kullanılan en basit yanma algoritmasıdır. Bu algoritmada yanma adımı başlangıcı (YAB) için nötron transport hesaplaması yapılarak elde edilen nötron akısı ve tesir kesitleri değerleri yanma adımı boyunca sabit kabul edilir ve malzeme konsantrasyonundaki değişimler bu sabit değerler kullanılarak hesaplanır. Nötron akısı ve tesir kesitleri sadece YAB için hesaplanmasından dolayı basamak fonksiyonu şeklinde davranış gösterir. Bu yüzden sonuçlar tahmin-düzeltilme (predictor-corrector) adımlarına sahip algoritmalarla kıyasla daha isabetsizdir. Öte yandan tahmin-düzeltilme algoritmalarında nötron transport hesaplamaları birkaç defa tekrarlandığı için aynı yanma adım uzunluğunda sabit ekstrapolasyon algoritması daha hızlıdır.

Algoritma 1: Sabit Ekstrapolasyon (CE) [27]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1} \leftarrow e^{A(\phi_i)(T_{i+1}-T_i)}x_i$$

4: Sonraki yanma adımına geçiş

3.2.2.2. Doğrusal Ekstrapolasyon

Bu algoritmada Yanma Adımı Sonu (YAS) için tekrar transport hesaplaması yapmak yerine nötron akısı ve tesir kesitleri değerleri YAB ve Önceki Yanma Adımı (ÖYA) akı ve tesir kesitleri değerleri kullanılarak doğrusal ekstrapolasyon ile hesaplanır ve yanma denklemleri çözülerek YAS kompozisyonları elde edilir. İlk yanma adımı için ÖYA değerleri olmadığı için ilk adımda sabit ekstrapolasyon algoritması uygulanır. YAS için transport çözümü yapılması gerekmediği için doğrusal ekstrapolasyon sabit ekstrapolasyon ile yaklaşık olarak aynı hesaplama süresine sahiptir.

Algoritma 2: Doğrusal Ekstrapolasyon (CE) [27]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: ÖYA ve YAB değerleri arasında doğrusal ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi} \leftarrow (\phi_{i-1}, \phi_i)$$

4: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1} \leftarrow e^{A(\bar{\phi})(T_{i+1}-T_i)}x_i$$

5: Sonraki yanma adımına geçiş

3.2.2.3. Sabit Ekstrapolasyon/Doğrusal İnterpolasyon

Yanma sırasında sürekli değişen tepkime hızlarının Sabit Ekstrapolasyon algoritmasına kıyasla daha iyi benzeşimini gerçekleştirmek için birçok Monte Carlo kodu, yanma adımlarının uzunluklarına göre koşullu kararlı olan tahmin-düzeltilme (predictor-corrector) algoritmaları kullanmaktadır [28]. Bu algoritmalarda taşınım denklemi her yanma adımı için birkaç kez çözülür. Bu tür algoritmaların en basit olanı, tahmin adımını için YAB akı ve tesir kesitlerinin hesaplanıp sabit kabul edildiği ve düzeltme adımı için doğrusal interpolasyon kullanılan CE/LI (Constant Extrapolation / Linear Interpolation) algoritmasıdır. Bu algoritma iki adımdan oluşur, ilk adımda YAB nötron akısı tesir kesitleri değerleri hesaplanır ve yanma adımı boyunca sabit kabul edilerek yanma denklemleri çözülür. Çözülen yanma denklemlerine göre Yanma Adımı Sonrası (YAS) malzeme kompozisyon değerleri tahmin edilir. YAS kompozisyon değerleri için nötron akısı ve tesir kesitleri değerleri hesaplanarak YAB ve YAS akı ve tesir kesitleri değerleri arasında doğrusal interpolasyon yapılır. Elde edilen ortalama değerler ile yanma denklemleri tekrar çözülerek sonraki yanma adımına taşınacak kompozisyon değerleri elde edilir.

CE/LI algoritması, CE algoritmasına göre daha fazla hesaplama süresi gerektirse de yanma aralığı yarıya indirilmiş CE hesaplamaları ile kıyaslandığında dahi çoğu zaman daha isabetli sonuçlar sunmaktadır.

Algoritma 3: Sabit Ekstrapolasyon/Doğrusal İnterpolasyon (CE/LI) [27]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\phi_i)(T_{i+1}-T_i)} x_i$$

4: YAS için tahmin edilen malzeme kompozisyonları kullanılarak nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_{i+1}^T \leftarrow \phi(x_{i+1}^T)$$

5: YAB ve tahmin edilen YAS değerleri arasında doğrusal ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi} \leftarrow (\phi_{i+1}^T + \phi_i)/2$$

6: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^D \leftarrow e^{A(\bar{\phi})(T_{i+1}-T_i)} x_i$$

7: Sonraki yanma adımına geçiş

3.2.2.4. Doğrusal Ekstrapolasyon ve Doğrusal İnterpolasyon

YAB ve ÖYA değerleri arasında doğrusal ekstrapolasyon yapılarak tahmin adımıyla kullanılacak nötron akı ve tesir kesitleri hesaplanır. Yanma denklemleri çözülerek tahmin adımı için YAS kompozisyonu belirlenir. Bu kompozisyon kullanılarak nötron akı ve tesir kesitleri değerleri tekrar hesaplanır ve bu değerler ile YAB değerleri arasında doğrusal interpolasyon yapılır. Yanma denklemleri bu değerler kullanılarak çözülür ve YAS kompozisyonu hesaplanır.

Algoritma 4: Doğrusal Ekstrapolasyon/Doğrusal İnterpolasyon (LE/LI) [27]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: ÖYA ve YAB değerleri arasında doğrusal ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi} \leftarrow (\phi_{i-1}, \phi_i)$$

4: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\bar{\phi})(T_{i+1}-T_i)} x_i$$

5: YAS için tahmin edilen malzeme kompozisyonları kullanılarak nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_{i+1}^T \leftarrow \phi(x_{i+1}^T)$$

6: YAB ve tahmin edilen YAS değerleri arasında doğrusal ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi} \leftarrow (\phi_{i+1}^T + \phi_i)/2$$

7: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^D \leftarrow e^{A(\bar{\phi})(T_{i+1}-T_i)} x_i$$

8: Sonraki yanma adımına geçiş

3.2.2.5. Doğrusal Ekstrapolasyon ve İkinci Derece İnterpolasyon

Bu algorithmada düzeltici adımdaki interpolasyon YAB ve YAS değerleri ile beraber ÖYA değerleri de kullanılarak gerçekleştirilir. İlk yanma adımı için ÖYA değerleri bulunmadığından dolayı bu adımda doğrusal interpolasyon kullanılır. Bu algoritma en doğru sonuçları veren algoritma olarak kabul edilse de çoğu durumda LE/LI ile aralarındaki fark ihmal edilebilecek düzeydedir [27].

Algoritma 5: Doğrusal Ekstrapolasyon/İkinci Derece İnterpolasyon (LE/QI) [27]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: ÖYA ve YAB değerleri arasında doğrusal ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi} \leftarrow (\phi_{i-1}, \phi_i)$$

4: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\bar{\phi})(T_{i+1}-T_i)} x_i$$

5: YAS için tahmin edilen malzeme kompozisyonları kullanılarak nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_{i+1}^T \leftarrow \phi(x_{i+1}^T)$$

6: ÖYA, YAB ve tahmin edilen YAS değerleri arasında İkinci Derece ekstrapolasyon yapılarak nötron akısı ve tesir kesitlerinin yeniden hesaplanması

$$\bar{\phi}^T \leftarrow (\phi_{i+1}^T, \phi_{i-1}, \phi_i)$$

7: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^D \leftarrow e^{A(\bar{\phi}^T)(T_{i+1}-T_i)} x_i$$

8: Sonraki yanma adımına geçiş

3.2.2.6. Stokastik Örtük Euler Yöntemi

Serpent kodunda, bu kısımdaki diğer beş başlıkta açıklanan ve “**set pcc**” komutu ile atanan yaygın tahmin edici-düzeltilici bağlantı şemaları dışında ayrıca Stokastik Örtük Euler (Stochastic Implicit Euler) bağlantı şeması bulunmaktadır. “**set sie**” komutu ile etkin hale getirilen bu bağlantı şeması diğer bağlantı şemalarının kararsız ve sapmalı sonuçlar verebildikleri durumlar için yanma aralıklarının uzunluğundan bağımsız olarak kararlı sonuçlar elde edilmesi amacı ile geliştirilmiştir [29].

Algoritma 6: Stokastik Örtük Euler Yöntemi (set sie) [29]

1: Döngüyü başlat

2: YAB için nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_i \leftarrow \phi(x_i)$$

3: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\phi_i)(T_{i+1}-T_i)} x_i$$

4: Belirlenen düzeltici adım sayısı için döngü

$$n \leftarrow 1, 2, 3, \dots, N$$

4.1: YAS için tahmin edilen malzeme kompozisyonları kullanılarak nötron akısı ve tesir kesitlerinin hesaplanması

$$\phi_{i+1}^T \leftarrow \phi(x_{i+1}^T)$$

4.2: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\phi_i)(T_{i+1}-T_i)} x_i$$

4.3: YAS için tahmin edilen tüm akı ve tesir kesitlerinin ortalamasının alınması

$$\bar{\phi}_{i+1}^n = \sum_{j=1}^n \frac{\phi_{i+1}^j}{n}$$

4.4: Yanma matrisinin oluşturulup Bateman yanma denklemlerinin çözülmesi, malzeme kompozisyonlarının hesaplanması

$$x_{i+1}^T \leftarrow e^{A(\bar{\phi}_{i+1}^n)(T_{i+1}-T_i)} x_i$$

5: Sonraki yanma adımına geçiş

Bu çalışmada yapılan benzeşim için farklı algoritmalar denenmiş fakat tezde sadece en tutarlı sonuçların elde edildiği Stokastik Örtük Euler algoritması kullanılan hesaplamalara yer verilmiştir.

4. GİRDİLERİN HAZIRLANMASI VE BENZEŞİM

Bu tez çalışmasında LWBR için tam güç işletme durumunda nötronik analiz yapmak için Serpent Monte Carlo kodu ile benzeşimler yapılmıştır. Bu benzeşim çalışmalarında farklı yanma algoritma ve adım uzunlukları denenerek referans olarak kullanılan üretkenlik parametrelerine yakın değerler elde edilmeye çalışılmıştır. Tez çalışmasındaki benzeşimler için girdilerin hazırlanması reaktörün yüksek heterojenlikteki yakıt demetleri, yakıt modüllerinin düzgün olmayan çokgen geometrileri ve yakıt demetleri ile reaktivite kontrolü gibi özgün özellikleri yüzünden girdi hazırlanması ve hesaplama gücü ihtiyacı artmış ayrıca ek adımlar gerekmiştir. Tezin bu kısmında reaktör modellemesinden bahsedilecektir.

4.1 Serpent Kodunda Geometrinin Oluşturulması

Serpent Monte Carlo kodunda benzeşimlerde kullanılan geometriler evrenlere, hücrelere ve yüzeylere dayalı bir modelleme ile oluşturulur. Bu modelleme katman veya seviyelerden oluşur, en dış veya en üst seviyede “0” evreni bulunur. Bu evren geometriyi oluşturan tüm alt hücreleri ve bu geometrinin sınırlarını kapsar. Bu sınırların dışı “**outside**” olarak adlandırılır. Hücreler homojen malzeme kompozisyonları ile başka evrenler ile veya “boşluk” (“**void**”) ile doldurulan ve sınırları basit yüzeyler ile tanımlanan geometri birimleridir. Modellenen bir reaktör için bir yakıt çubuğunu içeren evren veya evrenler yakıt kafesini oluşturan evrenin içine tekrarlanarak konabilir. Daha sonra bu evrenlerin de başka bir evrenin içine tekrarlanarak dizilmesi ile reaktör korunun tamamını için bir evren oluşturulabilir.

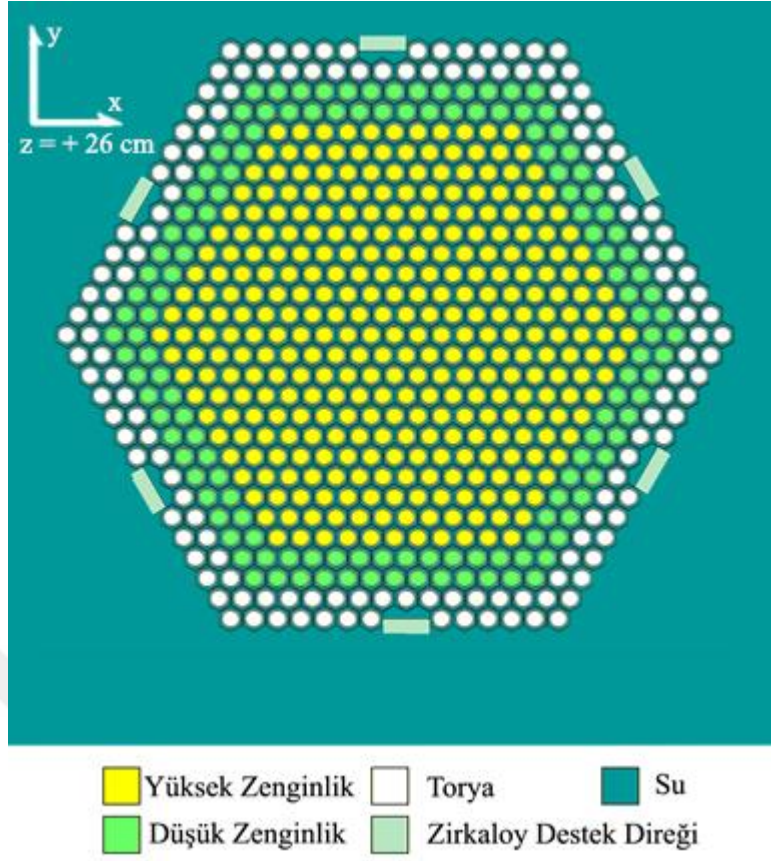
Serpent Monte Carlo kodu, diğer Monte Carlo kodlarından farklı olarak özellikle nükleer reaktörlerdeki yakıt kafeslerini ilgilendiren fiziksel hesaplamalar için oluşturulmuş bir Monte Carlo kodudur. Bu sebeple geometri, hücre ve evrenler oluşturulurken nükleer reaktörlerin modellenmelerini kolaylaştıran komutlara sahiptir. Bunlardan biri, kare veya altıgen dizilimde kafes geometrisi oluşturulmasında kullanılan “**lat**” komutudur. Bu komut, beraberinde yazılan evrenlerinin yakıt kafesi içine dizilimlerini içeren bir evren oluşturur, daha sonra oluşturulan bu evren de bir hücrenin içini doldurarak kullanılır. Benzer yüzeyler ile birbirinden ayrılan malzeme evrenlerini tanımlarken kullanılan “**nest**” komutu ve bu komuta benzer bir işleve sahip olan ve farklı malzemelerden oluşan iç içe halkasal bölgeleri tanımlamada kullanılan “**pin**” komutunu da geometri modellemesini kolaylaştıran Serpent komutlarındandır.

Yaygın birçok reaktörün Serpent ile tüm kor modellemesinde “**lat**” komutu büyük kolaylık sağlar. Örneğin önce “**pin**” komutu ile yakıt çubukları tanımlanıp daha sonra “**lat**” komutu ile bu yakıt çubuklarından bir yakıt kafesi oluşturulabilir, bu yakıt kafesleri de tekrar “**lat**” komutu ile dizilerek korun tamamını kapsayan bir model elde edilebilir.

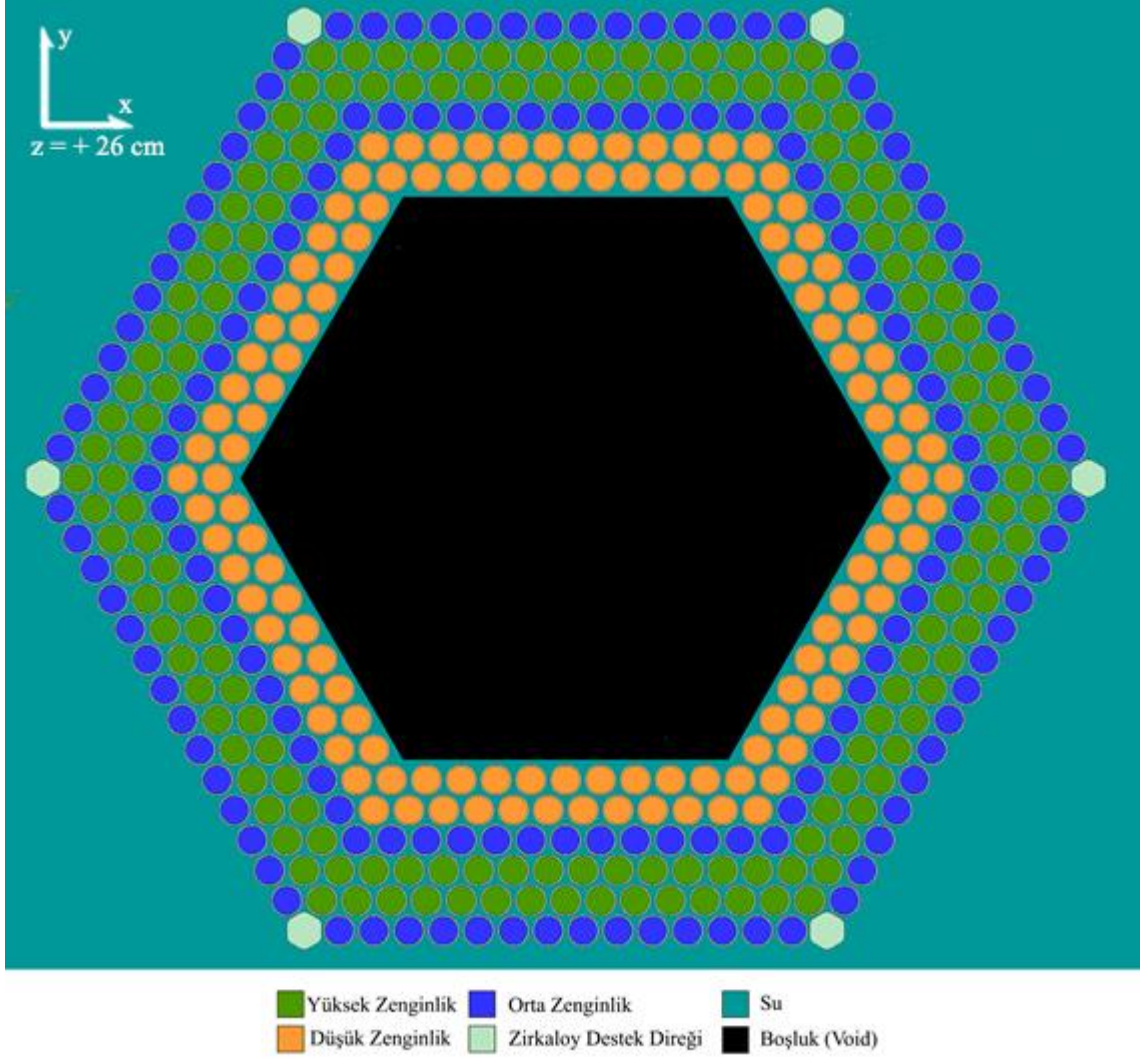
LWBR yakıt demetlerinde destek ızgaralarının soğurma tesir kesiti zirkonyum alaşımlarına kıyasla daha yüksek olan AM350 çeliğinden yapılmış olması sebebiyle bu destek ızgaralarının modellenmesi de geometriye eklenmiştir. Yakıt ızgaralarının altıgen kesit alanları ve yakıt çubuklarının halka kesit alanları sebebi ile “**pin**” yerine “**nest**” komutu kullanılmış ve yakıt çubukları eksenel yönde yığıtlardan oluşan evrenler olarak tanımlanmıştır. Daha sonra bu evrenler “**lat**” komutları kullanılarak besleyici, örtü veya yansıtıcı modülleri oluşturan evrenlerin tanımlanmasında kullanılmıştır.

Hareketli besleyici yakıt demetleri ile etraflarını saran örtü yakıt demetlerinin ayrı evrenler olarak tanımlanmaları gerekmiştir. Bunun için örtü yakıt demetlerinin iç kısımları “**void**” olarak tanımlanmıştır.

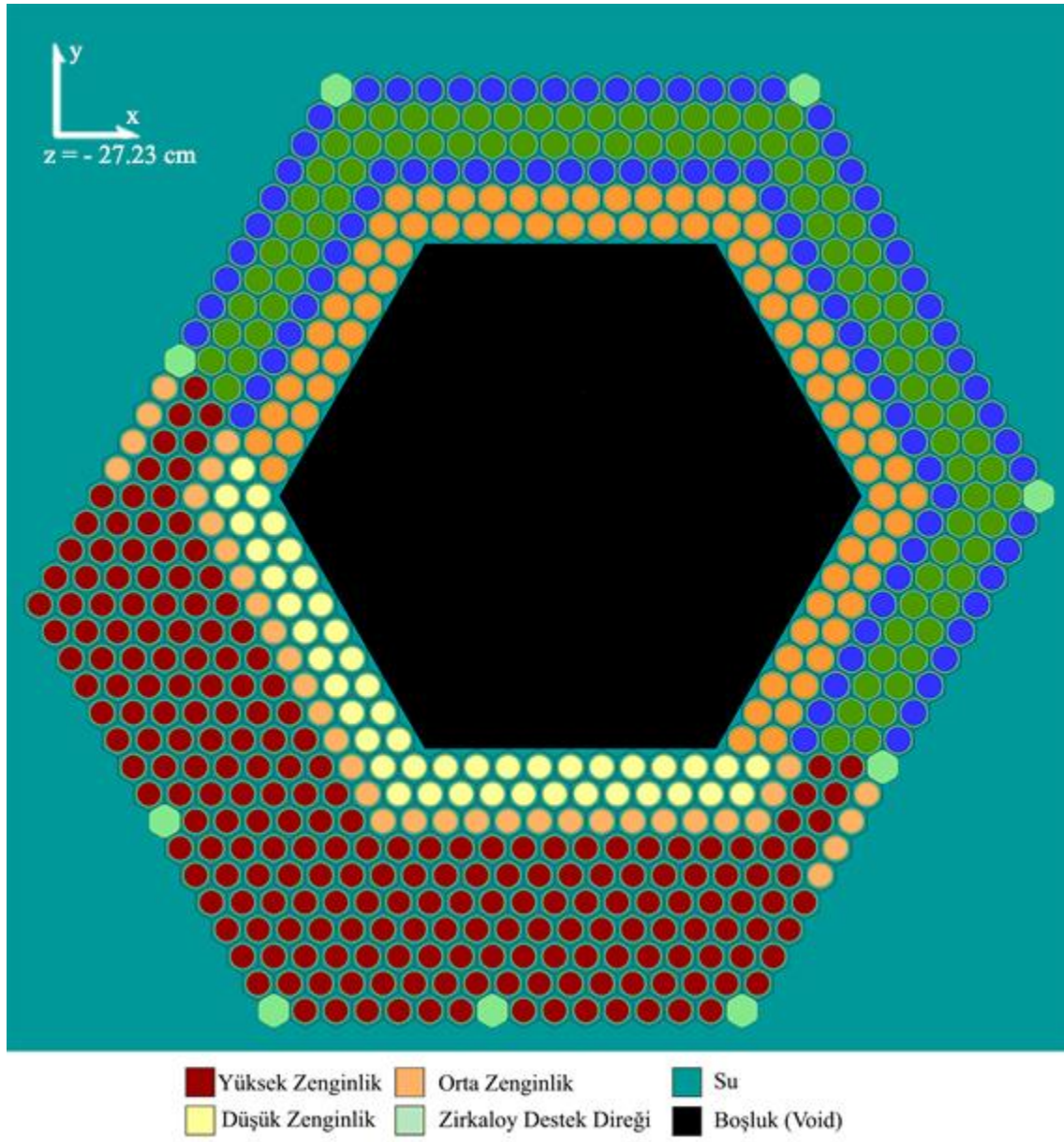
Bu tez çalışmasında modellenen 27 modülden sadece en merkezdeki 3 tanesi düzgün altıgen geometriye sahiptir. Bunun bir sonucu olarak “**lat**” komutları ile kor diziliminin elde edilmesi mümkün olmamıştır. Bu noktada bu düzgün olmayan çokgen geometriler için hücre tanımlamaları yapıp bu hücrelere tekabül eden modül evrenleri ile doldurulması gerekmiştir. Bunun için bir Bilgisayar Destekli Çizim (CAD) programı olan FreeCAD [30] ile reaktörün kesit alanı birebir ölçüde modellenip hücreleri oluşturacak düzlem tanımlamaları için düzlemlerin sınır koordinatları elde edilmiştir. Modüllerin iç kısımlarına besleyici yakıt demetlerini dolduracak şekilde sınırları belirlenmiş bir düzgün altıgen hücre, besleyici yakıt demetini dışında kalan kısımlara da örtü demetlerini dolduracak şekilde de modül hücreleri tanımlanmıştır.



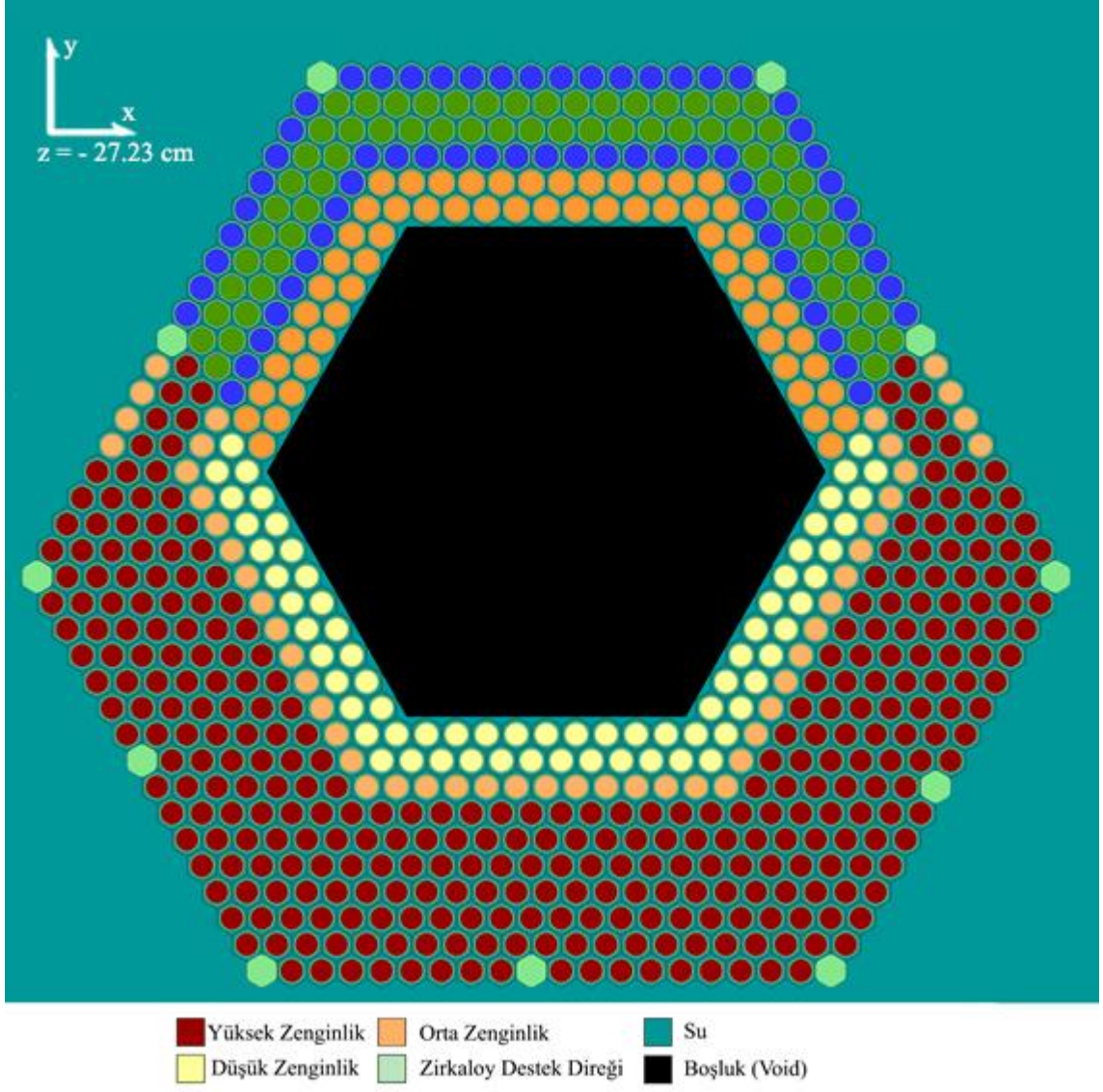
Şekil 4.1 Besleyici yakıt demeti Serpent modellemesi kesit alanı görünümü



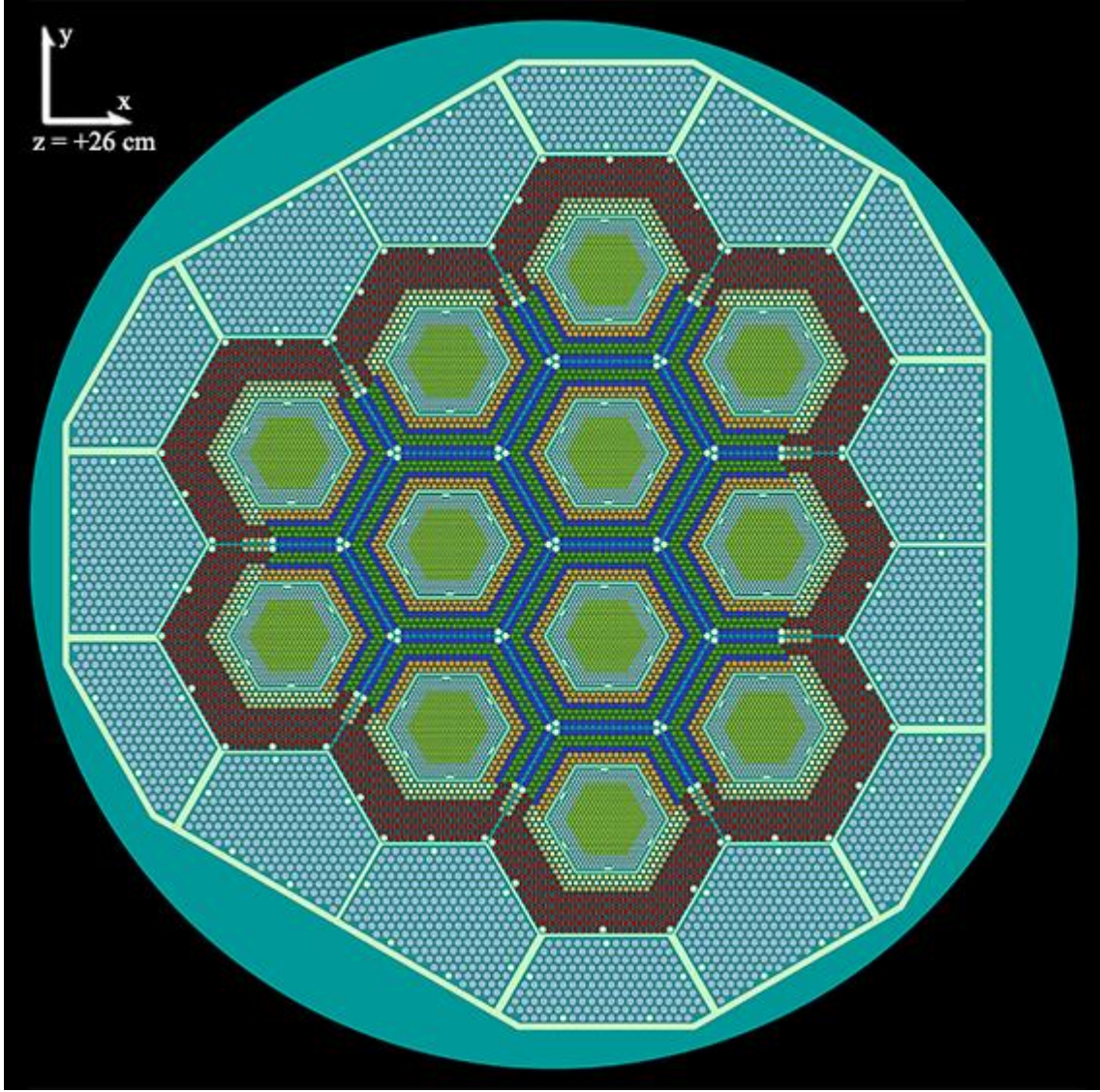
Şekil 4.2 Tip I yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü



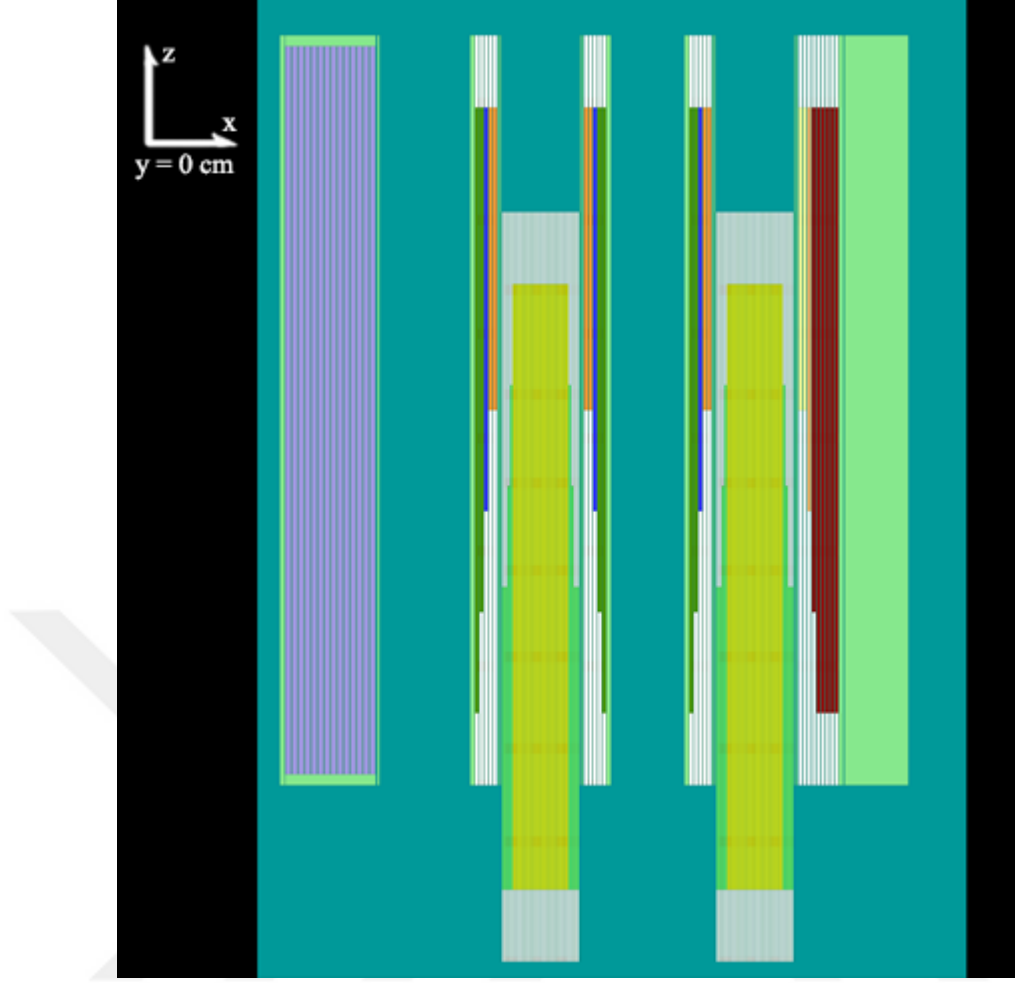
Şekil 4.3 Tip II yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü



Şekil 4.4 Tip III yakıt modülü örtü yakıt demeti için Serpent modellemesi kesit alanı görünümü



Şekil 4.5 LWBR koru Serpent Modellemesi.



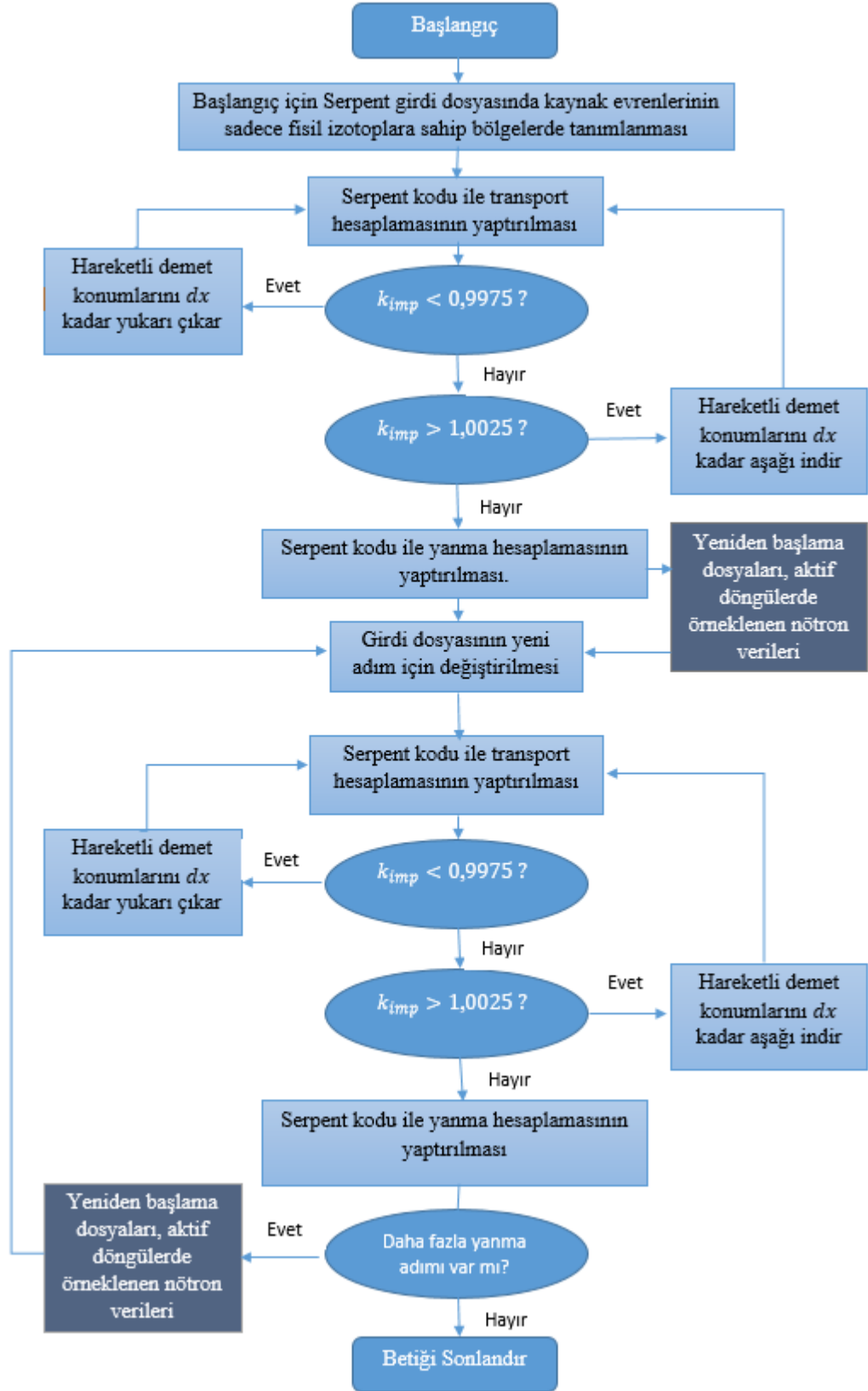
Şekil 4.6 Besleyici yakıt demetlerinin 24 inç aşağıda olduğu durum için LWBR koru Serpent modellemesi dikey kesit görünümü.

4.2 Python Kodu

Serpent kodu ile yanma hesaplamaları yapmak için “**dep**” komutu, beraberinde yanma adımı tipi ve yanma adımı değerleri girilerek kullanılır. Reaktivitenin yakıt demetlerinin hareket ettirilmesi ile kontrol edildiği LWBR’de reaktör geometrisi ve nötron akısı doğrudan ilişkilidir. Nötron akısının da malzeme kompozisyonuna bağlı olmasından dolayı her bir yanma hesaplanmasından sonra hareketli yakıt demetlerinin konumları reaktörü kritik yapacak şekilde tekrar hesaplanmalıdır. Serpent Monte Carlo kodunun bu tez çalışmasında kullanılan sürümünde bu kritiklik işlemi yapacak bir komut bulunmamaktadır. Bu sebeple LWBR reaktör modelinin kritikliğinin sağlanıp yanma işlemlerinin gerçekleştirilmesi için bir Python kodu yazılmıştır. Bu kod ilk olarak reaktör modelini statik kritiklik hesaplamasını Serpent’e çözdürerek nötron çoğaltma faktörünü

± 0.0025 'lık bir tolerans aralığı ile kritik tutan besleyici yakıt demeti pozisyonunu bulana kadar yineleme (iterasyon) gerçekleştirir. Daha sonra bulunan bu pozisyon için Serpent'i tekrar çalıştırıp yanma hesaplamalarını gerçekleştirir. Yanma hesaplaması sonunda malzeme kompozisyonlarındaki değişimler “.wrk” uzantılı Serpent çıktı dosyalarına kaydedilir. Bunun yanında her adımdaki aktif çevrimlerdeki örneklendirilen nötronların verileri de “.src” uzantılı Serpent dosyalarına kaydedilir. Python kodunun döngüsündeki bir adım böylece tamamlanmış olur. Sonraki döngüye geçmeden önce Serpent girdi dosyası kopyalanıp yeni kopyadaki bazı komut satırları değiştirilir:

- 1 Önceki yanma adımındaki “.wrk” uzantılı dosyayı yeniden başlatma dosyası girdisi olarak tanımlandığı komut → Yeni yanma adımı için girdi dosyası tekrar adlandırılır, dosyadan okunulacak yanma adımı (gün) değiştirilir.
- 2 Yeni yanma adımından sonra farklı isimde yeni “.wrk” uzantılı yeniden başlatma dosyası çıktısının tanımlandığı komut → Yeni yanma adımı için çıktı dosyası tekrar adlandırılır.
- 3 Pasif çevrimlerinde ilk tahmin için okunacak önceki yanma adımında kaydedilmiş “.src” uzantılı kaynak dosya girdisinin tanımlandığı komut → Yeni yanma adımı için girdi dosyası tekrar adlandırılır.
- 4 Aktif çevrimlerdeki nötronların verilerinin “.src” uzantılı kaynak dosyasına kaydedildiği komut → Yeni yanma adımı için çıktı dosyası tekrar adlandırılır.
- 5 Yanma toplam gün sayısının olduğu komut → Yanma aralığı kadar artırılır



Şekil 4.7. Python kodu algoritması.

4.3 Benzeřim

Bu kısımda, benzeřim hesaplamaları ile ilgili genel bilgiler ile bu hesaplamalar sırasında karřılařılan hatalar ve üretilen çözümler özetlenmiřtir. Benzeřim için Serpent Monte Carlo kodu hesaplamaları, Hacettepe Üniversitesi Nükleer Enerji Mühendislięi bünyesinde bulunan FORMULA Yüksek Başarımlı Bilgisayar Kümesi kullanılmıřtır. Farklı sunucularda paralel olarak Serpent kodunun kořulması mümkün olsa da bu çalıřma için denendięinde sebebi bilinmeyen hatalarla karřılařıldıęı için hesaplamalar tek sunucunun çekirdekleri arasında paylařtırılması ile sınırlandırılmıřtır. Bilgisayarın 64 çekirdekli ve 226 GB RAM'e sahip GP20 isimli sunucusu kullanılmıř olup 18.000 ETGS için yapılan benzeřim hesaplamalarından her biri ortalama 15-25 gün arası sürelerde tamamlanmıřtır.

Serpent kodunun bu çalıřma için kullanılan sürümünde normal çalıřma için bazı hata durumlarında hatanın kaynaęına dair geri bildirim verilmemektedir. Bu gibi durumlarda hatanın sebebinin belirlenmesi Serpent kodunun **“DEBUG”** modu aktif hale getirilerek yeniden derlenmesi ile mümkün olmaktadır fakat bu mod yapılan hesaplamaları daha da yavařlattıęı ve hesaplama gücü ihtiyacını arttırdıęı için kullanılmamıřtır. Bunun yerine deneme ve yanılma yöntemleri ile hataya sebep olan girdi kısımları düzeltilmiřtir. Bu kısımda bahsedilen bu hatalar kısaca özetlenmiřtir.

LWBR korunda bulunan destek ızgarası yapılarının bulunduęu yükseklikteki yakıt çubuklarının Serpent'de modellenmesi için **“nest”** komutu kullanılması gerekmektedir. Öte yandan destek ızgaraları tarafından çevrelenmeyen yakıt çubukları **“pin”** komutları ile de modellenebilmektedir. Çalıřmanın ilk evrelerinde yakıt çubuęu evrenleri, **“pin”** ve **“nest”** komutları kullanılarak oluřturulan evrenlerin üst üste dizilmesi ile elde edilmiř fakat bu řekilde yapılan benzeřimler kodun hata vermesi ve çalıřmayı durdurması ile sonuçlanmıřtır. Çözüm olarak geometri sadece **“nest”** evrenleri oluřturacak řekilde ve **“lat”** komutunun 9. Seçeneęi kullanılarak yeniden oluřturulmuřtur.

Örtü bölgelerinde bulunan ve zirkonyum alařımından oluřan destek direkleri de gerçek reaktördeki řekilleri ile modellenmiř fakat transport hesaplamalarında bilinmeyen bir sebepten dolayı benzeřimi yapılan nötronlar hareketleri sırasında destek direklerinin geometri sınırlarından geçmeleri durumunda Serpent kodu hata verip çalıřmayı durdurmuřtur. Çözüm olarak destek direęi geometrileri **“pin”** komutu kullanılarak eřdeęer hacimde silindir yapılar olarak tanımlanmıřtır.

Geometri modellemesinde reaktörün alt ve üst kısımlarında belli mesafelerden sonra modellemenin sınırlarını belirleyen “outside” bölgeleri atanmış fakat ilk yanma adımında bu bölgelere taşan hareketli yakıt demeti kısımları bir sonraki yanma adımında benzeşim yapılan aralığa çıkarıldığında Serpent kodu hata verip çalışmayı durdurmuştur. Bu duruma çözüm olarak reaktörün alt ve üst kısımlarında hareketli yakıt demetlerinin uzunlukları kadar su ile doldurulan bölgeler modellenip “outside” bölgeleri bu kısımların ötesinde tanımlanmıştır. Serpent kodunun kaynak dosyalarında değişiklik yapıp yeniden derlenmesi de alternatif bir çözüm olarak bulunmuş olsa da bu şekilde bir değişikliğin doğurabileceği farklı hatalar göz önünde bulundurularak bu yönetime başvurulmamıştır.

Serpent kodunda oluşturulan geometriler “**plot**” komutları kullanılarak farklı çözünürlükler ile “.png” uzantılı dosyalara çizdirilebilmektedir. Eğer çizdirilen geometrilerde birbirlerinin üzerine taşan bölgeler varsa kırmızı, hiçbir evrenin tanımlanmadığı bölgeler var ise de yeşil renkler ile işaretlenir ve Serpent kodunun koşulduğu terminalde çizdirilen geometrilerde hata olduğu uyarısı yazdırılır. Fakat bu uyarıların terminalde çıkması geometri çizdirilirken kullanılan çözünürlüklere bağlıdır. LWBR için yapılan modellemede de geometrinin doğruluğu incelenirken çok yüksek çözünürlüklerde yapılan çizimlerde Serpent terminalinin bu hata uyarılarını verdiği görülmüştür. Eğer bu hatalı bölgeler çok ufak bölgeleri kapsıyorlarsa bunları çizim dosyalarına bakarak tespit etmek mümkün olmamaktadır. Python dilinde bir görüntü işleme kodu yazılarak bu noktaların konumları elde edilmiş, daha sonra FreeCAD programında çizilen taslak üzerinden bu noktalar incelenmiştir. İncelemeler sonunda bu geometri hatası bölgelerinin hepsinin reaktörün en dışında bulunduğu ve birkaç noktadan oluşan ve geometrinin tamamı ile karşılaştırıldığında ihmal edilebilir hacimler kaplayan bu bölgeler “**set lost**” komutu kullanılarak transport hesaplamaları sırasında ihmal edilmesi sağlanmıştır.

Tezin 4. Bölümünde Serpent’in yanma hesaplamaları ile ilgili bilgilerin yer aldığı kısımlarda da bahsedildiği üzere Serpent varsayılan biçiminde yanma hesaplamalarını, yanma hesaplaması yapılacak her farklı malzeme için tek bölgede homojen olarak gerçekleştirir. Bu sonuçlarda büyük hatalara sebep olacak kaba bir yaklaşım olduğu için Serpent yanma bölgelerini otomatik olarak ayıran “**div**” komutuna sahiptir. Bu komut kullanılarak aynı malzemeler farklı yanma bölgelerine ayrılabilir. Tez çalışmasında malzemelerin yanma bölgelerine ayrımlarının otomatik olarak yapılabilmesi sadece yakıt modülleri seviyesinde ayrımlar ile mümkün olmuştur, yani reaktörde bulunan yakıt

malzemeleri “**div**” komutu ile sadece 12 yakıt modülünde farklı yanma bölgelerine ayrılabilmiştir. Eksenel yönde otomatik yanma bölgelerine ayırmak denendiğinde Serpent kodu ilk adımdan sonra hata verip durmuştur. Buna çözüm olarak yanma bölgeleri otomatik olarak değil, girdi dosyasında aynı malzemelerin farklı malzeme adları ile tekrar eder şekilde tanımlanarak ayrılmıştır. Bu şekilde girdi hazırlanmasının doğurduğu birkaç olumsuz sonuç olmuştur:

- Otomatik yanma bölgelerinin kullanıldığı yaklaşık 4 bin satırlık girdi dosyaları yerine 50 bin satırı geçen girdi dosyaları elde edilmiş ve Serpent kodunun girdilerde tekrarlanarak atanmış evren veya malzeme adlarının olup olmadığını kontrol eden rutininin çalışma süresi birkaç saniyeden yarım saat civarında bir süreye çıkmıştır. Yinelemelerle beraber Serpent kodu her hesaplamada yüzlerce defa tekrarlandığı için bu rutinin toplam hesaplama süresine katkısı oldukça fazla olmuştur.
- Otomatik yanma bölgeleri komutuna kıyasla hem transport hem de yanma hesaplamaları kısımlarında hesaplamanın tamamlanması için gereken süre çok fazla artmıştır.
- RAM gereksinimi ve hesaplama süreleri otomatik yanma bölgesi ayrımı seçeneğine kıyasla çok fazla arttığı için benzeşimlerde kullanılan nötron sayısının küçük değerlerde tutulması gerekmiştir. Çözüm olarak aktif ve pasif döngü sayıları arttırılmış ve kaynak dosyaları önceki yanma adımlarındaki aktif döngülerdeki nötron verilerinden alınmıştır.
- Çıktı dosyalarının boyutları oldukça artmış, bazı çıktı dosyalarının satır sayıları milyonlara ulaşmıştır. Sonuç elde edilmesinde bu dosyalardan verilerin elde edilmesi için de Python kodlarının yazılması gerekmiştir.

5. SONUÇLAR VE DEĞERLENDİRME

Başlık 4.3’de de bahsedildiği üzere hesaplama sürelerini makul sınırlar içerisinde tutarak referans değerlerine yakın üretkenlik değerleri elde edilene kadar hesaplamalar farklı girdi ayarları ile tekrarlanmıştır. Bu hesaplamalardan bazılarına ait sonuçlara bu kısımda yer verilmiştir.

CAD programı ile geometri modellemesi, Python kodunun yazılması ve Serpent girdisinin başlık 4.3’te anlatıldığı gibi hata vermeyecek şekilde hazırlanması gibi aşamalardan sonra girdilerde kullanılan hesaplama seçeneklerinin de en uygun hale getirilmesi gerekmiştir. Modellenen reaktör korunun yüksek heterojenliği nedeniyle, yanma bölgelerinin ayrıntılı tanımlanması ve uygun yanma hesaplaması seçeneklerinin kullanılması, referans değerlerine yakın sonuçlar elde etmede önemli rol oynamıştır.

5.1 Hesaplamalar

Hesaplamalarda benzeşimler 2-3 hafta arası sürelerde tamamlanmıştır. Yapılan hesaplamalar farklı yanma aralıkları ile gerçekleştirildiği için 18.000 ETGS’lik bir benzeşimin sonuçlanma süresi de her hesaplama için farklılık gösterir. Bu farklılıkta seçilen girdi seçeneklerinin sebep olduğu farklı hesaplama sürelerinin, girdi uzunluklarının (sebebi başlık 4.3’de açıklanmıştır) ve besleyici yakıt demetlerinin farklılık gösteren hareketlerinin sebep olduğu yinleme sürelerinin uzunluklarının da etkisi olmuştur.

Tezin yanma hesaplamaları ile ilgili başlıklarında da bahsedildiği üzere Serpent kodu yanma hesaplamalarını aynı malzemeye sahip evrenleri tek bir bölgede homojen olarak kabul ederek gerçekleştirir. Bu sebeple yanma hesaplamaları varsayılan geometri ile yapıldığında sonuçlarda büyük sapmalar olur. Örneğin bu tez çalışmasındaki LWBR koru homojen yanma hesaplamasında kullanıldığında ilk yanma adımından sonra reaktör tekrar kritikliğini sağlayamamaktadır. Bu duruma çözüm olarak iki seçenek vardır; Serpent kodunun sağladığı otomatik yanma bölgelerine ayırım komutu veya Serpent aynı malzemelerde aynı kompozisyonları farklı isimlere sahip malzemelerin tanımlanmasında kullanmak için girdinin tekrarlayan malzeme komutları ile kullanıcı tarafından düzenlenmesi. Yapılan hesaplamalarda Başlık 4.3’te açıklanan sebeplerden dolayı bu iki seçenek farklı şekillerde ve bazen beraber olarak kullanılarak hesaplamalar gerçekleştirilmiştir.

5.1.1. Tutarsız Hesaplama Denemeleri

Tezde 5.1.2. başlığında açıklanan 1. hesaplamasından önce bu teze dahil edilmeyen çeşitli başka benzeşimler gerçekleştirilmiştir. Örneğin ilk olarak otomatik olarak yanma bölgelerine ayırma seçeneği farklı evren seviyeleri için denenmiştir. Bunlardan biri kordaki tüm yakıt çubuklarının ayrı yanma bölgeleri olarak hesaplanmasının denendiği birkaç hesaplama çalışmasıdır. On binlerce yanma bölgesine ayrılan geometrinin çözümü için kullanılan FORMULA Yüksek Başarımlı Bilgisayar Kümesi'nin RAM donanımının hesaplama için yetersiz kalması sebebi ile Serpent kodu farklı optimizasyon seçenekleri ile ("**set opti**") çalıştırılmış fakat bu seçeneklerde artan hesaplama süreleri sebebi ile 18.000 ETGS'lik bir benzeşimin makul bir süre içerisinde tamamlanmasının mümkün olmayacağı görülmüştür. Daha sonra bunun yerine yakıt demeti seviyesinde otomatik yanma bölgesine ayırma seçenekleri kullanılarak hesaplamalar gerçekleştirilmesi denenmiştir. Bu durumda hesaplama süresi makul bir seviyeye indirilmiş olsa da az sayıda yanma bölgesi olması sebebi ile birkaç yanma adımından sonra benzeşimde kritiklik sağlanamamıştır. Yanma bölgesi sayısının az olması dışında sonuçlardaki tutarsızlığın bir diğer büyük sebebi benzeşimde hesaplama süresini arttıran bir seçenek olan "**set ures**" komutunun kullanılmamasıdır. Bu komut, ayrışmamış rezonans (unresolved resonance) için olasılık tablosu örneklemini etkin hale getirir [24]. Nötron enerjisinin epitermal değerler aldığı bölgelerde tesir kesiti değerlerinin tayini için kullanılan bu yöntem, hızlı veya LWBR gibi epitermal nötron spektrumlarına sahip reaktörlerin modellerinde etkin hale getirilmesi gerekmektedir. Bu komutun kullanılmaması sebebi ile ayrışmamış rezonans bölgesindeki öz-kalkanlama (self shielding) etkilerinin sonuçların sapmasına ve FEO değerinin 18.000 ETGS'ni geçmeden 1 değerinin altına düşmesine sebep olmuştur.

5.1.2. Hesaplama 1 ve 2

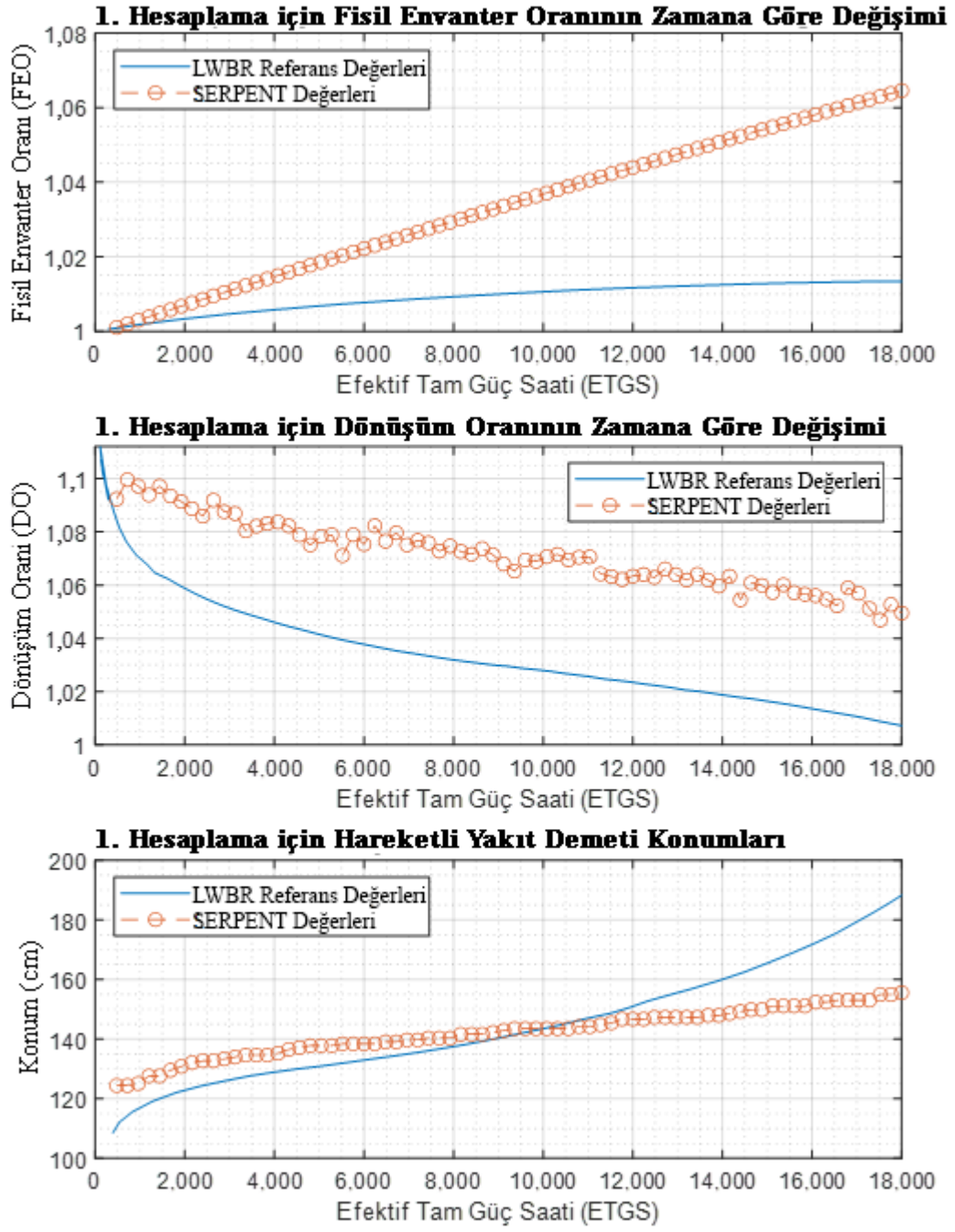
Bu hesaplarda besleyici yakıt demetleri Serpent girdisinde aynı malzemelerin tekrar tanımlanması ile aksel yönde 1'er inçlik aralıklarla farklı yanma bölgelerine ayrılmışlardır. Geometrinin geri kalanındaki malzemeler için yanma bölgesi ayrımı kullanılmamıştır.

Hesaplamalarda düşük nötron popülasyonunu telafi etmek için kaynak tanımlamaları kullanılmıştır. Her hesaplama için ilk yanma adımında kaynak noktaları fisil izotop içeren evrenler içerisinde ve 20.000 pasif döngüyü takip eden 1.000 aktif döngü için örneklenmiştir. Daha sonraki yanma adımlarda kaynak dosyası olarak kullanılmak için de aktif döngülerdeki örneklenen nötron verileri çıktı dosyalarına yazdırılmış ve Şekil 4.7'de gösterilen algoritma içerisinde hesaplamalar yapılmıştır.

Daha önceki kısımlarda da belirtildiği gibi yanma bölgesi ayrımı hiç yapılmadığında benzeşim birkaç yanma adımından sonra kritik olamamaktadır. Hesaplama 1 ve 2'nin amacı da mümkün olduğunca az yanma bölgesi kullanılarak sonuçlardaki sapmanın gösterilmesidir. Hesaplama 1 ve hesaplama 2 arasındaki farklar “set ures” komutunun hesaplama 2'de etkin olması ve hesaplama 2'de nötron popülasyonunun iki katına çıkarılıp aktif döngü sayısının yarıya indirilmesidir. Yanma bölgesi sayısının sonuçlara etkisi ile kıyaslanınca bu farklar sonuçlarda ihmal edilebilir değişikliklerden fazlasına yol açmamışlardır.

Çizelge 5.1 Hesaplama 1 için Serpent girdi seçenekleri

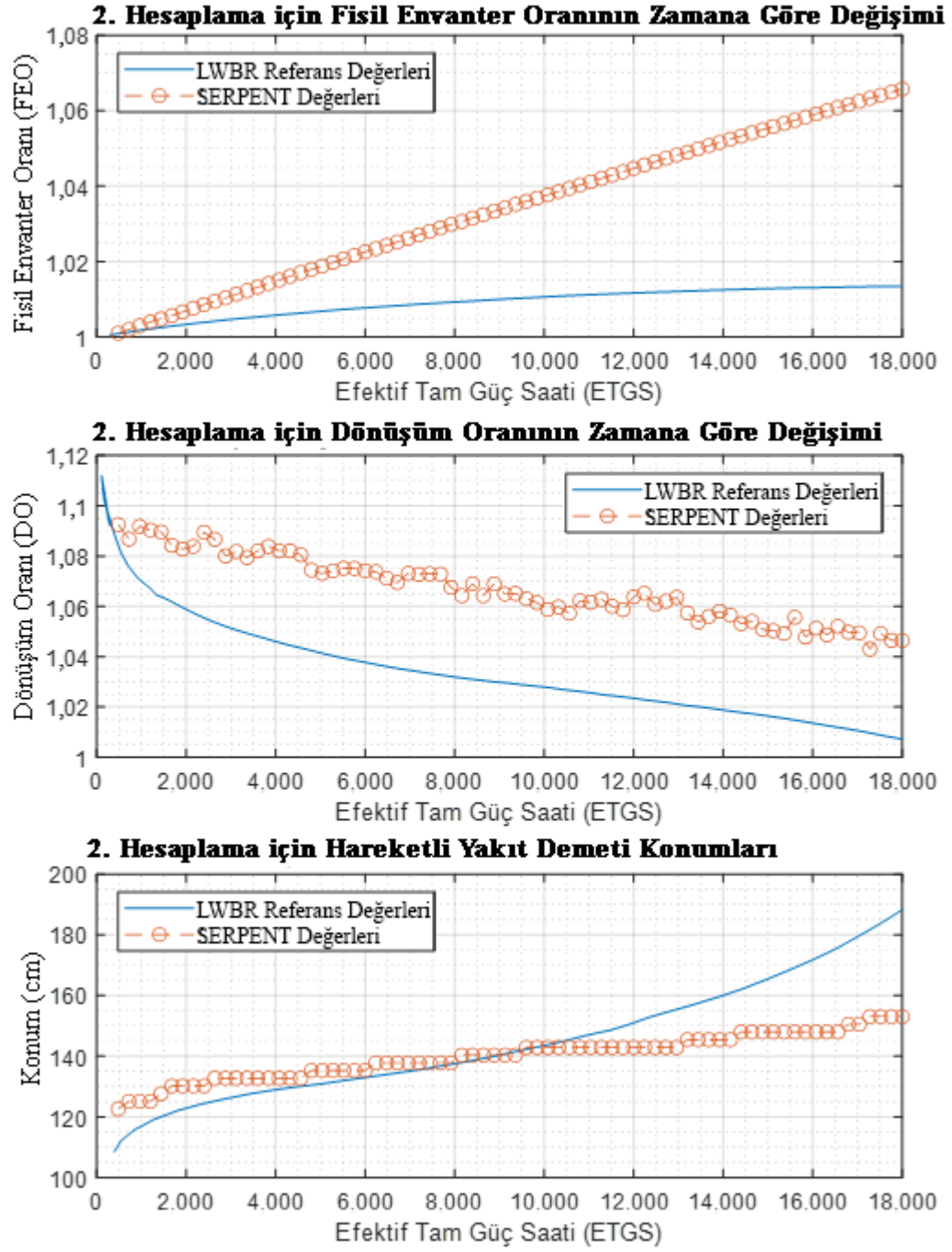
Hesaplama 1	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	1.000
Pasif Döngü Sayısı	2.000
Aktif Döngü Sayısı	1.000
Yanma Aralıkları	10 gün
Yanma Bölgesi Sayısı	92
Set ures	0
Set xenon	1
Set sie	2
Set fsp	1 200
Set usym	1/6 simetri
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için girdide tanımlanan 1 inçlik aksenal bölgeler



Şekil 5.1 1. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

Çizelge 5.2 Hesaplama 2 için Serpent girdi seçenekleri ve hesaplama süreleri

Hesaplama 2	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	2.000
Pasif Döngü Sayısı	2.000
Aktif Döngü Sayısı	500
Yanma Aralıkları	10 gün
Yanma Bölgesi Sayısı	92
Set ures	1
Set xenon	1
Set sie	2
Set fsp	1 500
Set usym	1/6 simetri
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için girdide tanımlanan 1 inçlik eksenel bölgeler



Şekil 5.2 2. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

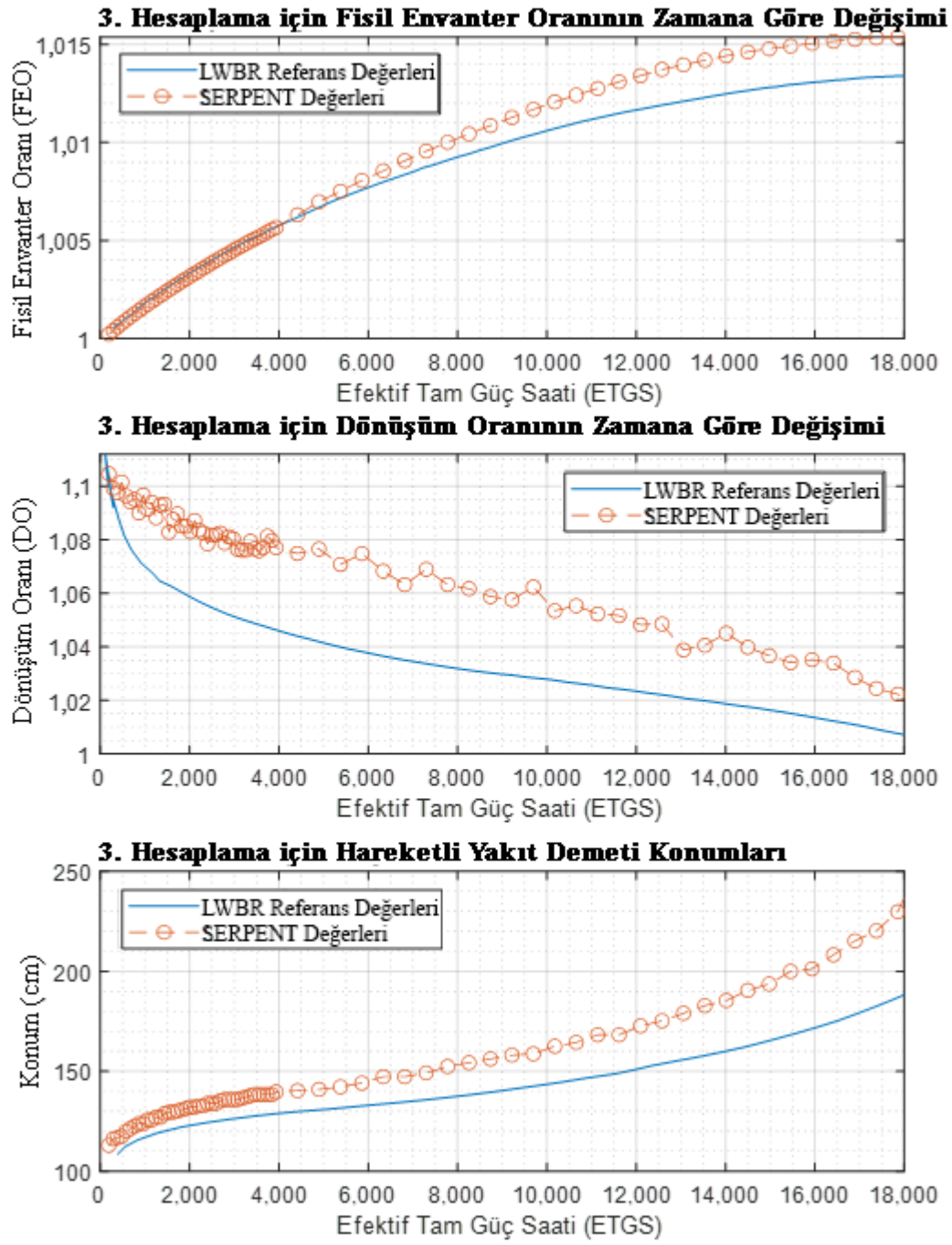
5.1.3. Hesaplama 3 ve 4

Hesaplama 1 ve 2’de olduđu gibi bu hesaplamalarda da besleyici yakıt bölgeleri eksenel yönde 1’er inç aralıklarla yanma bölgelerine ayrılmıştır ancak bunun yanında her modüldeki besleyici yakıt demetleri ayrı olarak girdide yanma bölgelerine ayrılmış ve hareketsiz bölgeler için de otomatik yanma bölgelerine ayırma komutu kullanılmıştır. Bu durumda 1950 farklı yanma bölgesi elde edilmiş ve benzeşimin tamamlanması için gereken süre kayda değer şekilde artmıştır. Ayrıca hesaplama 2 ve 3’ten farklı olarak denge durumu ksenon yoğunluğu (konsantrasyonu) durumunu geçerli kılan “set xenon” komutu ve hesaplamaları 1/6 simetride yapılmasını sağlayan “set usym” komutu bu hesaplamalarda etkinleştirilmemiştir.

Hesaplama 3 için hareketli yakıt demetlerinin daha hızlı hareket ettikleri başlangıç için yanma adımı aralıkları 4 gün, daha sonrası için ise 20 gün aralıklarla hesaplamalar gerçekleştirilmiştir. 4. Hesaplama için ise tüm yanma aralıkları 4 gün olarak alınmıştır

Çizelge 5.3 Hesaplama 3 için Serpent girdi seçenekleri ve hesaplama süreleri

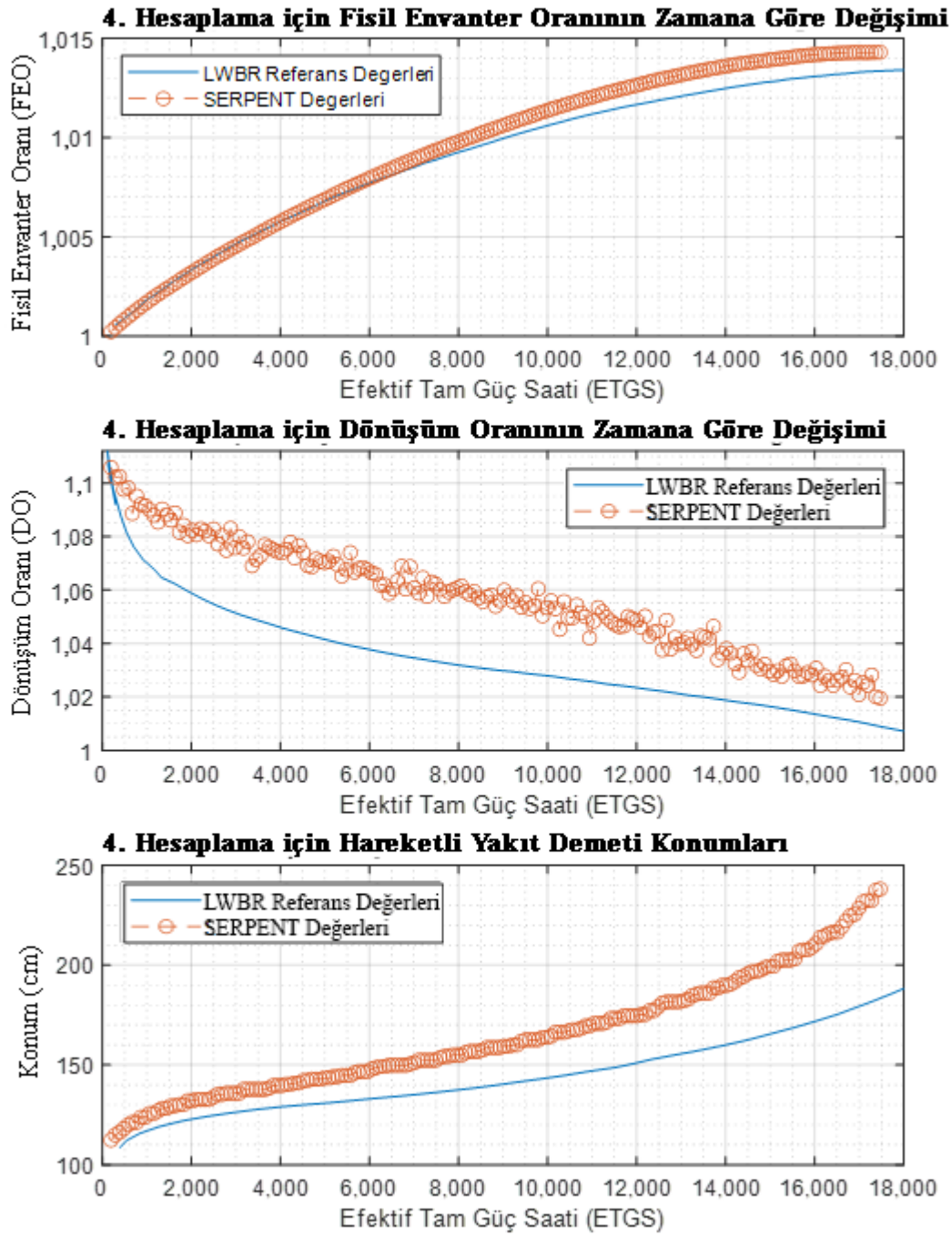
Hesaplama 3	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	2.000
Pasif Döngü Sayısı	2.000
Aktif Döngü Sayısı	500
Yanma Aralıkları	İlk 4.000 ETGS için 4 gün, kalan süre için 20 gün
Yanma Bölgesi Sayısı	1950
Set ures	1
Set xenon	0
Set sie	2
Set fsp	1 500
Set usym	-
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için girdide tanımlanan 1 inçlik aksenal bölgeler, örtü ve ThO ₂ bölgeleri için yakıt demeti seviyesinde otomatik ayırım



Şekil 5.3 3. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

Çizelge 5.4 Hesaplama 4 için Serpent girdi seçenekleri ve hesaplama süreleri

Hesaplama 4	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	2.000
Pasif Döngü Sayısı	2.000
Aktif Döngü Sayısı	500
Yanma Aralıkları	4 gün
Yanma Bölgesi Sayısı	1.950
Set ures	1
Set xenon	0
Set sie	2
Set fsp	1 200
Set usym	-
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için girdide tanımlanan 1 inçlik eksenel bölgeler, örtü ve ThO ₂ bölgeleri için yakıt demeti seviyesinde otomatik ayırım



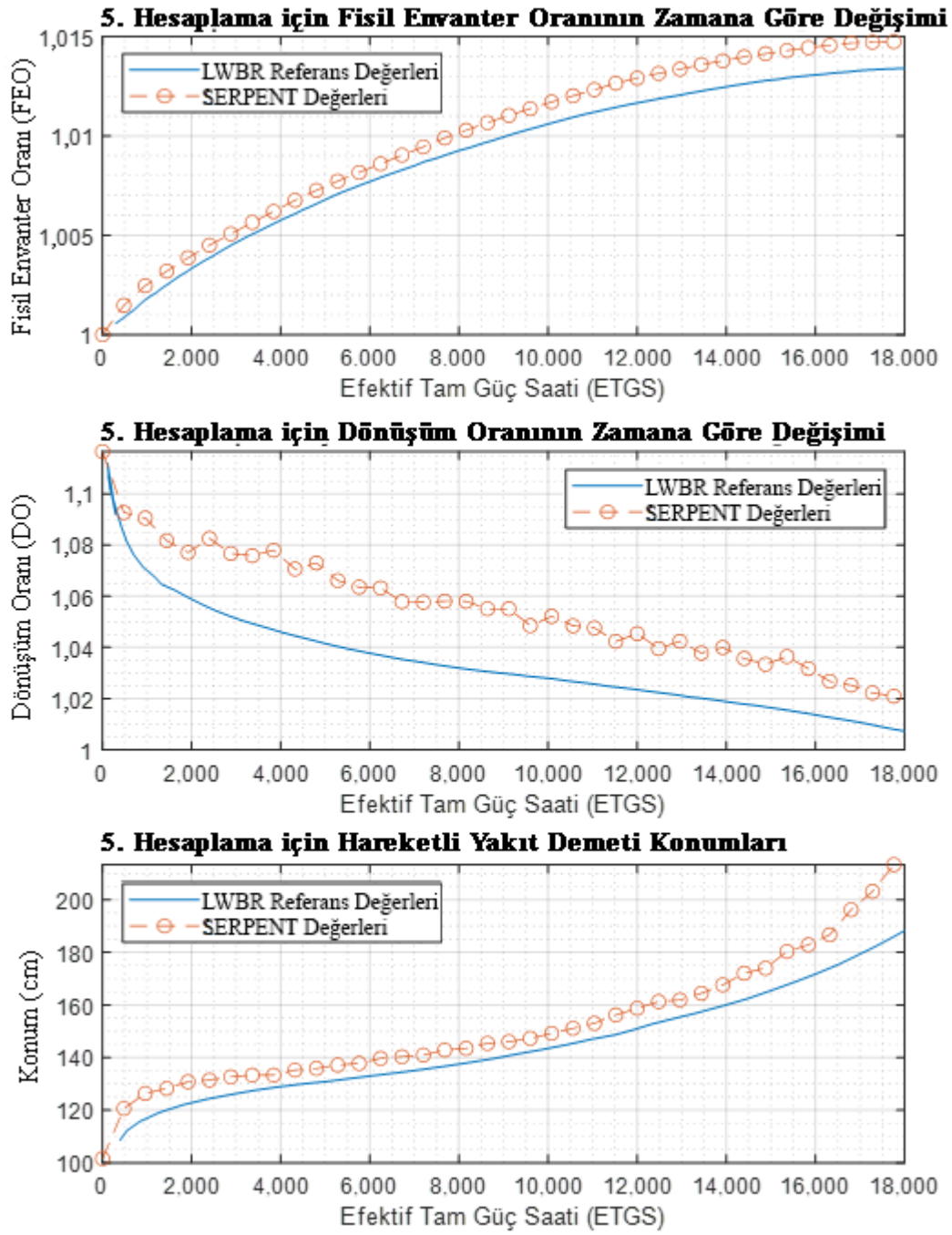
Şekil 5.4 4. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

5.1.4. Hesaplama 5 ve 6

Bu hesaplamalar için besleyici yakıt demetlerinde eksenel yöndeki yanma bölgelerinin sayısı azaltılıp yansıtıcı modüller dışındaki hareketsiz bölgeler de eksenel yönde 6 yanma bölgesine ayrılmıştır. Hesaplama 3 ve 4 ile kıyasla daha az sayıda yanma bölgesi olmasına rağmen eksenel yönde ayrılmış yanma bölgeleri sadece besleyici yakıt demetlerinde değil tüm kora yayılmış şekilde girdi dosyaları düzenlenmiştir. 5. Hesaplama ve 6. Hesaplama arasındaki fark yanma aralıklarının 20 günden 4 güne düşürülmesidir.

Çizelge 5.5 Hesaplama 5 için Serpent girdi seçenekleri ve hesaplama süreleri

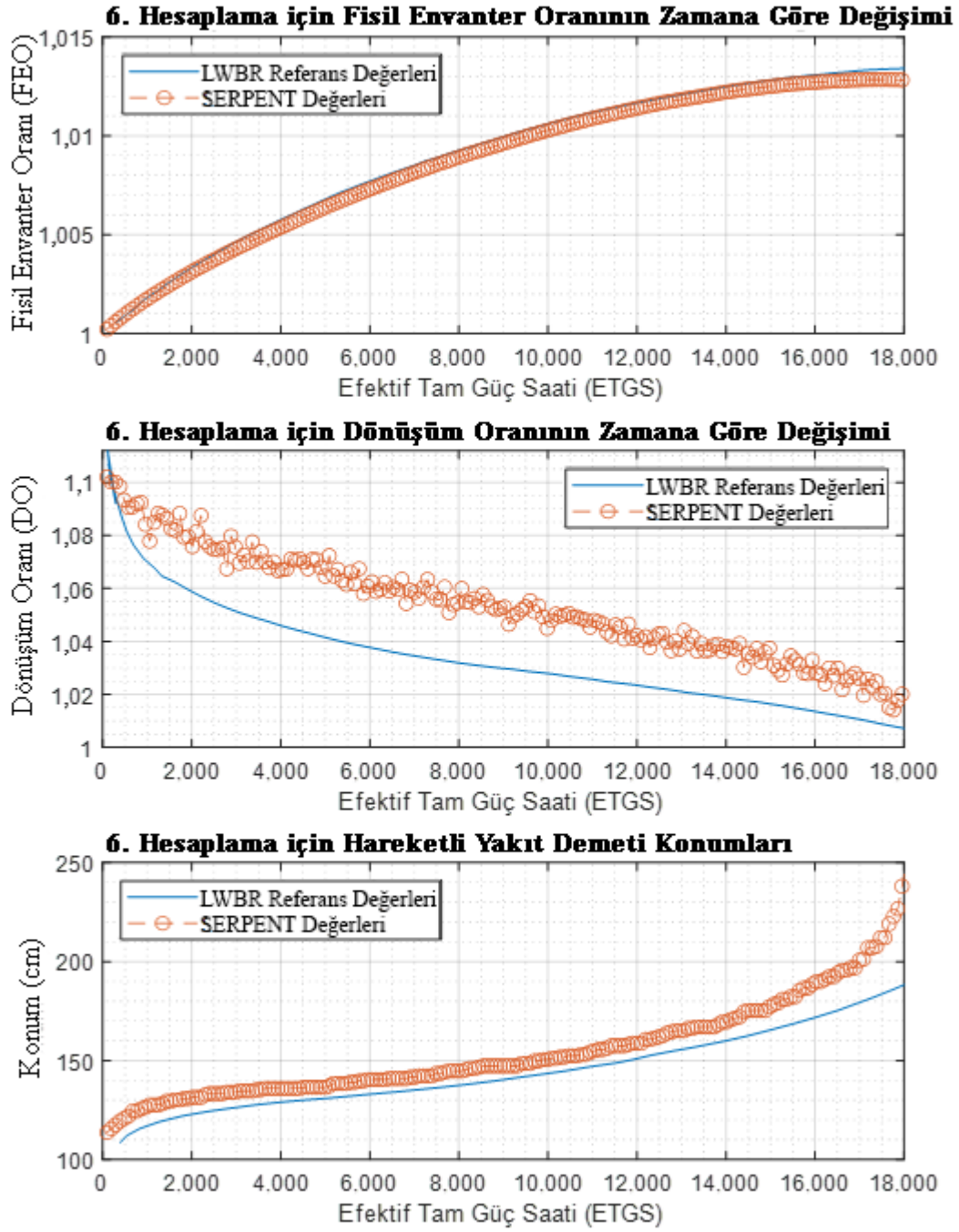
Hesaplama 5	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	2.000
Pasif Döngü Sayısı	200
Aktif Döngü Sayısı	500
Yanma Aralıkları	20 gün
Yanma Bölgesi Sayısı	689
Set ures	1
Set xenon	0
Set sie	2
Set fsp	1 200
Set usym	-
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için 7 inçlik, örtü bölgeleri için 14 inçlik aralıklarla girdide tanımlanmış eksenel bölgeler, yansıtıcı ThO ₂ bölgeleri için otomatik ayırım.



Şekil 5.5 5. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

Çizelge 5.6 Hesaplama 6 için Serpent girdi seçenekleri ve hesaplama süreleri

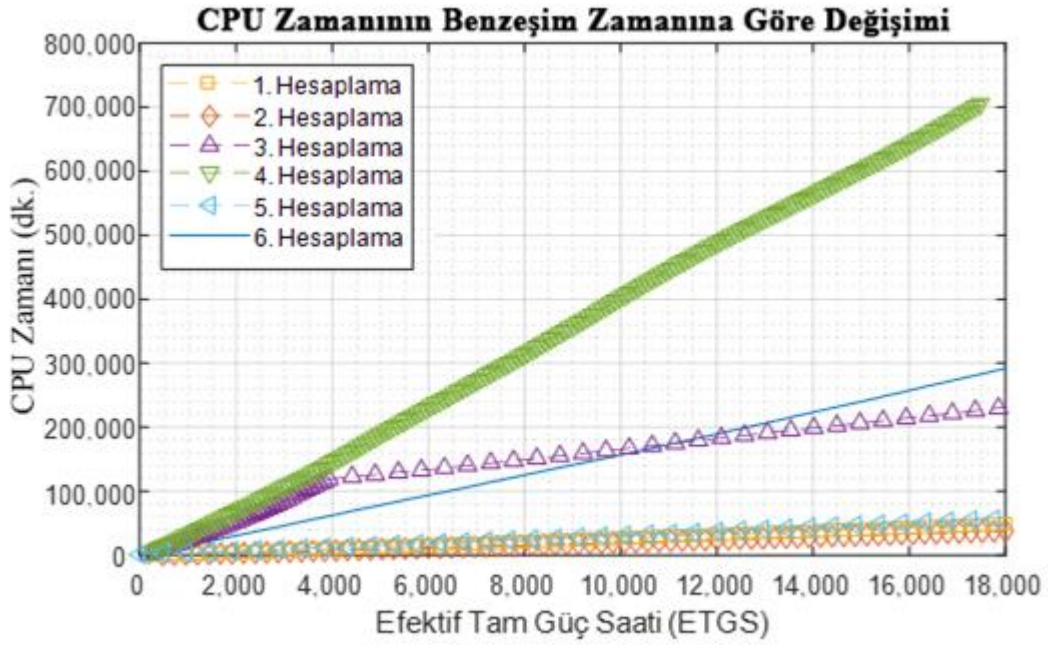
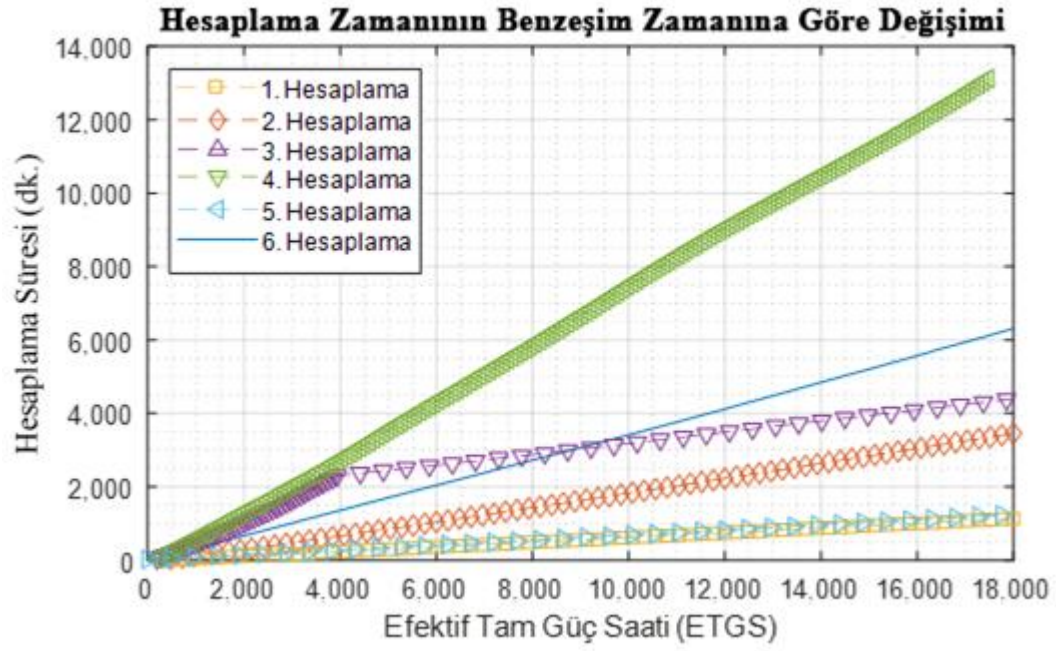
Hesaplama 6	
Bağlantı Şeması	Stokastik Örtük Euler
Nötron Popülasyonu	2.000
Pasif Döngü Sayısı	200
Aktif Döngü Sayısı	500
Yanma Aralıkları	4 gün
Yanma Bölgesi Sayısı	689
Set ures	1
Set xenon	0
Set sie	3
Set fsp	1 200
Set usym	-
Src	Önceki adımın aktif döngüleri kaynak olarak tanımlanır.
Yanma Bölgesi Ayrımı	Besleyici yakıt demetleri için 7 inçlik, örtü bölgeleri için 14 inçlik aralıklarla girdide tanımlanmış eksenel bölgeler, yansıtıcı ThO ₂ bölgeleri için otomatik ayırım.



Şekil 5.6 6. Hesaplama için üretkenlik sonuçları ve kritik yakıt demeti konumları

Çizelge 5.7 Hesaplama adımları arasında yapılan değişikliklerin özeti

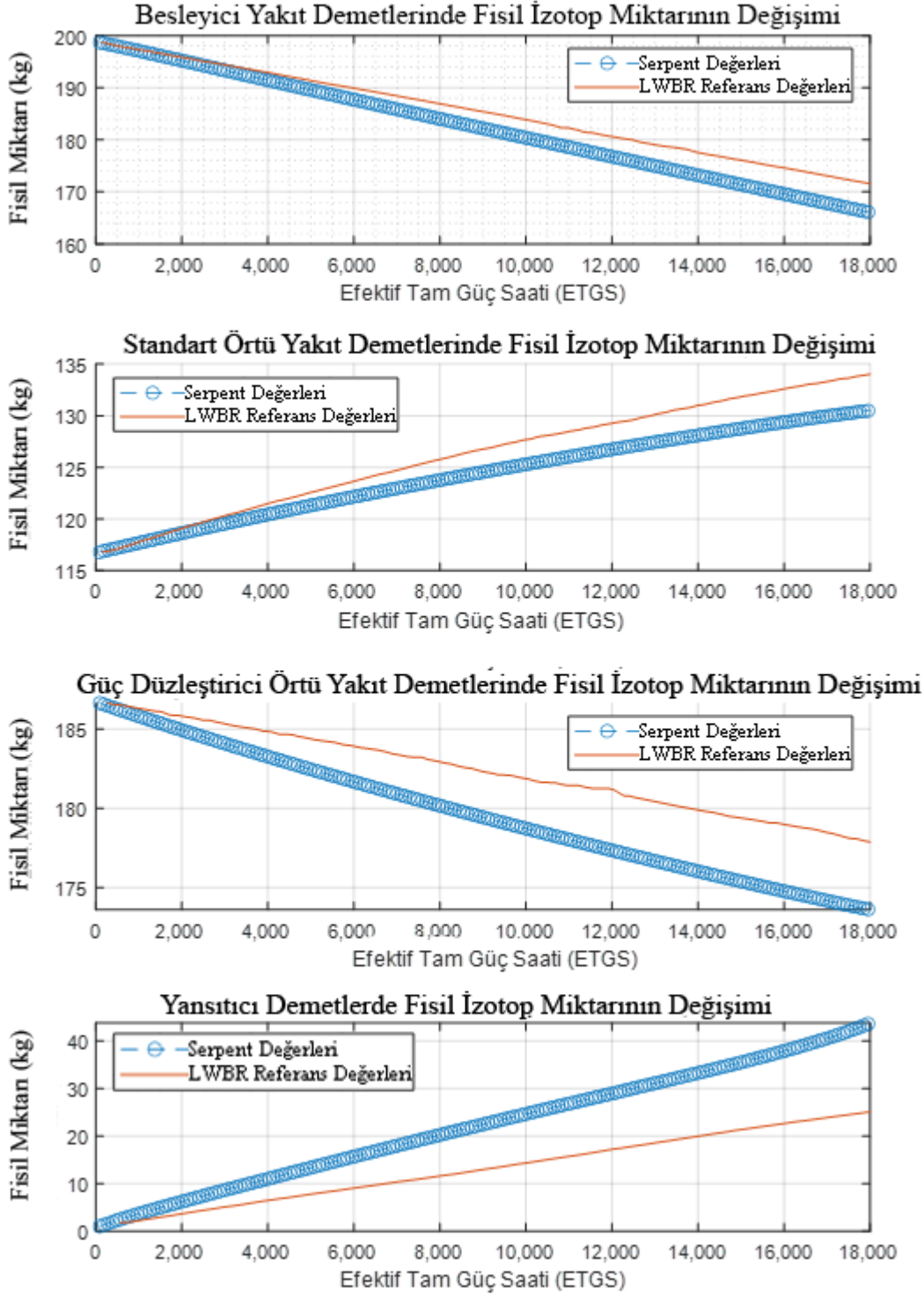
Hesaplama 1 → 2	Nötron popülasyonu ve döngü sayıları değiştirildi, “set ures” komutu etkin hale getirildi
Hesaplama 2 → 3	Denge durumu ksenon hesaplaması kapatıldı, hareketsiz kısımlarda otomatik yanma bölgesi ayrımı uygulandı. Besleyici yakıt demetleri için girdi dosyalarında 1 inçlik eksenel ayrımlara ek olarak yakıt demeti seviyesinde de yanma bölgesi ayrımları yapıldı.
Hesaplama 3 → 4	Yanma aralıkları azaltıldı
Hesaplama 4 → 5	Besleyici bölge için girdide tanımlanan yanma bölgelerinin sayısı azaltılıp korun geri kalan kısımlarına da uygulandı
Hesaplama 5 → 6	Yanma aralıkları 20 günden 4 güne düşürüldü



Şekil 5.7 Hesaplamalar için Serpent çözüm sürelerinin benzeşim zamanına göre değişimi

Gerçekleştirilen hesaplamalar için çözüm süreleri Şekil 5.8’de verilmiştir. Bu süreler Serpent kodunun transport hesaplamasına başladığı andan yanma hesaplamalarını bitirdiği ana kadar geçen sürelerdir. Bu sürelere yineleme adımlarında harcanan süreler ve transport hesaplaması başlayana kadar Serpent kodunda harcanan süreler dâhil değildir. Normal bir Serpent benzeşiminde transport hesaplamaları başlayana kadar geçen süreler transport hesaplamasının için harcanan sürelere kıyasla genelde ihmal edilebilir kısıklıkta olmaktadır. Bu tez çalışmasında ise özellikle aynı malzeme veya evren isimlerinin birden fazla defa tekrarlanıp tekrarlanmadığını kontrol eden rutinlerin uzunluğu sebebi ile bu süreler ihmal edilebilir kısıklıklarda değildir. Şekil 5.7’ye dahil olmayan yineleme adımlarında dahi bu rutinlerin tekrar tekrar yapılması sebebi ile gerçek hesaplama uzunlukları grafikte okunabilenin en az birkaç katına tekabül eder. Örneğin Hesaplama 6 için hesaplama süresi Şekil 5.7’den yaklaşık 5 gün olarak görülse de gerçekte bu hesaplama Python kodunun çalıştırılmasından bitişine kadar yaklaşık 25 gün sürmüştür.

Şekil 5.8’de, 6. Hesaplama için fisil izotop miktarlarının zamanla değişimi verilmiştir. Fisil izotop miktarlarındaki değişimler beklenen davranışta olsa da hesaplamanın sonunda referans değerler ile aralarında bir miktar fark bulunmaktadır. Daha isabetli sonuçların nötron popülasyonu ve yanma bölgesi sayısının miktarlarının arttırılarak benzeşimin tekrarlanması ile mümkün olacağı öngörülebilir. Bu çalışma için bir sonraki başlıkta da bahsedilen nedenlerden dolayı böyle bir benzeşim gerçekleştirilmemiştir.



Şekil 5.8 Hesaplama 6 için yakıt bölgelerindeki fisil izotop miktarının zamana göre değişimi.

6. TARTIŞMA VE YORUM

Bu tez çalışmasında LWBR için tam güçte sürekli işletme durumundaki üretkenlik parametreleri, Serpent Monte Carlo kodu ile ve Python programlama dilinde yazılmış bir kod ile tüm korun 3 boyutlu modeli için yaklaşık 18.000 saatlik nötronik benzeşimi hesaplamaları gerçekleştirilerek incelenmiştir.

Hacettepe Üniversitesi Nükleer Enerji Mühendisliği Bölümü bünyesinde bulunan FORMULA Yüksek Başarımlı Bilgisayar Kümesi'nin olanakları sınırlarında yapılan benzeşimlerde FEO için LWBR değerlerine yeteri yakınlıkta sonuçlar elde edilmiş ve böylece bu çalışmanın sonucunda LWBR tasarımının fisil izotop üretkenliği konusundaki başarımı Monte Carlo yöntemi ile de doğrulanmıştır. Bu çalışmadaki teorik hesaplamalar, reaktörün tam güçle 2 yıl boyunca işletildiğinde dönüşüm oranının 1 değerinin altına düşmediğini göstermektedir ve bu toryumun enerji üretimine verimli bir şekilde katkı sağladığını göstermektedir. Yapılan çalışma, verimli bir toryum çevrimi için gündemde olan ergimiş tuz yakıtlı reaktörler ve hızlandırıcı güdümlü kritik-altı reaktör tasarımlarına alternatif olarak bu tasarımın kullanılabileceğini göstermektedir. Bu tasarımın bilinen PWR tasarımlarına benzer bir soğutma çevrimine sahip olması ve PWR tasarımlarının yaygınlığı sebebiyle LWBR'ın ticari bir reaktör haline getirilmesi ile ilgili çalışmalar yapılmasının önemi anlaşılmaktadır.

Reaktör geometrisindeki düzgün olmayan yakıt modülleri, heterojen yakıt dizilimleri ve hareketli yakıt demetleri sebebi ile benzeşimlerin gerçekleştirilmesi yüksek RAM ihtiyacı ve uzun hesaplama süreleri gerektirmiştir. Bunun yanında literatürde Serpent için benzer çalışmalara rastlanmaması sebebi ile modelleme ve benzeşim aşamalarında karşılaşılan bazı sorunların deneme ve yanılma yöntemleri ile aşılması gerekmiştir. Elde edilen birkaç daha az isabetli hesaplama da sonuçların iyileştirilmesi safhalarının gösterilmesi adına bu teze eklenmiştir.

Tezde elde edilen sonuçların yakınlığı Serpent Monte Carlo kodunun zaman aralıklarının ve yanma bölgelerinin uygun şekilde belirlendiği durumlarda LWBR gibi değişen geometriye sahip reaktörlerin bile yanma hesaplamalarını yeterli isabetlilikle gerçekleştirebildiğini göstermiştir. Fakat, bu hassasiyetteki hesaplamalar için harcanan hesaplama gücü ve süreleri dikkate alındığında, daha gelişmiş bilgisayar teknolojilerine ulaşılabilir olunmadığı sürece, söz konusu kodun kullanımının pratik olmadığı sonucuna varılabilir. Öte yandan bu çalışmada kullanılan Serpent kodunun sürümünün (Serpent

2.1.30) daha sonraki sürümlerine kıyasla hesaplama süresi veya verimliliğinde daha alt kalitede olabileceği de dikkate alınmalıdır.

Hesaplamalar için gereken RAM ihtiyacının ve hesaplama sürelerinin Serpent kodunda bulunan ve geometriyi otomatik yanma bölgelerine ayıran komutun kullanımı ile azaltılması mümkündür fakat kodun kullanılan sürümünde hareketli yakıt demetlerinde eksenel yönde otomatik yanma bölgeleri ayrımı kodun hata verip çalışmasını durdurması ile sonuçlanmıştır. Elde edilen sonuçlardan görüleceği üzere tüm kor geometrisinde eksenel yönde yanma bölgesi ayrımı yapılması ve yanma adımları arasında aktif döngü nötron verilerinin aktarılması literatürdeki çalışmalara kıyasla daha az nötron popülasyonu kullanılmasına rağmen FEO için yeterince isabetli bir benzeşim gerçekleştirilmesini sağlamaktadır.

Bu çalışma sonunda yaygın Monte Carlo kodları ile üç boyutlu geometrilerdeki yanma hesaplamaları ile ilgili bilinen isabetsizliklerine rağmen yeterli hesaplama gücü ve uygun hesaplama seçenekleri ile Serpent kodunun karmaşık reaktör sistemlerinin üretkenliklerinin benzeşimini gerçekleştirebileceği sonucuna varılmıştır.

7. DEVAM ÇALIŞMALARI İÇİN ÖNERİLER

Bu çalışmada Serpent kodu ile gerçekleştirilen benzeşim hesaplamalarının sonucunda üretkenlik parametreleri için referans değerlerine yeterli yakınlıkta değerler elde edilmiş olması ile birlikte birkaç noktanın geliştirilmesi ile daha isabetli sonuçların elde edileceği benzeşimler yapmak mümkündür.

1. Bu çalışmada reaktörün devamlı olarak tam güç çalışması durumundaki değerler karşılaştırılmıştır. LWBR için gerçek işletme geçmişindeki değişen güç durumları için çalışma tekrarlanıp reaktörün gerçek üretkenlik parametreleri ile karşılaştırma yapılabilir.
2. Çalışmada gerçekleştirilen hesaplamalarda termal hidrolik bir benzeşim ile bağdaştırma yapılmamıştır. Özellikle ticari ölçekte üretim yapan bir reaktörde, termal hidrolik sınırlamalar önem kazanmaktadır. Söz konusu kısıtlamaların, dönüşüm oranına olası etkilerinin incelenmesi zorunludur. Ayrıca, termal-hidrolik analizle, nötronik analizin bağdaştırılması, elde edilen sonuçların daha gerçekçi olmasını da sağlayacaktır.
3. Yapılan hesaplamalar, tek bir sunucuda Serpent kodunun OpenMP özelliği kullanılarak işlemci çekirdekleri arasında girdi kodlarının paralel koşulması ile gerçekleştirilmiştir. Serpent kodunun farklı sunucu veya bilgisayarlar arasında MPI özelliği ile paralel koşma özelliği de vardır. Bu özellik bu çalışmada denenmiş fakat kullanılamamıştır. Bu özelliğin kullanıldığı bir çalışmada daha fazla nötron popülasyonu ve daha fazla yanma bölgesi ayrımlarının kullanılması mümkün olacağı için daha isabetli sonuçlar elde edilmesi mümkündür.
4. Bu çalışmada yapıldığı gibi Python dilinde yazılan harici bir kod ile yinelemelerin gerçekleştirilmesi yerine Serpent kodunun kaynak dosyalarının değiştirilip yeniden derlenmesi ile daha hızlı bir yineleme düzeni elde edilebilir.
5. Çalışmada kullanılan Serpent kodunun sürümü güncel değildir, güncel sürümler performans açısından avantaj sağlayabilir. Ayrıca bu sürümde denendiğinde hata alınan otomatik yanma bölgesi ayrımı gibi özellikler güncel sürümlerde çalışmaya uyumlu şekilde çalışması durumunda da benzeşim süreleri büyük oranda kısalabilir.

LWBR için yapılan bu çalışmanın iyileştirilmesi adına sıralanan bu önerilerin dışında ayrıca Gelişmiş Hafif Sulu Üretken Reaktör Uygulamaları (Advanced Water Breeder Applications, AWBA) programı dâhilinde geliştirilen reaktör tasarımları için de Serpent kodu ile nötronik analiz benzeşimlerinin gerçekleştirilmesi mümkündür. Bu reaktör tasarımlarının bazıları ile ilgili özet bilgilere EK-A kısmında yer verilmiştir.



8. KAYNAKLAR

- [1] INTERNATIONAL ATOMIC ENERGY AGENCY, World Distribution of Uranium Deposits, Non-serial Publications, IAEA, Vienna, **2018**
- [2] INTERNATIONAL ATOMIC ENERGY AGENCY, World Distribution of Thorium Deposits, Non-serial Publications, IAEA, Vienna, **2021**
- [3] Budd, W. A. Shippingport operations with the Light Water Breeder Reactor core, report, March 1, **1986**
- [4] Water cooled breeder program summary report, report, October 1, **1987**
- [5] Golesorkhi and Sourena, Evaluation of the Net Breeding Capability of Heterogeneously Fueled Pressure-Tube Heavy Water Reactors with the Thorium Fuel Cycle, **2015**
- [6] Leppänen, J., Pusa, M., Viitanen, T., Valtavirta, V. and Kaltiaisenaho, T. "The Serpent Monte Carlo code: Status, development and applications in 2013." Ann. Nucl. Energy, 82 (2015) 142-150, **2013**
- [7] Hecker, H. C. & Freeman, L. B. Design features of the Light Water Breeder Reactor (LWBR), which improve fuel utilization in light water reactors, report, **1981**
- [8] Rampolla, D. S.; Conley, G. H.; Candelore, N. R.; Cowell, G. K.; Estes, G. P.; Flanery, B. K. et al. Fuel utilization potential in light water reactors with once-through fuel irradiation, report, **1979**
- [9] Hecker, H. C. Nuclear analysis and performance of the Light Water Breeder Reactor (LWBR) core power operation at Shippingport, report, April 1, **1984**
- [10] Illum, D. B.; Olson, G. L. & McCardell, R. K. Fuel Summary Report: Shippingport Light Water Breeder Reactor, report, January 1, **1999**
- [11] Stout, J.W. Lerner, S. McWilliams, K.D. & Turner, J.R. (editors.). Summary of the hydraulic evaluation of LWBR, report, April 1, **1981**
- [12] Richardson, K. D. Summary of the Fuel Rod Support System (Grids) Design for LWBR (LWBR Development Program), report, February 1, **1979**
- [13] Connors, D. R. Milani, S. Fest, J. A; Atherton, R. Design of the Shippingport Light Water Breeder Reactor (LWBR Development Program), **1979**
- [14] Hecker, H. C. Summary of the nuclear design and performance of the Light Water Breeder Reactor (LWBR), report, June 1, **1979**
- [15] Freeman, L. B. Computational model used in the analysis of nuclear performance of the Light Water Breeder Reactor (LWBR), report, August **1978**
- [16] Isotalo, A. "Computational methods for burnup calculations with Monte Carlo neutronics." D.Sc. Thesis, Aalto University, **2013**.

- [17] I. Lux and L. Koblinger. Monte Carlo Particle Transport Methods: Neutron and Photon Calculations. CRC Press, inc. **1991**.
- [18] Leppänen, J. "On the use of delta-tracking and the collision flux estimator in the Serpent 2 Monte Carlo particle transport code." Ann. Nucl. Energy, 105 161-167, **2017**
- [19] Leppänen, J. "Development of a new Monte Carlo reactor physics code." D.Sc. Thesis, Helsinki University of Technology, **2007**.
- [20] E. R. Woodcock et al. Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry. In Proceedings of the Conference on Applications of Computing Methods to Reactor Problems, ANL-7050. Argonne National Laboratory, **1965**.
- [21] Leppänen, J. "Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code." Ann. Nucl. Energy, 37 715-722, **2010**
- [22] Isotalo, A. "Computational methods for burnup calculations with Monte Carlo neutronics." D.Sc. Thesis, Aalto University, **2013**.
- [23] Bateman, H. Solution of a system of differential equations occurring in the theory of radioactive transformations. Math. Proc. Cambridge 15, 423–427. (**1910**)
- [24] Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code June 18, **2015** User's Manual Jaakko Leppänen
- [25] Chadwick, M.B. et al. ENDF/B-VII.1 Nuclear Data for Science and Technology: Cross Sections, Covariances, Fission Product Yields and Decay Data. Nucl. Data Sheets 112, 2887–2996. (**2011**)
- [26] Pusa, M., Leppänen, J. Computing the Matrix Exponential in Burnup Calculations. Nucl. Sci. Eng. 164, 140–150. (**2010**)
- [27] Leppänen, J. and Isotalo, A. "Burnup calculation methodology in the Serpent 2 Monte Carlo code." In proc. PHYSOR 2012, Knoxville, TN, Apr. 15-20, **2012**.
- [28] Dufek, J. Hoogenboom, J.E. **2009**. Numerical stability of existing Monte Carlo burnup codes in cycle calculations of critical reactors. Nucl. Sci. Eng. 162, 307– 311
- [29] Dufek, J. Kotlyar, D. and Shwageraus, E. "The stochastic implicit Euler method – A stable coupling scheme for Monte Carlo burnup calculations." Ann. Nucl. Energy, 60 (**2013**) 295-300.
- [30] Juergen Riegel, Werner Mayer, Yorik van Havre (2001-2022). FreeCAD (Sürüm 0.20.1) [Yazılım] <http://www.freecadweb.org>.
- [31] Conley, G. H.; Cowell, G. K.; Detrick, C. A.; Kusenko, J.; Johnson, E. G.; Dunyak, J. et al. Three core concepts for producing uranium-233 in commercial pressurized light water reactors for possible use in water-cooled breeder reactors, report, December 1, **1979**; West Mifflin, Pennsylvania.

[32] Vogel, K. H. (editor), Advanced Water Breeder Applications Work – a Summary (AWBA), **1982**

[33] Johnson, E. G. (editor), Design Concept for a 900 Mw(E) Scale-Up of the 60 Mw(E) Shippingport LWBR (AWBA Development Program), **1982**

[34] Webb, G. B. (editor), Conceptual Design of a Movable Thoria Finger Rod Controlled Light-Water Breeder Reactor (AWBA Development Program), **1982**



EKLER DİZİNİ

EK A. AWBA Reaktörleri

1976 yılında LWBR için yapım çalışmalarının tamamlanmasına yakın bir zamanda Amerika Birleşik Devletleri Enerji Bakanlığı, bu teknolojinin ticari ölçeklere aktarılması için Gelişmiş Hafif Sulu Üretken Reaktör Uygulamaları (Advanced Water Breeder Applications, AWBA) programını başlatmıştır.

Bu program kapsamında ticari ölçekte LWBR uygulamaları için birkaç farklı reaktör konsepti üzerine çalışmalar yapılırken aynı zamanda bu üretken reaktörlerin ihtiyaçları olan U-233 izotoplarının üretimi safhası için “prebreeder” reaktör konseptleri üzerine de çalışmalar yapılmıştır.

Bahsedilen “prebreeder” ve diğer üretken reaktörler ile ilgili çalışmalara olan erişimin sınırlılığı sebebi ile bu kısımda yalnızca özet bilgilere yer verilmiştir.

EK.A.1 AWBA Prebreeder Reaktörleri

U-233 doğada bulunan bir izotop olmadığı için üretken reaktörlerde kullanılacak miktarlarda U-233 izotopunun prebreeder reaktörlerden elde edilmesi gerekmektedir. Bu reaktörler için önerilen konseptler LWBR tasarımına benzer şekilde besleyici ve örtü yakıt bölgelerinden oluşan, reaktivite kontrolünün hareketli yakıt demetleri ile sağlandığı tasarımlar ve yaygın PWR korları gibi kare dizilimde reaktivitenin soğutucudaki borik asit oranının değiştirilerek sağlandığı tasarımlardan oluşur. LWBR koruna benzeyen prebreeder kor tasarımları PWR korlarına benzeyenlere kıyasla daha yüksek U-233 üretim hızına sahiptirler [31]. Referans verilen çalışmada 3 farklı prebreeder kor konsepti özetlenmiştir.

1. kor, zenginleştirilmiş uranyum besleyici yakıt çubukları ve ThO₂ örtü çubuklarından oluşur. 2. korda da besleyici yakıt çubuklarında zenginleştirilmiş uranyum bulunurken örtü yakıt çubukları ThO₂ ve plütonyumdan oluşur. 3. kor ThO₂ örtü çubukları ile ThO₂ ve zenginleştirilmiş uranyum içeren besleyici yakıt çubuklarından oluşur.

Bu korların yıllık Uranyum-233 üretimleri karşılaştırıldığında 1.2. ve 3. korların sırasıyla 161, 255 ve 198 kg üretim potansiyelleri olduğu hesaplanmıştır. Böylece 1000 MWe gücünde bir LWBR reaktörünün ihtiyacı olan U-233 miktarı için beş adet prebreeder reaktörün 5 ile 8 yıl arası üretim yapması gerekmektedir [31].

EK.A.2 AWBA Üretken Reaktör Konseptleri

LWBR teknolojisinin başarısının ticari ölçeklere aktarılması için Bettis Atomik Güç Laboratuvarı (Bettis Atomic Power Laboratory, BAPL) ve Knolls Atomik Güç Laboratuvarı (Knolls Atomic Power Laboratory, KAPL) toplamda dört farklı reaktör koru geliştirmişlerdir [32]. Birçok tasarım özelliği açısından birbirlerinden farklılık gösteren bu reaktörler yaklaşık 900-1000 MW elektrik gücünde olacak şekilde tasarlanmışlardır.

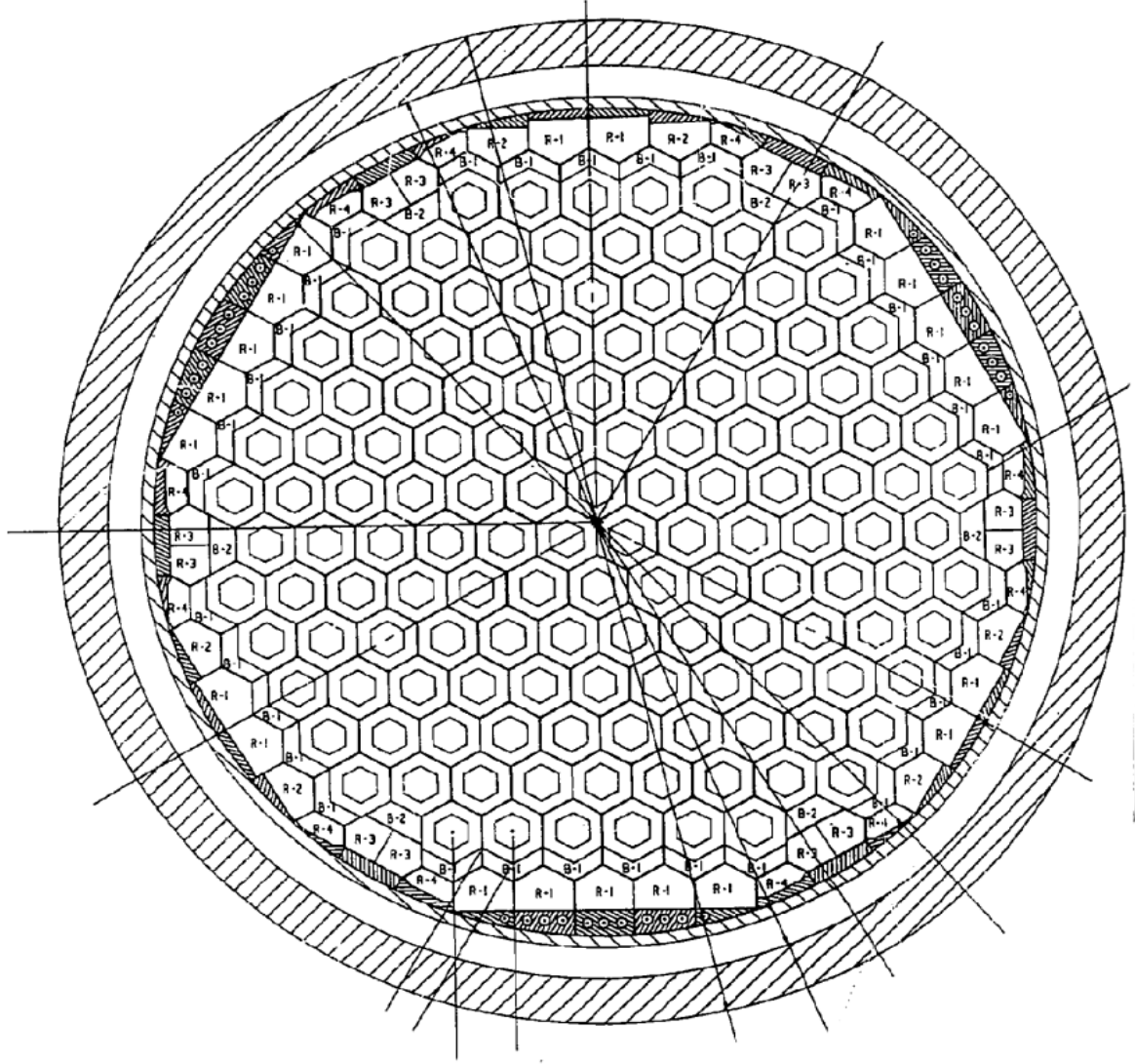
Tasarlanan üretken reaktörlerden ilki, LWBR korunun daha fazla yakıt modülü ile ticari ölçeklere çıkarıldığı bir tasarımdır. İkinci tasarım, LWBR korundaki gibi reaktivite kontrolünün sağlanması için besleyici yakıt demetlerinin tamamının hareket ettirilmesi yerine “ThO₂ parmak”larını yani PWR’larda yaygın olarak kullanılan kontrol çubukları benzeri hareketli ThO₂ çubuklarının kullanıldığı bir reaktördür. Üçüncü kor tasarımı yıllık olarak yeniden yakıt yüklenmelerinin gruplar halinde yapıldığı bir reaktördür. Dördüncü tasarım hareketsiz besleyici ve örtü yakıt demetlerinden oluşan ve reaktivite kontrolünün hareketli ThO₂ kontrolü (“thoria shim”) ve PWR reaktörlerindeki yöntemlerle sağlandığı bir tasarımdır. Bu tasarımlardan ilk üçü BAPL tarafından, dördüncü tasarım ise KAPL tarafından geliştirilmiştir.

EK.A.2.1 Ticari Ölçekli LWBR Konsepti

Bu reaktör tasarımı, başarısı ispatlanmış bir teknoloji olan LWBR reaktörünün ticari boyutlara ölçeklendirilmiş halidir. Mekanik tasarımı, reaktivite kontrolü ve yakıt modülü tasarımları LWBR ile tamamen aynıdır. LWBR korunda merkezde bulunan Tip 1 yakıt modülü gibi 141 adet yakıt modülünden oluşur. Bu 141 adet yakıt modülünden en dışarıda olanların etrafını LWBR korundakine benzer şekilde fisil izotopça daha zengin güç düzleştirici örtü yakıt bölgeleri çevreler [33]. Bu güç düzleştirici bölgelerin etrafında da ThO₂ yansıtıcı modüller bulunur.

Reaktör çalışmasına ilk başladığında prebreeder reaktörlerden elde edilen U-233 izotopunca zengin yakıtlarla yüklendiği için reaktörün ilk yakıt döngüsünde FEO en yüksek değerde olacak ve zamanla her yakıt döngüsünden sonra yeniden işlemeye uğrayan yakıtlardaki U-233 oranı azalacak ve U-235 oranı artacaktır [32]. Bunun doğurduğu bir sonuç olarak ve tezin önceki kısımlarında da açıklandığı şekilde üretkenlik

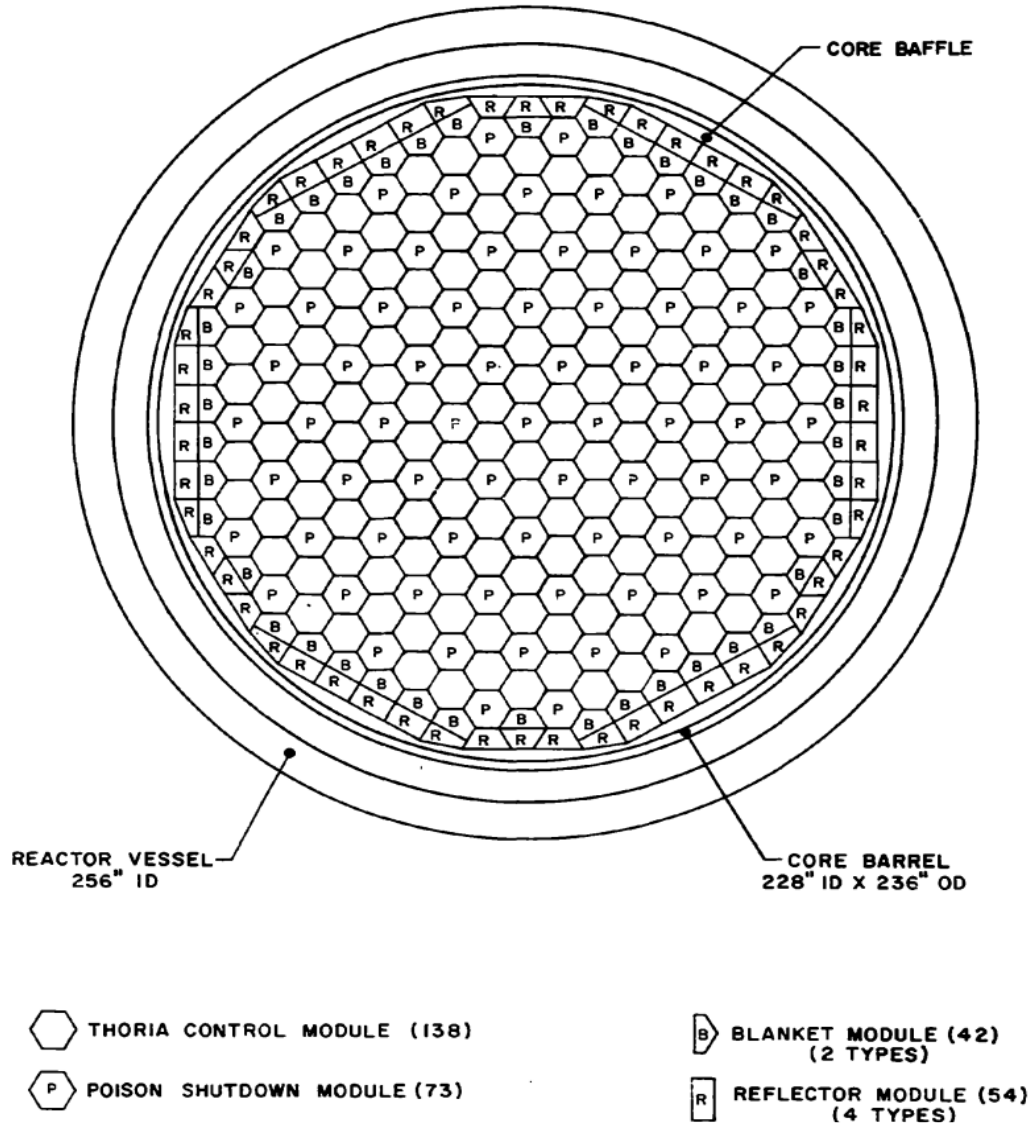
performansı düşecektir. Ticari Ölçekli LWBR reaktörü için ilk yakıt döngüsünde FEO 1,017, denge durumundaki yakıt döngülerinde ise 1,005 olarak hesaplanmıştır [32][33]. Denge durumundaki yakıt döngülerindeki FEO değeri üretkenlik sınırının biraz üstünde olmasına rağmen yakıt yeniden işlemindeki kayıplar %0,5'in altında tutulduğu sürece bu reaktörün kendi fisil izotop ihtiyacını karşıladığı söylenebilir.



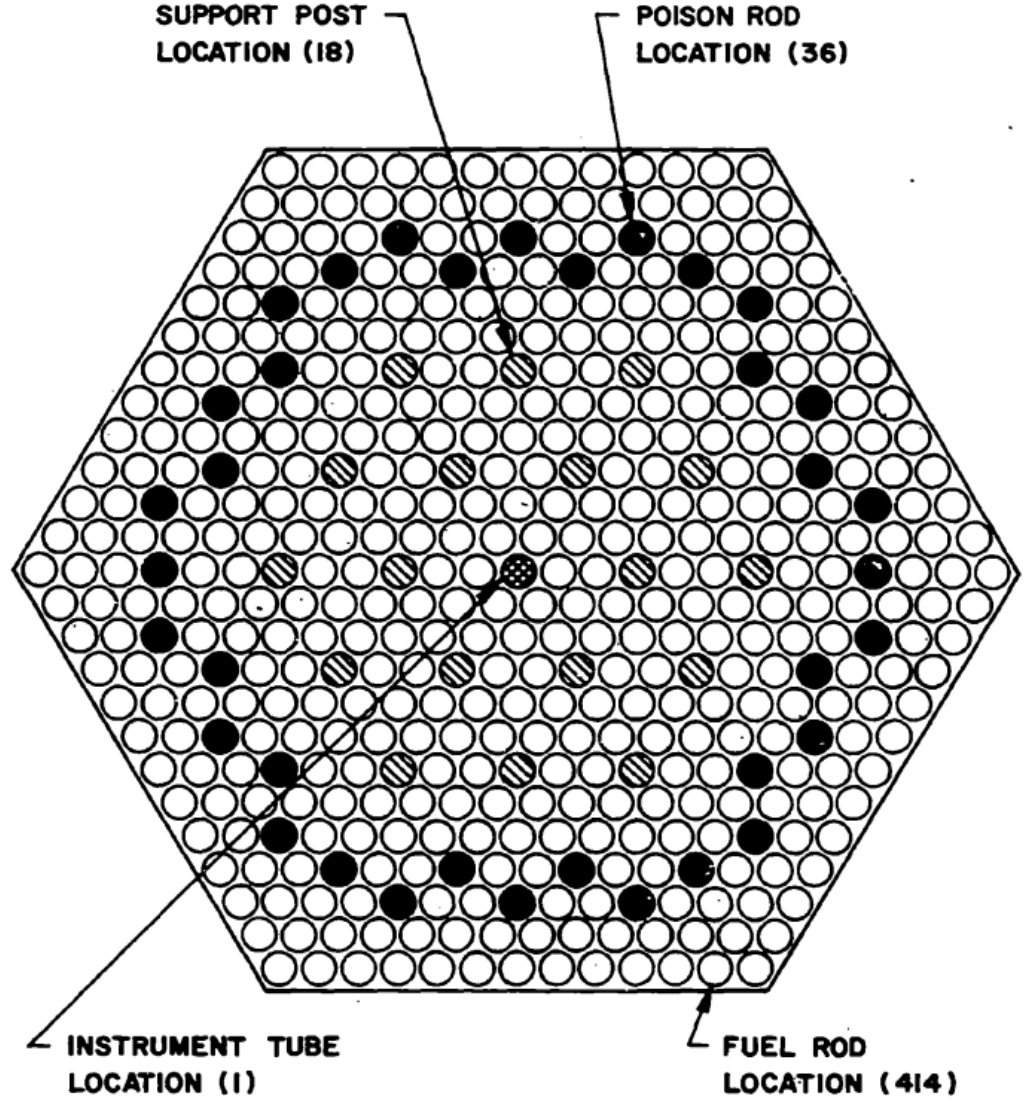
Şekil A.1 Ticari Ölçekte LWBR kor konsepti kesit alan görüntüsü ve modül dizilimi [33]

EK.A.2.2 Hareketli ThO₂ Çubuğu Kontrollü Üretken Reaktör

1000MWe gücündeki bu reaktör tasarımı, 211 adet altıgen yakıt demetinden ve 42 adet yarı altıgen örtü demetinden oluşur [34]. LWBR korunda olduğu gibi eksenel ve radyal yönlerde koru yansıtıcı ThO₂ bölgeleri çevreler. LWBR korundan farklı olarak bu korda destek ızgaralarının malzemesi için Zirkaloy alaşımı seçilmiştir. Reaktörün işletimi sırasında reaktivite kontrolü, çoğu yakıt demetlerinde bulunan ve sayıları 18 olan ThO₂ kontrol çubuklarının eksenel yöndeki hareketi ile sağlanır. Bu yakıt çubukları, çap olarak yakıt çubuklarından daha büyüktür ve dış tarafı ThO₂ iç tarafı zirkonya olan ikili bir yapıdadır. Korda bulunan ve ThO₂ kontrol çubuklarına sahip olmayan yakıt demetleri, bunun yerine PWR korlarında kullanılan kontrol çubuklarına benzer özelliklerde 36 adet hareketli zehir çubuklarına sahiptirler. Bu reaktör için ilk yakıt döngüsünde FEO 1,03, denge durumundaki yakıt çevrimleri içinse 1,01 olarak hesaplanmıştır [32][34].



Şekil A.2 Hareketli ThO₂ Çubuğu Kontroll Üreken Reaktör konsepti için kesit alan görüntüsü ve modül dizilimi [34]



Şekil A.3 Hareketli ThO_2 Çubuğu Kontrollü Üreken Reaktör konsepti için zehir çubuklu yakıt demetlerinin kesit alanı ve yakıt dizilimi [34]

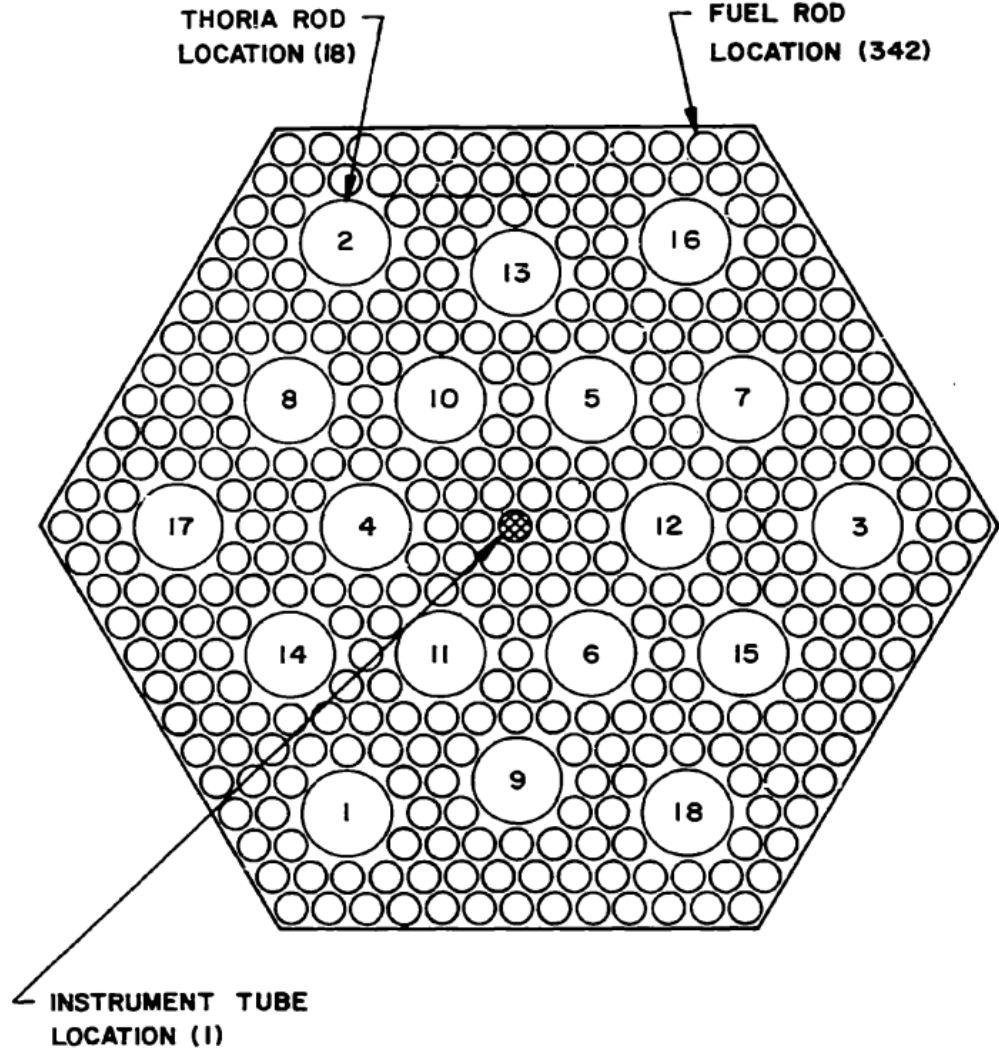


Figure 5
THORIA CONTROL MODULE ROD ARRANGEMENT

Şekil A.4 Hareketli ThO_2 Çubuğu Kontrollü Üreken Reaktör konsepti için ThO_2 kontrol çubuklu yakıt demetlerinin kesit alanı ve yakıt dizilimi [34]

EK.A.2.3 Gelişmiş Hareketli Yakıtlı Üretken Reaktörü

Bu reaktör, Ticari Ölçekli LWBR tasarımı ile benzer bir tasarıma sahiptir. En büyük farkları bu tasarımda besleyici yakıt demetlerinin zirkaloy kılıfları bulunmamaktadır. Kılıfların olmaması, besleyici yakıt demetlerindeki yakıt çubuğu sayısının artmasına ve kordaki yakıt-soğutucu oranının artmasına sebep olur. Böylece reaktörün FEO performansı artar [32]. İki reaktör tasarımı arasındaki diğer farklar da Gelişmiş Hareketli

Yakıtlı Üretken Reaktörde örtü yakıt bölgelerinin besleyici yakıt bölgelerindeki yakıt çubukları ile aynı çapta yakıt çubuklarına sahip olmaları, yani örtü yakıt bölgelerinin büyüklüklerinin Ticari Ölçekli LWBR korundakilere göre daha az yer kaplamasıdır. Ayrıca daha uzun kora sahip olan bu tasarım 1000 MWe'lik güce sahiptir. FEO performansını arttıran bir diğer özellik ise destek ızgaralarının malzemesi olarak Zirkaloy alaşımı seçilmesidir.

Bu tasarım için ilk yakıt döngüsünde FEO 1,03, denge durumundaki FEO değerleri ise 1,01 olarak hesaplanmıştır [32].

EK.A.2.4 Besleyici-Örtü Prebreeder/Üretken Reaktörü

Bu reaktör hem prebreeder hem de üretken reaktör görevinde işletilen bir tasarımdır. LWBR gibi bu tasarım da besleyici ve örtü yakıt bölgelerinden oluşur fakat bu bölgeler hareketli değildirler. Bu tasarımın korunda iki tip yakıt demeti bulunur. Toplam 163 tane yakıt demetine sahip olan bu reaktörde yakıt demetlerindeki besleyici bölgeler Uranyumca zenginleştirilmiş çubuklara sahipken örtü bölgelerinde sadece ThO₂ peletleri bulunur. Yakıt demetlerinden bazıları zehir kontrol çubuklarına sahipken bazıları hareketli ThO₂ çubuk gruplarına sahiptir. EK.A.2.2 başlığında özetlenen kor tasarımındaki ThO₂ kontrol çubuklarının aksine bu kordaki ThO₂ kontrol çubuklarının çapları küçüktür. Bunlarla birlikte soğutucudaki çözülebilir boron miktarı ile de reaktivite kontrolü sağlanır.

Bu tasarım için denge FEO değeri 1,01'dir. Başlangıç yakıt çevriminde reaktör için FEO 0,99 değerinde olup 6. Yakıt çevriminde yaklaşık 1,006 değerine ulaşır [32].

EK B Python Kodu

```
import io
import time
import subprocess
import sys
import math
import signal
import glob
import os.path
import shutil
position=[]

def main(i):
```

```

abd=["serpentOpenMP"+' '+'-omp'+' '+'64'+' '+'LWBR__"+str(i)+'.txt'+' '+'-
noplots"] #Girdi dosyasi adini iceren serpent kosma komutu
cmd="".join(abd)
print(abd)
process = subprocess.Popen(("bash", "-c", cmd),stdout=subprocess.PIPE)

while True:
    output = process.stdout.readline()
    if process.poll() is not None and output == "":
        main.token=1
        path='/fusers/alim/AlimTezNew/LWBR__'+str(i)+'.txt' #Yinelemeler icin girdi
dosyasi konumu
        h99= open(path,"r+")
        lines99 = h99.readlines()
        h99.truncate(0)
        h99.close()
        h99=open(path,"r+")
        for j99 in range(len(lines99)):
            if x12 in lines99[j99]:
                h99.write(lines99[j99].replace(x12,x13))
            else:
                h99.write(lines99[j99])
        break
    if output:
        print (output.strip())
    if x11 in output:
        print (output.strip())
        data = [process.stdout.readline().strip() for i1 in range(20)]
        data2=data[8].strip().split()
        data3=data[16].strip().split()
        print(data)
        print(data2)
        print(data2[2])
        print(data3)
        if float(data2[2]) == 500: # aktif dongu sayisi
            if float(data3[3]) < 0.9975:
                process.terminate()
                print('boi')
                path='/fusers/alim/AlimTezNew/LWBR__'+str(i)+'.txt' #Yinelemeler icin
girdi dosyasi konumu
                h= open(path,"r+")
                lines2 = h.readlines()
                h.truncate(0)
                h.close()
                h= open(path,"r+")
                y='%change position of seed'
                y2='%changing volume'
                for ji in range(len(lines2)):
                    if y in lines2[ji]:
                        b2 = float(lines2[ji].strip().split()[5])

```

```

c2=str(b2)
d2=str(b2+dx)
h.write(lines2[ji].replace(c2,d2))
d2F=float(d2)
volTh=0
volH=0
volL=0
if -132.08 > d2F:
    print("error")
if -25.4 > d2F > -132.08:
    volH=dx*math.pi*((0.32004)**2)*331*12
    volL=dx*math.pi*((0.32004)**2)*288*12
    volTh=0
if 0 > d2F > -25.4:
    volTh=dx*math.pi*((0.32004)**2)*619
    volH=0
    volL=0
if 25.4 > d2F > 0:
    volTh=(-1)*dx*math.pi*((0.32004)**2)*619
    volL=0
    volH=0
if 60.96 > d2F > 25.4:
    volH=(-1)*dx*math.pi*((0.32004)**2)*331*12
    volTh=(-1)*dx*math.pi*((0.32004)**2)*288*12
    volL=0
if 96.52 > d2F > 60.96:
    volH=(-1)*dx*math.pi*((0.32004)**2)*331*12
    volTh=(-1)*dx*math.pi*((0.32004)**2)*222*12
    volL=(-1)*dx*math.pi*((0.32004)**2)*66*12
if 132.08 > d2F > 96.52:
    volH=(-1)*dx*math.pi*((0.32004)**2)*331*12
    volTh=(-1)*dx*math.pi*((0.32004)**2)*150*12
    volL=(-1)*dx*math.pi*((0.32004)**2)*138*12
if d2F > 132.08:
    process.terminate()
    print("error")

elif y2 in lines2[ji]:
    b12a=float(lines2[ji].strip().split()[2])
    b12a2=lines2[ji].strip().split()[2]
    b12b=float(lines2[ji+1].strip().split()[2])
    b12b2=lines2[ji+1].strip().split()[2]
    b12c=float(lines2[ji+2].strip().split()[2])
    b12c2=lines2[ji+2].strip().split()[2]
    c12a=str(b12a)
    c12b=str(b12b)
    c12c=str(b12c)
    d1a=str(float(b12a+volTh))

```

```

        d1b=str(float(b12b+volH))
        d1c=str(float(b12c+volL))
        h.write(lines2[ji].replace(b12a2,d1a))
        h.write(lines2[ji+1].replace(b12b2,d1b))
        h.write(lines2[ji+2].replace(b12c2,d1c))

    elif x4 in lines2[ji]:
        taken_useless=1

    else:
        h.write(lines2[ji])

h.close()
main(i)

if float(data3[3]) > 1.0025:
    process.terminate()
    path='/fusers/alim/AlimTezNew/LWBR__'+str(i)+'.txt'
    h= open(path,"r+")
    lines2 = h.readlines()
    h.truncate(0)
    h.close()
    h= open(path,"r+")
    y='%change position of seed'
    y2='%changing volume'
    for j in range(len(lines2)):
        if y in lines2[j]:
            b = float(lines2[j].strip().split()[5])
            c=str(b)
            d=str(b-dx)
            lines2[j].replace(c,d)
            h.write(lines2[j].replace(c,d))
            dF=float(d)
            volTh=0
            volH=0
            volL=0
            if -132.08 > dF :
                print("went too far")
            if -25.4 > dF > -132.08:
                volH=(-1)*dx*math.pi*((0.32004)**2)*331*12
                volL=(-1)*dx*math.pi*((0.32004)**2)*288*12
                volTh=0
            if 0 > dF > -25.4:
                volTh=(-1)*dx*math.pi*((0.32004)**2)*619
                volH=0
                volL=0
            if 25.4 > dF > 0:

```

```

        volTh=dx*math.pi*((0.32004)**2)*619
        volH=0
        volL=0
    if 60.96 > dF > 25.4:
        volH=dx*math.pi*((0.32004)**2)*331*12
        volTh=dx*math.pi*((0.32004)**2)*288*12
    if 96.52 > dF > 60.96:
        volH=dx*math.pi*((0.32004)**2)*331*12
        volTh=dx*math.pi*((0.32004)**2)*222*12
        volL=dx*math.pi*((0.32004)**2)*66*12
    if 132.08 > dF > 96.52:
        volH=dx*math.pi*((0.32004)**2)*331*12
        volTh=dx*math.pi*((0.32004)**2)*150*12
        volL=dx*math.pi*((0.32004)**2)*138*12
    if dF > 132.08:
        print("error")

elif y2 in lines2[j]:
    b12a=float(lines2[j].strip().split()[2])
    b12b=float(lines2[j+1].strip().split()[2])
    b12c=float(lines2[j+2].strip().split()[2])
    c12a=str(b12a)
    c12b=str(b12b)
    c12c=str(b12c)
    d1a=str(float(b12a+volTh))
    d1b=str(float(b12b+volH))
    d1c=str(float(b12c+volL))
    # lines2[j].replace(c12a,d1a)
    # lines2[j+1].replace(c12b,d1b)
    # lines2[j+2].replace(c12c,d1c)
    h.write(lines2[j].replace(c12a,d1a))
    h.write(lines2[j+1].replace(c12b,d1b))
    h.write(lines2[j+2].replace(c12c,d1c))

elif x4 in lines2[j]:
    token_useless=1

else:
    h.write(lines2[j])

h.close()
main(i)

retval = process.poll()

for i in range (1,250):

```

```

x = 'Processing data for'
x2='Transport calculation: step = 1 / 2 (corrector 2/2)'
x3='Active cycle 200 / 200'
x4='%skip%'
x5='%changeresetday%'
x6='%change day%'
x7='%sourceinput'
x8='%sourceoutput'
x9='%wrkin'
x10='%wrkout'
x11a='Input file: '+ "LWBR__"+str(i)+".txt"
abd33=["Input"+' '+ "file:"+' '+ "LWBR__"+str(i)+".txt"]
x11="Serpent 2.1.30 -- Static criticality source simulation"
print(x11a)
print(x11)
x12='%dep'
x13='dep'
volTh=0
volH=0
volL=0
dx=0.635 #yinelemeler arasi hareket miktarı
dt=20
checkVol=[]
checkVolb=[]
checkVol2=[]
checkVol2b=[]

main(str(i))

if main.token == 1:
    abd=["cd"+' '+ "AlimTezNew/"+' '+ "&&"+' '+ "serpentOpenMP"+' '+ "-omp"+' '+ "64"+' '+ "LWBR__"+str(i)+".txt"+' '+ "-noplot"]
    cmd=" ".join(abd)
    print(abd)
    process2 = subprocess.Popen(("bash", "-c", cmd),stdout=subprocess.PIPE)
    #process2 = subprocess.Popen(("serpentOpenMP", "-omp", "64",
    "LWBRFullDivThr.txt", "-noplot"),stdout=subprocess.PIPE)
    while True:
        output2 = process2.stdout.readline()
        if process2.poll() is not None and output2 == "":
            break
        if output2:
            print (output2.strip())

    token=[]

    src="/fusers/alim/AlimTezNew/"+"LWBR__"+str(i)+".txt"
    dst="/fusers/alim/AlimTezNew/"+"LWBR__"+str(i+1)+".txt"

```

```
shutil.copyfile(src, dst)
```

```
path='/fusers/alim/AlimTezNew/LWBR__'+str(i+1)+'.txt'
h5= open(path,"r+")
lines6 = h5.readlines()
h5.truncate(0)
h5.close()
h5= open(path,"r+")
y='%change position of seed'
y2='%changing volume'
for j5 in range(len(lines6)):
    if x5 in lines6[j5]:
        db = float(lines6[j5].strip().split()[3])
        db2 = str(db)
        istr=str(db+dt)

        s4b = (lines6[j5].strip().split()[4])
        s4b2 = list(s4b)
        s4b3 = int(s4b2[8])
        s4b4=1+s4b3
        s4b2[8]=str(s4b4)
        s4b5=[]
        s4b2="".join(s4b2)
        h5.write(lines6[j5].replace(s4b,s4b2).replace(db2, istr))

    elif x6 in lines6[j5]:
        dba = float(lines6[j5].strip().split()[2])
        db2a = str(dba)
        istr2a=str(dba+dt)
        h5.write(lines6[j5].replace(db2a,istr2a).replace(x13,x12))
    elif y in lines6[j5]:
        b2e = float(lines6[j5].strip().split()[5])
        c2e=str(b2e)
        position.append(c2e)
        h5.write(lines6[j5])
    elif x7 in lines6[j5]:
        s1b = (lines6[j5].strip().split()[3])
        s1b2 = list(s1b)
        s1b3 = int(s1b2[8])
        s1b4=1+s1b3
        s1b2[8]=str(s1b4)
        s1b5=[]
        s1b2="".join(s1b2)
        print(s1b2)
        h5.write(lines6[j5].replace(s1b,s1b2))
    elif x8 in lines6[j5]:
        s2b = (lines6[j5].strip().split()[2])
        s2b2 = list(s2b)
```

```
s2b3 = int(s2b2[8])
s2b4=1+s2b3
s2b2[8]=str(s2b4)
s2b5=[]
s2b2="".join(s2b2)
print(s2b2)
h5.write(lines6[j5].replace(s2b,s2b2))
elif x9 in lines6[j5]:
    s3b = (lines6[j5].strip().split())[3]
    s3b2 = list(s3b)
    s3b3 = int(s3b2[8])
    s3b4=1+s3b3
    s3b2[8]=str(s3b4)
    s3b5=[]
    s3b2="".join(s3b2)
    print(s3b2)
    h5.write(lines6[j5].replace(s3b,s3b2))

else:
    h5.write(lines6[j5])

h5.close()

print(position)

#Serpent girdi dosyalarına http://www.openaccess.hacettepe.edu.tr:8080/xmlui/
adresinden ulaşılabilir.
```