T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SU ALTI AKUSTİK DALGA YAYILIMI PROBLEMİNİN ZAMAN UZAYI SONLU FARKLAR YÖNTEMİ İLE GPU ÜZERİNDE ÇÖZÜMÜ

YUNUS EMRE ALÇAKAKAN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2019

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SU ALTI AKUSTİK DALGA YAYILIMI PROBLEMİNİN ZAMAN UZAYI SONLU FARKLAR YÖNTEMİ İLE GPU ÜZERİNDE ÇÖZÜMÜ

YUNUS EMRE ALÇAKAKAN

YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
PROF. DR. SERKAN AKSOY

GEBZE
2019

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SOLUTION FOR UNDERWATER ACOUSTIC WAVE PROPAGATION PROBLEM WITH FINITE DIFFERENCE TIME DOMAIN METHOD ON GPU

YUNUS EMRE ALÇAKAKAN

A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRONIC ENGINEERING

THESIS SUPERVISOR

PROF. DR. SERKAN AKSOY

GEBZE

2019

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 03/07/2019 tarih ve 2019/30 sayılı kararıyla oluşturulan jüri tarafından 02/08/2019 tarihinde tez savunma sınavı yapılan Yunus Emre ALÇAKAKAN'ın tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : PROF. DR. SERKAN AKSOY

ÜYE : PROF. DR. OLEG TRETYAKOV

ÜYE : DR. ÖĞR. ÜYESİ ERKUL BAŞARAN

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

Bu tez çalışmasında iki boyutlu su altı akustik dalga yayılım problemi Zaman Uzayı Sonlu Farklar (ZUSF) yöntemi kullanılarak Grafik İşlemci Ünitesi (GPU) üzerinde çözülmüştür. Problem olarak büyük ölçekli bir su altı akustik uygulaması olan SOFAR kanalı seçilmiştir. Kaynak olarak sürekli sinusoidal dalga kullanılmış, problem uzayının gerekli dış sınırları Mur tipi soğurucu sınır koşulu uygulanarak sonlandırılmıştır. SOFAR kanalı problemi birinci aşamada ZUSF yöntemiyle MATLAB ortamında nümerik olarak çözülmüştür. İkinci aşamada, problem CUDA Uygulama Programlama Arayüzü (API) ile GPU üzerinde C/C++ programlama dili kullanılarak ve paralelleştirilerek çözülmüştür. Karşılaştırmalı elde edilen sonuçlara göre, yaklaşık 24 kat hızlanma elde edilmiştir.

Anahtar Kelimeler: Zaman Uzayında Sonlu Farklar (ZUSF), GPU programlama, CUDA, SOFAR kanalı.

SUMMARY

In this work, underwater acoustic wave propagation problem in two-dimensional space is solved with Finite Difference Time Domain (FDTD) method on Graphics Processing Unit (GPU). The SOFAR channel is chosen as a large-scale underwater acoustic application. A continuous sinusoidal wave is used as a source; the necessary outer boundaries of the problem space are terminated by Mur type absorbing boundary condition. In the first stage, the SOFAR channel problem is solved numerically by the FDTD method in MATLAB. In the second stage, the problem is solved in parallel by using the CUDA Application Programming Interface (API) with C/C++ programming language on the GPU. Almost 24 times speedup is observed according to the compared results.

Key Words: Finite Difference Time Domain (FDTD), GPU programming, CUDA, SOFAR channel.

TEŞEKKÜR

Yüksek lisans eğitimim boyunca bilgi ve tecrübelerini aktaran ve tezin gerçekleşmesi için gerekli laboratuvar altyapısı desteğini sunan danışmanım Prof. Dr. Serkan AKSOY'a ve hayatımın her alanında beni destekleyen aileme en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

	Sayfa
ÖZET	v
SUMMARY	vi
TEŞEKKÜR	vii
İÇİNDEKİLER	viii
SİMGELER ve KISALTMALAR DİZİNİ	x
ŞEKİLLER DİZİNİ	xi
TABLOLAR DİZİNİ	xiii
1. GİRİŞ	1
1.1. Akustik Dalga Denklemi	2
1.2. SOFAR Kanalı	4
1.2.1. SOFAR Kanalında İletim Kaybı	6
2. ZAMAN UZAYI SONLU FARKLAR YÖNTEMİ	9
2.1. Sonlu Farklar Yöntemi	9
2.2. Güncelleme Denkleminin Çıkarılması	11
2.3. Sayısal Dispersiyon	13
2.4. Sayısal Kararlılık	14
3. GPU PROGRAMLAMA	15
3.1. CUDA ile GPU Programlama	17
3.1.1. CUDA Programlama Temelleri	19
3.1.1.1. Kernel	19
3.1.1.2. İş Parçacığı Hiyerarşisi	19
3.1.2. CUDA Program Akışı	22
3.1.3. Hafıza Türleri	23
3.2 GPU ile ZUSF Uygulamaları	24
3.2.1. İki Boyutlu Uygulamalar	24
3.2.2. Üç Boyutlu Uygulamalar	25
4. SAYISAL SONUÇLAR	28
4.1. MATLAB ve GPU Çözümünün Doğrulanması	28

4.1.1. MATLAB Sonuçları 30
4.1.2. CUDA ile Paralelleştirme Sonuçları 37
4.1.3. Literatüre Göre Sonuçların Doğrulanması 46
4.2. Farklı Frekanslar için GPU Çözümü Sonuçları 46
4.2.1. 1500 Hz'lik Kaynak İşaret 47
4.2.2. 1940 Hz'lik Kaynak İşaret 50
4.2.3. Performans Değerlerinin Karşılaştırılması 53
5. SONUÇLAR ve GELECEK ÖNERİLERİ 55

KAYNAKLAR 56
ÖZGEÇMİŞ 58

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
∇	:	Nabla operatörü
Hz	:	Hertz
km	:	Kilometre
m	:	Metre
sn	:	Saniye
RAM	:	Random Access Memory
VRAM	:	Video RAM
GPU	:	Graphics Processing Unit
ZUSF	:	Zaman Uzayı Sonlu Farklar
ALU	:	Arithmetic Logic Unit
DRAM	:	Dynamic Random Access Memory
OpenCL	:	Open Computing Language
CUDA	:	Compute Unified Device Architecture

ŞEKİLLER DİZİNİ

Şekil No:		Sayfa
1.1:	Ses hızı profili.	2
1.2:	SOFAR kanalı.	5
1.3:	Nümerik olarak elde edilen SOFAR kanalı.	5
1.4:	Farklı derinliklerde ses hızı profili.	6
1.5:	SOFAR kanalının başlangıcından 10 m derinlikte yayılım kaybı grafiği	7
1.6:	Ses hızının minimum olduğu 150 m derinlikte iletim kaybı.	7
1.7:	SOFAR kanalının sonlandığı 1000 m derinlikte iletim kaybı grafiği.	8
3.1:	CPU ile ekran kartının hesaplama gücü karşılaştırması.	16
3.2:	CPU-GPU temel mimari yaklaşımı.	17
3.3:	GPU'da ölçeklenebilirlik.	18
3.4:	Izgara üzerindeki blok ve iş parçacığı hiyerarşisi.	20
3.5:	Örnek CUDA kodu.	21
3.6:	CUDA'da örnek iş akışı.	22
3.7:	CUDA bellek erişim yapısı.	23
4.1:	Tezde kullanılan ses hızı profili.	28
4.2:	MATLAB ile hesaplanan SOFAR kanalı alan dağılımı grafiği.	30
4.3:	MATLAB ile 20 m derinlik ve 2311 m menzil için hesaplanan işaret.	31
4.4:	MATLAB ile 20 m derinlik ve 4622 m menzil için hesaplanan işaret.	31
4.5:	MATLAB ile 20 m derinlik ve 9244 m menzil için hesaplanan işaret.	32
4.6:	MATLAB ile 50 m derinlik ve 2311 m menzil için hesaplanan işaret.	32
4.7:	MATLAB ile 50 m derinlik ve 4622 m menzil için hesaplanan işaret.	33
4.8:	MATLAB ile 50 m derinlik ve 9244 m menzil için hesaplanan işaret.	33
4.9:	MATLAB ile 100 m derinlik ve 2311 m menzil için hesaplanan işaret.	34
4.10:	MATLAB ile 100 m derinlik ve 4622 m menzil için hesaplanan işaret.	34
4.11:	MATLAB ile 100 m derinlik ve 9244 m menzil için hesaplanan işaret.	35

4.12:	MATLAB ile 20 m derinlikte iletim kaybı.	36
4.13:	MATLAB ile 50 m derinlikte iletim kaybı.	36
4.14:	MATLAB ile 100 m derinlikte iletim kaybı.	37
4.15:	CUDA ile paralel hesaplanan alan dağılımı grafiği..	38
4.16:	CUDA ile 20 m derinlik ve 2311 m menzil için hesaplanan işaret.	39
4.17:	CUDA ile 20 m derinlik ve 4622 m menzil için hesaplanan işaret.	39
4.18:	CUDA ile 20 m derinlik ve 9244 m menzil için hesaplanan işaret.	40
4.19:	CUDA ile 50 m derinlik ve 2311 m menzil için hesaplanan işaret.	41
4.20:	CUDA ile 50 m derinlik ve 4622 m menzil için hesaplanan işaret.	41
4.21:	CUDA ile 50 m derinlik ve 9244 m menzil için hesaplanan işaret.	42
4.22:	CUDA ile 100 m derinlik ve 2311 m menzil için hesaplanan işaret.	42
4.23:	CUDA ile 100 m derinlik ve 4622 m menzil için hesaplanan işaret.	43
4.24:	CUDA ile 100 m derinlik ve 9244 m menzil için hesaplanan işaret.	43
4.25:	CUDA ile 20 m derinlikte iletim kaybı.	44
4.26:	CUDA ile 50 m derinlikte iletim kaybı.	45
4.27:	CUDA ile 100 m derinlikte iletim kaybı.	45
4.28:	CUDA ile 1500 Hz kaynak frekansı için SOFAR alan dağılımı.	47
4.29:	CUDA ile 1500 Hz kaynak frekansı için 20 m derinlikte iletim kaybı.	48
4.30:	CUDA ile 1500 Hz kaynak frekansı için 50 m derinlikte iletim kaybı.	49
4.31:	CUDA ile 1500 Hz kaynak frekansı için 100 m derinlikte iletim kaybı.	50
4.32:	CUDA ile 1940 Hz kaynak frekansı için SOFAR alan dağılımı.	51
4.33:	CUDA ile 1940 Hz kaynak frekansı için 20 m derinlikte iletim kaybı.	52
4.34:	CUDA ile 1940 Hz kaynak frekansı için 50 m derinlikte iletim kaybı.	52
4.35:	CUDA ile 1940 Hz kaynak frekansı için 100 m derinlikte iletim kaybı.	53

TABLOLAR DİZİNİ

Tablo No:		**Sayfa**
4.1:	Süre ve hafıza kullanımı sonuçları.	46
4.2:	Farklı frekans değerleri için ölçüm sonuçları.	53

1. GİRİŞ

Su altı akustik dalga yayılımı problemi özellikle son 30 yıldır ticari, bilimsel ve askeri amaçlar için üzerinde çalışılan bir konudur. Su altında akustik dalga yayılımının nasıl gerçekleştiği su altı haberleşme sistemlerinin kurulması, SONAR sistem performansının incelenmesi vb. konular için bilinmesi gerekmektedir. Geliştirilecek sistemler için açık denizde deney yapmadan önce, bilgisayar ortamında analiz yapabilmek amacıyla su altı akustik dalga yayılımını düşük ve yüksek frekanslı problemler için makul sürelerde çözebilen bilgisayar benzetiminin yapılması ihtiyacı vardır. Bu tez çalışması kapsamında bu ihtiyacı karşılayacak bir çözüm hedeflenmiştir.

Su altında ses dalgası yaklaşık 1500 m/sn hızla hareket etmektedir. Bu ses hızı sıcaklık, basınç, denizin tuzluluk oranı değişkenlerine bağlı olarak değişmektedir. 1974 yılında W. H. Munk tarafından ses hızının derinliğe göre denizdeki değişim oranını ifade eden denklem [Munk, 1974]

$$c = c_{minimum}[1 + \varepsilon(\eta + e^{-\eta} - 1)] \tag{1.1}$$

burada

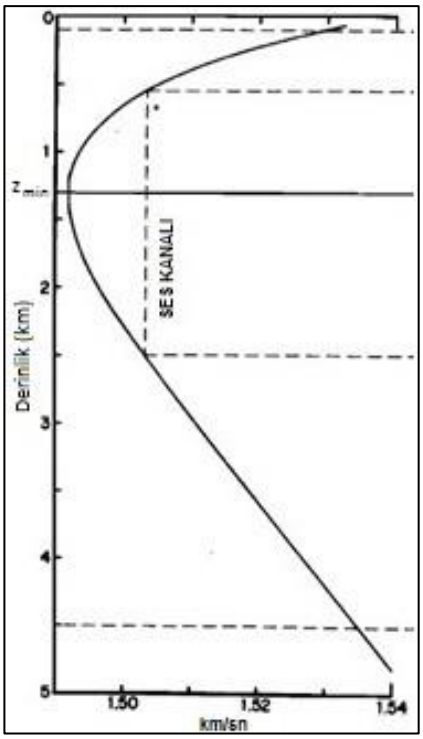
$$\varepsilon = 7.37 \times 10^{-3} \tag{1.2}$$

$$\eta = \frac{2(z - z_{minimum})}{z_{minimum}} \tag{1.3}$$

şeklinde verilmiştir.

Bu denklemlerde c ses hızını, $c_{minimum}$, sesin minimum hızını, $z_{minimum}$, ses hızının minimum olduğu derinliği, ε perturbasyon sabitini ifade etmektedir.

Denklem (1.1)'e göre ses hızının derinliğe göre değişimi (ses hızı profili) Şekil 1.1'de verilmiştir.



Şekil 1.1: Ses hızı profili.

Ses hızı profilinin bu değişimi su altında yayılan dalganın da izlediği yolu etkilemektedir. Şekil 1.1'de ses hızının en yavaş olduğu $z_{minimum}$ noktasından özel bir yayılım kanalı ortaya çıkmaktadır. Bu tez kapsamında incelenen problemde de bu özel yayılım kanalı olan SOFAR kanalının çözümü incelenmiştir.

1.1. Akustik Dalga Denklemi

Akışkanlar mekaniğinde kullanılan bazı temel denklemlerin doğrusallaştırılması ile elde edilen denklemler su altı akustik dalga yayılımını modellemek için kullanılmaktadır. Bu denklemler Durum, Süreklilik ve Euler denklemleridir. Durum denklemi

$$u(\vec{r},t) = \beta s(\vec{r},t) \tag{1.4}$$

olarak ifade edilir. Burada $u(\vec{r},t)$ akustik basınç Newton/m^2, $s(\vec{r},t)$ yoğunlaşma (birimsiz), β esneklik (hacim katsayısı) Newton/m^2 olarak verilir.

Belirli bir bölgede hareket eden bir su kütlesi için parçacık hızı ($\vec{v}(\vec{r},t)$) ve su yoğunlaşması arasındaki ilişkiyi gösteren Lineerleştirilmiş Süreklilik Denklemi ise

$$\frac{\partial s(\vec{r},t)}{\partial t} + \nabla \cdot \vec{v}(\vec{r},t) = 0 \tag{1.5}$$

olarak verilir.

Akustik basınç dağılımı ve parçacık hızı arasındaki ilişkiyi gösteren Lineerleştirilmiş Euler denklemi ise

$$\rho(\vec{r})\frac{\partial \vec{v}(\vec{r},t)}{\partial t} + \nabla u(\vec{r},t) = 0 \tag{1.6}$$

olarak gösterilir. Denklemde yer alan $\rho(\vec{r})$ kg/m^3 su yoğunluğunu ifade etmektedir.

Akustik dalga denklemini çıkartmak için Lineerleştirilmiş Euler denkleminin diverjansı alınır

$$\begin{aligned} &\nabla \cdot \left[\rho(\vec{r})\frac{\partial \vec{v}(\vec{r},t)}{\partial t}\right] + \nabla \cdot \nabla u(\vec{r},t) = 0 \\ &\Rightarrow \nabla \cdot \left[\rho(\vec{r})\frac{\partial \vec{v}(\vec{r},t)}{\partial t}\right] + \Delta u(\vec{r},t) = 0 \end{aligned} \tag{1.7}$$

$\rho(\vec{r})$'nin konuma göre yavaş değiştiği varsayımı ve $\nabla \cdot (\alpha\vec{A}) = \nabla\alpha \cdot \vec{A} + \alpha\nabla \cdot \vec{A}$ bağıntısı kullanılarak

$$\left[\nabla\rho(\vec{r})\frac{\partial \vec{v}(\vec{r},t)}{\partial t} + \rho(\vec{r})\nabla \cdot \frac{\partial \vec{v}(\vec{r},t)}{\partial t}\right] + \nabla u(\vec{r},t) = 0 \tag{1.8}$$

denklemi bulunur. Lineerleştirilmiş Euler denkleminden $\partial\vec{v}(\vec{r},t)/\partial t$ çekilerek yukarıdaki denklemde yerine konulursa

$$\left[-\nabla\rho(\vec{r})\frac{\nabla u(\vec{r},t)}{\rho(\vec{r})} + \rho(\vec{r})\nabla \cdot \frac{\partial \vec{v}(\vec{r},t)}{\partial t}\right] + \nabla u(\vec{r},t) = 0 \tag{1.9}$$

bulunur. Lineerleştirilmiş Süreklilik denkleminin zaman türevi alınıp Durum denkleminden $s(\vec{r},t) = u(\vec{r},t)/\beta$ ilişkisi kullanılıp $\nabla \cdot \partial\vec{v}(\vec{r},t)/\partial t$ çekilip yukarıdaki denklemde yerine konulursa

$$\left[-\nabla\rho(\vec{r})\frac{\nabla u(\vec{r},t)}{\rho(\vec{r})} - \frac{\rho(\vec{r})}{\beta}\frac{\partial^2\vec{u}(\vec{r},t)}{\partial t^2}\right] + \nabla\vec{u}(\vec{r},t) = 0 \tag{1.10}$$

bulunur. Yukarıdaki bağıntı düzenlenip $c(\vec{r}) = \sqrt{\beta/\rho(\vec{r})}$ konuma bağlı ses hızı olmak üzere

$$\nabla^2 u(\vec{r},t) - \nabla\rho(\vec{r})\left[\frac{1}{\rho(\vec{r})}\right]\nabla u(\vec{r},t) - \frac{1}{c^2(\vec{r})}\frac{\partial^2 u(\vec{r},t)}{\partial t^2} = 0 \tag{1.11}$$

bulunur. Burada $a = \rho(\vec{r})$, $b = \rho(\vec{r})^{-1}$, $A = \nabla u(\vec{r},t)$ ve $\psi = 1/\rho(\vec{r})$ seçilerek $\nabla ab = a\nabla b + b\nabla a$ ve $\nabla \cdot \psi A = A \cdot \nabla\psi + \psi\nabla \cdot A$ bağıntısı ile akustik dalga denklemi

$$\rho(\vec{r})\nabla \cdot \left[\frac{1}{\rho(\vec{r})}\nabla u(\vec{r},t)\right] - \frac{1}{c^2(\vec{r})}\frac{\partial^2 u(\vec{r},t)}{\partial^2 t^2} = 0 \tag{1.12}$$

olarak bulunur.

Su altı akustiğinde ρ suyun kimyasal yapısı ile çok az değişirken β sıcaklık ve basınçla değişir. β'deki değişimler ρ'ye göre daha önemli olduğundan, ρ sabit olarak düşünülüp akustik dalga denklemi

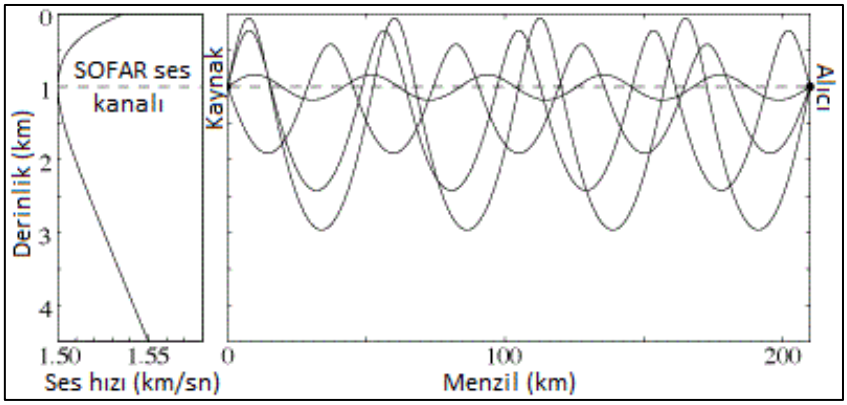
$$\Delta u(\vec{r},t) - \frac{1}{c^2(\vec{r})}\frac{\partial^2 u(\vec{r},t)}{\partial t^2} = 0 \tag{1.13}$$

olarak elde edilir.

1.2. SOFAR Kanalı

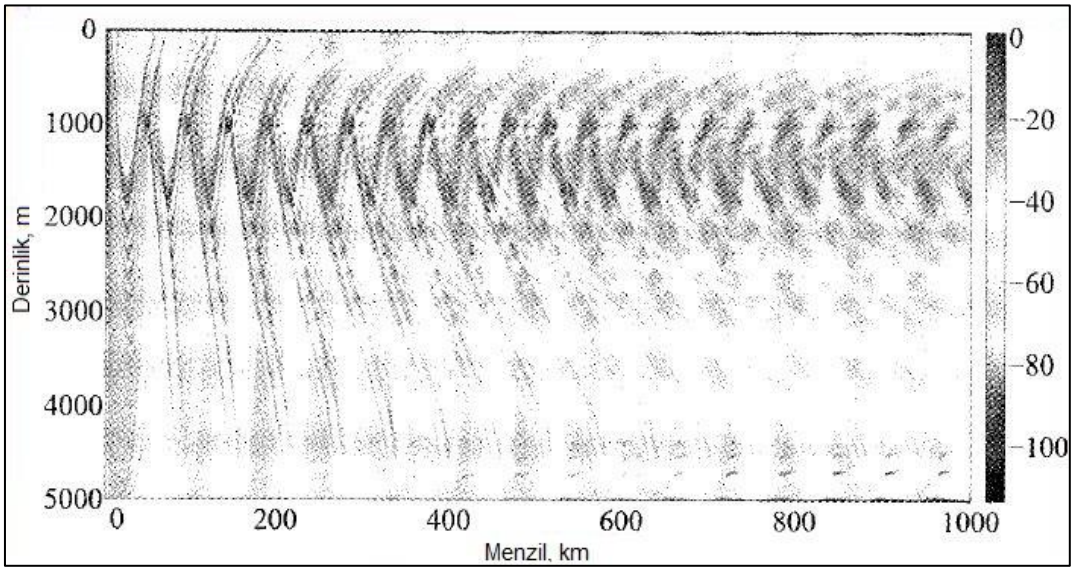
Ses hızının minimum olduğu noktada yayılan akustik dalgalar yayılım kayıpları çok küçük olduğundan çok uzak mesafelere iletilebilir. Dalga kılavuzu olarak işlem gören bu kanaldaki dalgaların 1150 km uzaklığa kadar iletildiği

deneysel olarak gösterilmiştir [Kutschale, 1961]. Çok uzak mesafelere işaret ulaştığı için deniz altı haberleşmesi için SOFAR önemli bir kanaldır. Şekil 1.2'de ses hızına göre SOFAR kanalı verilmiştir [Munk, 1974].



Şekil 1.2: SOFAR kanalı.

ZUSF yöntemiyle GPU üzerinde SOFAR kanalının nümerik olarak elde edilmesiyle ilgili yapılan çalışmada SOFAR kanalı görüntüsü Şekil 1.3'te verilmiştir [Nakai et al, 2001].

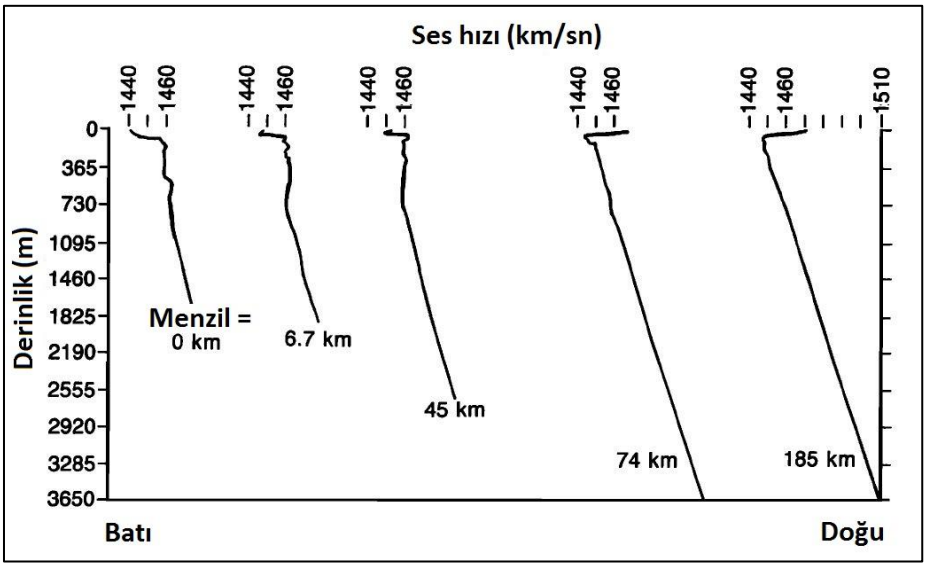


Şekil 1.3: Nümerik olarak elde edilen SOFAR kanalı.

Buna göre söz konusu çalışmada 32 adet GPU kullanılarak elde edilen sonucun Şekil 1.2'de verilen SOFAR kanalı ile uyumlu olduğu gözükmektedir.

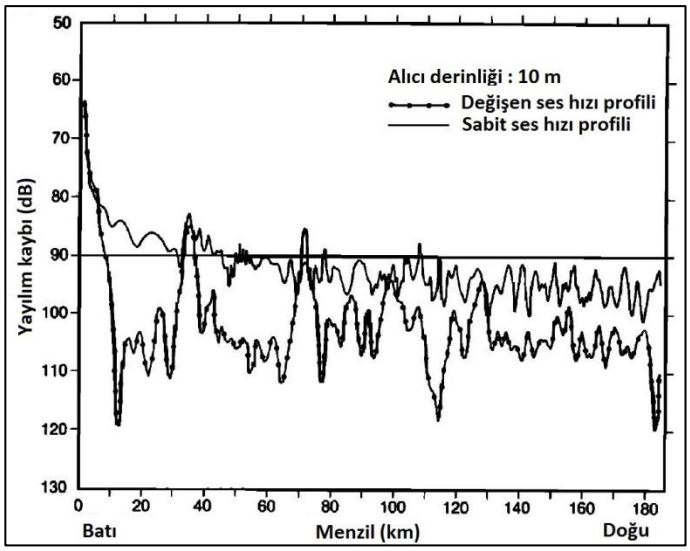
1.2.1. SOFAR Kanalında İletim Kaybı

İletim kaybının bilinmesi SOFAR kanalının kullanılabilirliği açısından önemlidir. Bunun için SOFAR kanalının üst kısmındaki başlangıcından, ses hızının minimum olduğunu noktadan ve kanalın alt tarafta sonlandığı noktadan alınan yayılım kaybı sonuçlarının incelenmesi gereklidir. Sonuçlar bölümünde elde edilen yayılım kaybı grafiklerini yorumlayabilmek için Grönland Denizi'nde yapılan çalışma referans alınmıştır [Mellberg et al., 1991]. Söz konusu çalışmada Grönland Denizi için farklı menzillerindeki ses hızı profilleri Şekil 1.4'te verilmiştir.



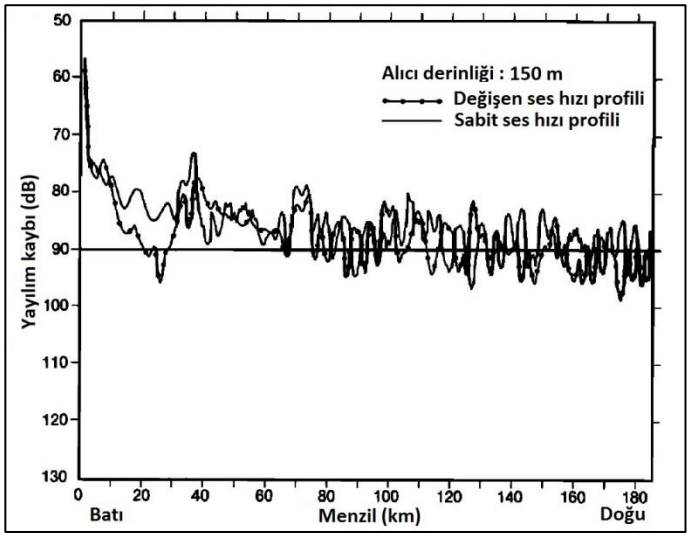
Şekil 1.4: Farklı derinliklerde ses hızı profili.

Buna göre SOFAR kanalının başladığı nokta olan 10 m derinlikte elde edilen yayılım kaybı grafiği Şekil 1.5'te verilmiştir. Yayılım kaybı grafiğinde yer alan değişen ses hızı profili ifadesi, Şekil 1.4'te menzile göre değişen ses hızı profili durumundaki yayılım kaybını; sabit ses hızı profili ise Şekil 1.4'te menzilin sıfır km olduğu ses hızı profilinin tüm uzay boyunca sabit olduğu durumu ifade etmektedir.



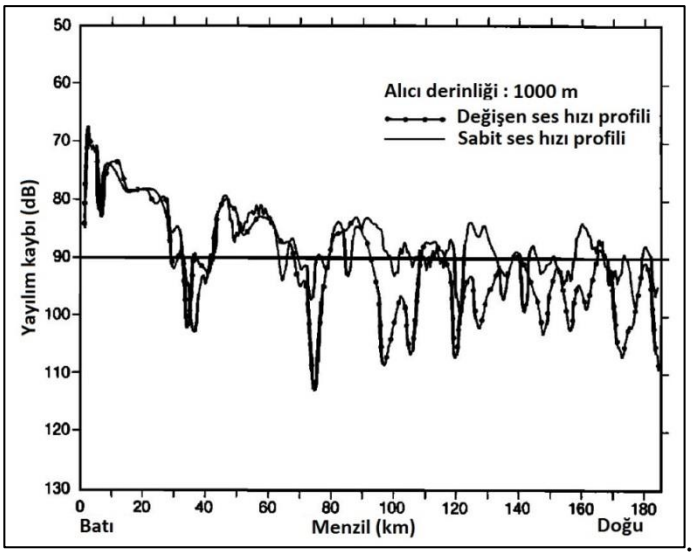
Şekil 1.5: SOFAR kanalının başlangıcından 10 m derinlikte yayılım kaybı.

Yine SOFAR kanalında ses hızının minimum olduğu 150 m derinlikteki yayılım kaybı grafiği Şekil 1.6'da verilmiştir.



Şekil 1.6: Ses hızının minimum olduğu 150 m derinlikte yayılım kaybı.

SOFAR kanalının sonlandığı 1000 m derinlikte yayılım kaybı grafiği Şekil 1.7'de verilmiştir.



Şekil 1.7: SOFAR kanalının sonlandığı 1000 m derinlikte yayılım kaybı grafiği.

2. ZAMAN UZAYI SONLU FARKLAR YÖNTEMİ

Zaman Uzayı Sonlu Farklar, ZUSF (Finite Difference Time Domain, FDTD) yöntemi analitik türevlerin yerine sayısal türev eşdeğerleri konularak zamanda iterasyona dayalı çözüm sağlayan bir yöntemdir. Bilinmeyen büyüklüklerin Taylor serisine açılmasıyla yapılan iterasyon boyunca yakınsak sayısal sonuç elde edilir. ZUSF yönteminin diğer yöntemlerde kullanılan matris denklemi çözmek yerine güncelleme denkleminin zamanda ve uzayda iteratif olarak çözümü gerçekleştirmesi önemli bir avantajdır.

2.1. Sonlu Farklar Yöntemi

Diferansiyel denklem çözümü için analitik yöntemlerin uygulanmasının zor olduğu problemlerde sayısal çözüm sunan yöntemlerden biri olan Sonlu Farklar (Finite Differences) yöntemi çözülmek istenen fonksiyonun ayrıklaştırılması üzerine kuruludur. Sürekli bir $f(x)$ fonksiyonunun i tane Δx aralıklarla eşit olarak bölünmesi

$$f(x) = f(x_i) = f(i\Delta x) \tag{2.1}$$

şeklinde gösterilir. Ayrık hale getirilen $f(x)$ fonksiyonunun $x = x_i$ noktasında Taylor serisine açılımı

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n(x_i)}{n!} = f(x)|_{x=x_i} + f'(x)|_{x=x_i}(x - x_i) + f''(x)|_{x=x_i}\frac{(x - x_i)^2}{2!} + \dots \tag{2.2}$$

olarak ifade edilir. Bu ifadede $x - x_i = \Delta x$ seçilerek $x = x_i + \Delta x$ yerine yazılırsa

$$f(x_i + \Delta x) = f(x_i) + f'(x_i)\Delta x + f''(x_i)\frac{\Delta x^2}{2} + f'''(x_i)\frac{\Delta x^3}{6} + f''''(x_i)\frac{\Delta x^4}{12} + \cdots \tag{2.3}$$

olmak üzere

$$\frac{f(x+\Delta x)-f(x)}{\Delta x}=f'(x)+\frac{\Delta x}{2}f''(x)+\cdots \tag{2.4}$$

ifadesi bulunur. Burada hatanın azaltılması için Δx'in sıfıra yaklaşması durumunda denklemin sağ tarafındaki ikinci terim $\frac{\Delta x}{2}f''(x)$ Δx'in birinci kuvveti ile sıfıra yaklaşmaktadır. Bu tür yaklaşım İleri Farklar (Forward Differences) olarak adlandırılır. Benzer şekilde Geri Farklar (Backward Differences) yaklaşımı

$$\begin{aligned} f(x_i-\Delta x)=f(x_i)-f'(x_i)\Delta x+f''(x_i)\frac{\Delta x^2}{2}-f'''(x_i)\frac{\Delta x^3}{6} \\ +f''''(x_i)\frac{\Delta x^4}{12}+\ldots \end{aligned} \tag{2.5}$$

ve

$$-\frac{f(x-\Delta x)-f(x)}{\Delta x}=f'(x)-\frac{\Delta x}{2}f''(x)+\cdots \tag{2.6}$$

olarak bulunur. İleri Farklar ve Geri Farklar denklemi taraf tarafa toplanarak

$$\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}=f'(x)+\frac{\Delta x^2}{6}f'''(x)+\cdots \tag{2.7}$$

şeklinde ifade edilen Merkezi Farklar (Central Differences) yaklaşımı bulunur.

$f(x+\Delta x)$ ve $f(x-\Delta x)$ Taylor serisi açılımlarının doğrudan toplanması ile ikinci mertebeden türevler

$$\frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2}=f''(x)+\frac{\Delta x^2}{12}f''''(x)+O(\Delta x^4) \tag{2.8}$$

şeklinde bulunur.

Birinci ve ikinci mertebeden türev ifadeleri

$$\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}=f'(x)+O(\Delta x^2)$$
$$\Rightarrow f'(x)\cong\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x} \tag{2.9}$$

ve

$$\frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2}=f''(x)+O(\Delta x^2)$$
$$\Rightarrow f''(x)\cong\frac{f(x+\Delta x)-2f(x)+f(x-\Delta x)}{\Delta x^2} \tag{2.10}$$

olarak gösterilirler.

2.2. Güncelleme Denkleminin Çıkarılması

Pratikte çoğu zaman ses hızının derinlikle değişimi durumu ile karşılaşıldığından zaman uzayı akustik dalga denklemi lineer, hiperbolik ve ikinci dereceden zamanda bağlı kısmi türevli bir diferansiyel denklem olarak

$$\Delta u(\vec{r},t)-\frac{1}{c^2(\vec{r})}\frac{\partial^2}{\partial t^2}u(\vec{r},t)=0 \tag{2.11}$$

biçimine dönüşür. Burada $u(\vec{r},t)$ akustik dalganın konum ve zamana göre dağılımını (akustik alan basıncını), $c(\vec{r})$ konuma göre ses hızını ifade etmektedir.

İki boyutlu durumda ZUSF yöntemini dalga denklemi Kartezyen koordinat üzerinden inceleyelim. Buna göre

$$\frac{\partial^2}{\partial x^2}u(x,z,t)+\frac{\partial^2}{\partial z^2}u(x,z,t)-\frac{1}{c^2}\frac{\partial^2}{\partial t^2}u(x,z,t)=0 \tag{2.12}$$

denklemi konumda $(i\Delta x,k\Delta z)$ ve zamanda $(n\Delta t)$ olmak üzere ayrıklaştırmak için sayısal türevler

$$\frac{\partial^2}{\partial x^2}u(x,z,t)\Big|_{i\Delta x,k\Delta z}^{n\Delta t}$$
$$= \frac{u^n(x+i\Delta x,k\Delta z)-2u^n(x,k\Delta z)+u^n(x-i\Delta x,k\Delta z)}{\Delta x^2} \tag{2.13}$$
$$= \frac{u_{i+1,k}^n-2u_{i,k}^n+u_{i-1,k}^n}{\Delta x^2}$$

ve

$$\frac{\partial^2}{\partial z^2}u(x,z,t)\Big|_{i\Delta x,k\Delta z}^{n\Delta t}$$
$$= \frac{u^n(z+i\Delta x,k\Delta z)-2u^n(z,k\Delta z)+u^n(z-i\Delta x,k\Delta z)}{\Delta z^2} \tag{2.14}$$
$$= \frac{u_{i+1,k}^n-2u_{i,k}^n+u_{i-1,k}^n}{\Delta z^2}$$

ve

$$\frac{\partial^2}{\partial t^2}u(x,z,t)\Big|_{i\Delta x,k\Delta z}^{n\Delta t} = \frac{u_{i,k}(t+n\Delta t)-2u_{i,k}(t)+u_{i,k}(t-n\Delta t)}{\Delta t^{\,2}} \tag{2.14}$$
$$= \frac{u_{i,k}^{n+1}-2u_{i,k}^n+u_{i,k}^{n-1}}{\Delta t^{\,2}}$$

olmak üzere dalga denkleminde yerine konulursa

$$\frac{u_{i+1,k}^n-2u_{i,k}^n+u_{i-1,k}^n}{\Delta x^2}+\frac{u_{i,k+1}^n-2u_{i,k}^n+u_{i,k-1}^n}{\Delta z^2} - \frac{1}{c^2}\frac{u_{i,k}^{n+1}-2u_{i,k}^n+u_{i,k}^{n-1}}{\Delta t^{\,2}}=0 \tag{2.15}$$

bulunur. Burada zamanda en ileride olan terim çekilerek ZUSF güncelleme denklemi

$$u_{i,k}^{n+1} = 2u_{i,k}^{n} - u_{i,k}^{n-1} + (c\Delta t)^2 \left[\frac{u_{i+1,k}^{n} - 2u_{i,k}^{n} + u_{i-1,k}^{n}}{\Delta x^2} + \frac{u_{i,k+1}^{n} - 2u_{i,k}^{n} + u_{i,k-1}^{n}}{\Delta z^2} \right] \quad (2.16)$$

olarak bulunur.

2.3. Sayısal Dispersiyon

ZUSF yönteminde problem uzayının ızgaralanması nedeniyle ortaya çıkan sayısal hataların kontrol altına alınması gerekmektedir. Sayısal dispersiyon dalganın faz hızının frekansa olan bağımlılığı olarak ifade edilir. ZUSF yönteminin ayrıklaştırdığı her bir ızgara hücresinde dalganın faz hızı değiştiğinden hatalara yol açılmaktadır. Bu hatalar

- Gerçek dışı sonuçlar,
- Çoklu saçılma problemlerinde yetersiz faz sıfırlamaları,
- Sayısal anizotropi,
- Sahte kırılmalar

olarak ortaya çıkmaktadır. Sayısal dispersiyonun etkisi

- Dalga boyu,
- Izgaradaki dalga yayılım yönü,
- Izgara ve ayrıklaştırma türü

faktörlerine göre değişmektedir. İki boyutlu Kartezyen koordinatlarda ZUSF için sayısal dispersiyon bağıntısı

$$\left[\frac{1}{c\Delta t} sin\left(\frac{w\Delta t}{2}\right)\right]^2 = \left[\frac{1}{\Delta x} sin\left(\frac{k_x^N \Delta x}{2}\right)\right]^2 + \left[\frac{1}{\Delta y} sin\left(\frac{k_y^N \Delta y}{2}\right)\right]^2 \quad (2.17)$$

olarak verilir. Burada k_x^N ve k_y^N sırasıyla *x* ve *y* yönlerindeki sayısal dalga sayısını ifade etmektedir. $k^2 = (k_x^N)^2 + \left(k_y^N\right)^2$ eşitliği ile dispersiyonsuzluk sağlanabilir. Burada $k = \omega/c$ analitik dalga sayısını gösterir. Sayısal dispersiyon hatasını azaltmak için Δx ve Δy mümkün olduğu kadar küçük seçilmelidir. Genellikle ızgaradaki hücre boyutu en az $\lambda/10$ seçmek yeterlidir [Aksoy, 2019].

2.4. Sayısal Kararlılık

Sayısal yöntemlerin kararlı olması gerekmektedir. Yani sınırlı giriş büyüklüğüne sınırlı çıkış olmalıdır. Buna göre ZUSF yönteminin Courant-Friedrichs-Lewy (CFL) kararlılık koşulunu sağlamalıdır. Bu kapsamda iki boyutlu Kartezyen koordinatlarda ZUSF çözümünün kararlı olması için Δt birim zaman adımının

$$\Delta t \leq \frac{1}{c\sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}}} \tag{2.18}$$

şartını sağlamalıdır. Burada c dalga hızını, Δx ve Δy sırasıyla *x* ve *y* yönlerindeki birim hücre boyutlarını gösterir [Aksoy, 2019]. Buna göre f frekanslı kaynak için, dalga boyu ($\lambda = c/f$) olmak üzere, birim hücre uzunluğu

$$\Delta x = \frac{\lambda}{10} \tag{2.19}$$

seçilerek, L_x bir boyutta problem boyunu göstermek üzere, N_x hesap noktası sayısı

$$N_x = \frac{L}{\lambda} \tag{2.20}$$

olarak bulunur.

3. GPU PROGRAMLAMA

Grafik İşlemci Ünitesi (Graphical Processing Unit, GPU) ekran kartı olarak da bilinen bu birim bilgisayarlarda ekrana yazılacak bilgilerin işlendiği donanım birimidir. Barındırdığı çok çekirdekli mimari sayesinde grafik işlemlerini hızlı bir şekilde yaparken aynı zamanda bilimsel hesaplama alanında da hızlı sonuçlar vermektedir.

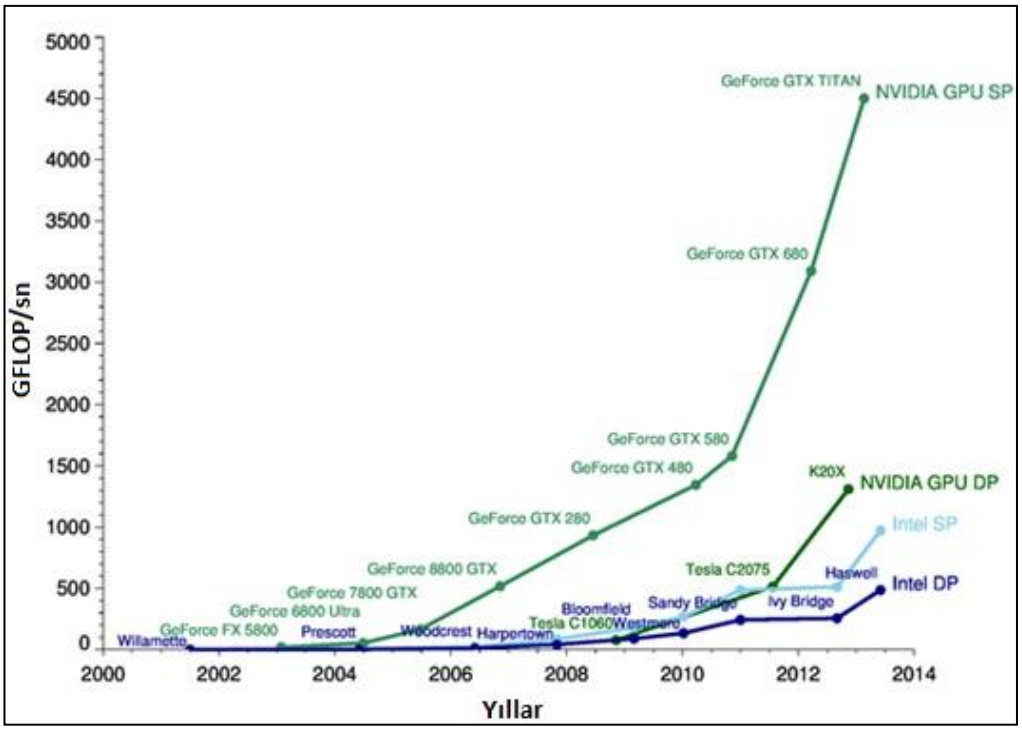
Ekran kartının yapısına bakıldığında temelde anakartta olduğu gibi bir işlemci, hafıza birimi ve giriş çıkış birimi bulunmaktadır. Bununla birlikte GPU yüksek matematik ve geometrik hesaplamalar gerektiren grafik işleme işi için özel olarak tasarlanmıştır. Günümüzdeki hızlı GPU'ların CPU'lardan daha fazla transistör içerdiği modelleri bulunmaktadır. Örneğin Intel Core i7 Broadwell-E işlemcisi 3.2 milyar transistörden oluşurken NVIDIA GV100 Volta ekran kartı 21.1 milyar transistörden oluşmaktadır [Web, 1], [Web, 2].

Ekran kartları için grafik bilgisinin hafızada tutulması amacı ile kullanılan Video RAM (VRAM) ve çerçeve tamponu (frame buffer) bulunmaktadır. Ekran kartı performansını etkileyen faktörlerden biri olan VRAM yüksek hızlarda çalışır ve yazma - okuma yapabilmek için iki kanallı (dual ported) yapıya sahiptir [Web, 3].

Yeni nesil ekran kartları bilgisayar kasasına PCIe hattı üzerinden bağlanmaktadır. Bu tezde de kullanılan NVDIA 1050 Ti ekran kartının kullandığı veri yolu PCIe v3 olup tek bir hattın hızı 985 MB/sn, 16 hatta sahip kartın anakart ile iletişim hızı ise 15.75 GB/sn'dir [Web, 4]. Bunun dışında ekran kartının performansı aşağıdaki parametrelerle bağlıdır:

- GPU saat hızı (MHz),
- Artırılmış GPU saat hızı (MHz),
- Çekirdek sayısı,
- Hafıza veri yolu boyutu (bit),
- Hafıza türü,
- Hafıza miktarı (MB),
- Hafıza saat hızı (MHz),
- Hafıza bantgenişliği (GB/sn),
- RAM analog-dijital dönüştürücü hızı (MHz).

Pazarda son yıllarda artan yüksek çözünürlüklü, 3 boyutlu grafik uygulamalarıyla beraber programlanabilir GPU gelişimi de yüksek paralel işlem kabiliyeti, çok iş parçacıklı (multi-thread), çok çekirdekli ekran kartları geliştirilmiş olup, CPU ile arasındaki saniyede yapılan gezer nokta işlem sayısı (floating point operations per second, FLOPS) birimi bakımından karşılaştırılması Şekil 3.1'de ifade gösterilmiştir [Web 6, 2019]. Buna göre son yıllardaki ekran kartının işlem yapabilme gücünün artışının, CPU'ya göre çok daha fazla olduğu gözükmektedir.



Şekil 3.1: CPU ile ekran kartının hesaplama gücü karşılaştırması.

CPU ile GPU arasındaki hesaplama gücü farkı yapılarında bulunan artimetik mantık birimi (Arithmetic Logic Unit, ALU) sayısından kaynaklanmaktadır. GPU'da resmetme (rendering) işlemi için gereken yüksek paralel hesaplamayı sağlayan daha çok sayıda transistör ile verinin daha hızlı işlenmesi, veri yakalama ve akış kontrolünün elde edilmesi gerekmektedir. Bunu için GPU yapısında çok daha fazla sayıda hesaplama ünitesi bulunmaktadır. Bu hesaplama üniteleri hem tek duyarlı kayan nokta formatı (single point floating point) hem çift duyarlı kayan nokta formatı (double point floating point) işlemlerini yapabilmektedir. Şekil 3.2'de işlemci ile ekran kartında bulunan işlem bloklarının karşılaştırması gözükmektedir [NVIDIA, 2017].

Şekil 3.2: CPU-GPU temel mimari yaklaşımı.

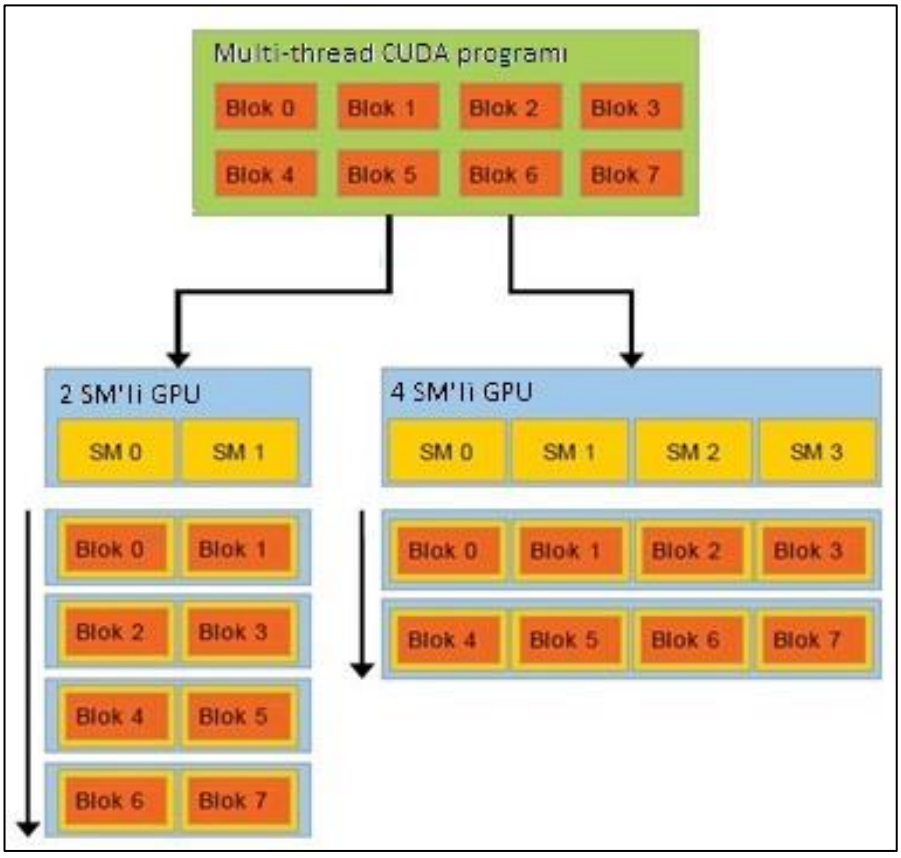
GPU'lar özel olarak veri yoğun paralel hesaplama problemleri için çok uygun yapıdadır. Aynı programın yüksek aritmetik hassasiyetle farklı veriler üzerinde çalışmasına olanak sağlar. Özellikle hesaplama yapılacak veri üzerinde yapılacak işlem sayısı fazla olup hafızaya erişim ihtiyacı az olduğu sistemlerde daha yüksek performanslı çözümler gerçekleştirilebilmektedir. Ekran kartlarının en yoğun kullanıldığı 3 boyutlu resmetme işlemi için büyük piksel setleri ve köşe nokta bilgileri paralel iş parçacıklarında (thread) hesaplanarak hızlı işlemler gerçekleştirilir.

3.1. CUDA ile GPU Programlama

2007 yılında NVIDIA firması tarafından kendi ekran kartlarında kompleks hesaplamalı problemleri işlemciye göre yüksek verimlilikte çözebilen CUDA (Compute Unified Device Architecture) genel amaçlı paralel hesaplama platformu ve programlama modelinin tanıtımını yapılmıştır. CUDA platformu C, C++, Fortran, Pyton, Java programlama dilleri için kullanılabilir olarak uygulama geliştiricilerinin kullanıma sunulmuştur [Web, 5].

CUDA geliştirme ortamında paralel programlamanın da temelini oluşturan iş parçacıklarının hiyerarşisi, paylaşımlı bellek ve bariyer senkronizasyonu konularını C diline birkaç uzantı yapılarak soyutlaştıran ve programcının kullanımına basitleştirerek sunan uygulama program arayüzü (Application Programming Interface, API) bulunmaktadır. Bu soyutlamalarla birlikte tam problemin parçalara ayrılarak alt problemlerinin çıkarılması ve tanımlanan ortak iş paketinin farklı veriler için tüm çekirdekte bir blok içerisinde çözülmesiyle tek bir saat çevriminde hesaplanan veri miktarı artışı sağlanmaktadır.

Hiyerarşik olarak geliştirilen iş parçacıklarının organizasyon yapısı sayesinde bloklar halinde çalışan iş parçacıkları mevcut donanım durumuna göre paralel veya seri çalışması ile geliştirilen program ölçeklenebilir yapıda olup, aynı programın daha yüksek hızlarda çalışan NVDIA ekran kartının takılı olduğu sistemde doğrudan çalışabilmesini sağlanmaktadır. GPU'da her bir akış çoklu işlemcileri (streaming multiprocessor, SM) içerisinde 128 tane CUDA çekirdeği bulunmaktadır. Bu tezde kullanılan ekran kartında 6 tane SM olup toplam 768 CUDA çekirdeği bulunmaktadır. Multi-thread hazırlanan program ölçeklenebilir yapısı sayesinde farklı sayıdaki SM içeren sistemlerde de çalışabilmektedir. Şekil 3,3'de 4 SM içeren sistem 2 SM içeren GPU'ya göre elindeki iş parçacığı bloklarını iki katı oranda hızlı çalıştırmaktadır. Şekil 3.3'de çok iş parçacıklı CUDA programının farklı donanım özelliklerine sahip ekran kartlarında nasıl işlendiği gözükmektedir. Buna göre işlem yapacak akış çoklu işlemci sayısı arttıkça paralel iş yapabilme kabiliyeti artmaktadır [NVIDIA, 2017].



Şekil 3.3: GPU'da ölçeklenebilirlik.

3.1.1. CUDA Programlama Temelleri

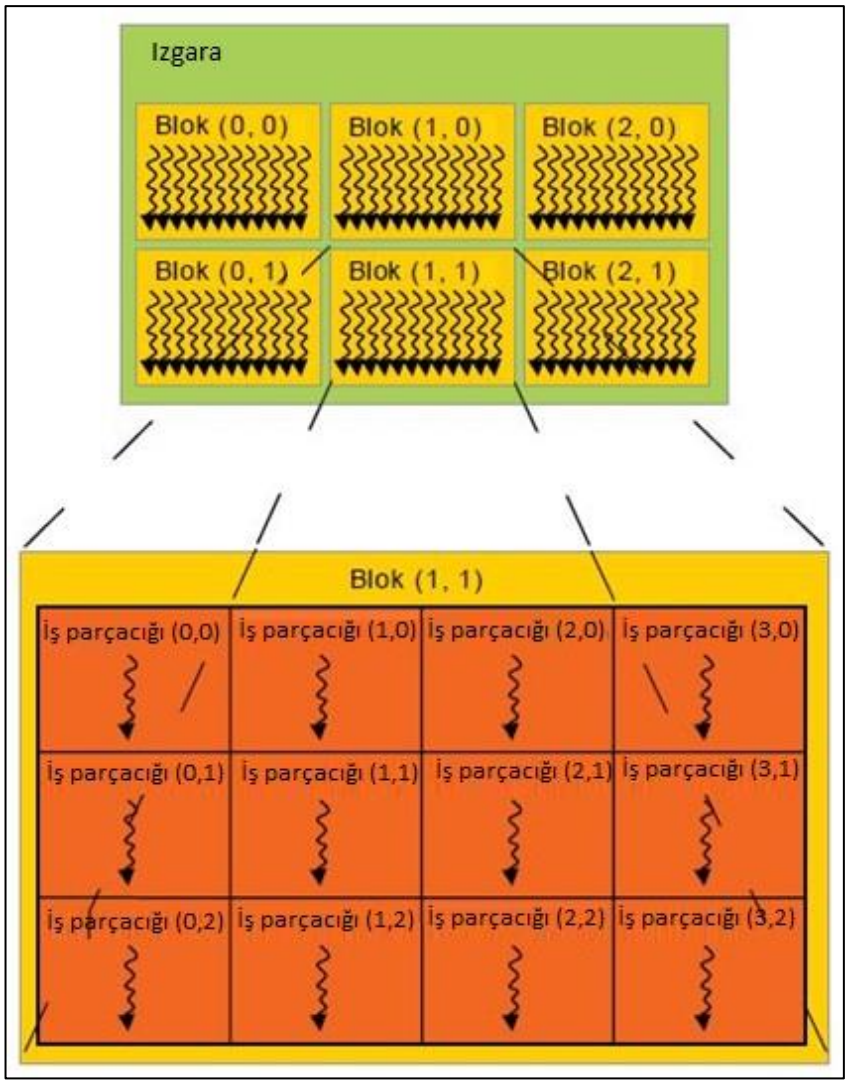
3.1.1.1 Kernel

CUDA, C dili sözdizimini genişleterek programcıya C dilinde fonksiyonlar hazırlamasına olanak verir. ***Kernel*** adı verilen C fonksiyonu cihazda *N* kez *N* farklı CUDA iş parçacıkları tarafından paralel olarak çalışmasını sağlar.

Kernel fonksiyonu oluşturmak için fonksiyon deklarasyonunda en başa *__global__* anahtar kelimesi yazılır. İlgili kernel fonksiyonu kullanılacağı zaman CUDA'da çalıştırma ayarı sözdizimi olan <<<...>>> ifadesi kullanılır. Kernel fonksiyonu içerisinde o anda hangi bloktaki hangi iş parçacığının bulunduğunun bilgisi yer almaktadır. Bu sayede çözülen problem alan dekompozisyonu yapılarak ayrılan bloklar ve thread indeks değerlerine göre hesaplama işlemi yapılır.

3.1.1.2 İş Parçacığı Hiyerarşisi

Çalıştırma ayarı sözdizimi *(<<<N,M>>>)* ile GPU'da çalıştırılacak kernelin kaç blokta ve her blokta kaç tane iş parçacığından oluşacağı belirlenir. Burada *N* değeri kaç tane blok olduğu, *M* değeri her blokta kaç iş parçacığı olduğunu ayarlamak için kullanılır. Blok ve iş parçacıklarından oluşa bu yapıya ızgara (grid) adı verilir. Şekil 3.4'te ızgarada yer alan bloklar ve blok içerisinde yer alan iş parçacıklarının düzeni verilmiştir.



Şekil 3.4: Izgara üzerindeki blok ve iş parçacığı hiyerarşisi.

Bir çekirdekteki bellek ve işlem gücü kaynağının ortak olarak kullanımından dolayı bir blokta yer alan iş parçacığı sayısı 1024 olarak sınırlandırılmıştır. Bununla birlikte ızgarada yer alacak toplam blok sayısının sınırı yoktur. Paralel işlenemeyecek kadar çok blok olursa seri olarak hesaplama yapılır.

Bloklar bir boyutlu, iki boyutlu ve üç boyutlu olarak tanımlanabilmektedir. Bir kernel o anda hangi blokta çalıştığını *blockIdx.x, blockIdx.y, blockIdx.z* yerleşik değişkenleri ile öğrenilir. Bir bloğun kaç iş parçacığından oluştuğunu ise *blockDim* yerleşik değişkeni ile öğrenilir. Şekil 3.5'de örnek CUDA kod parçasında A ve B matrislerini toplayıp sonucunu C matrisine yazılmasını sağlayan kod parçası gözükmektedir [NVIDIA, 2017].

```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

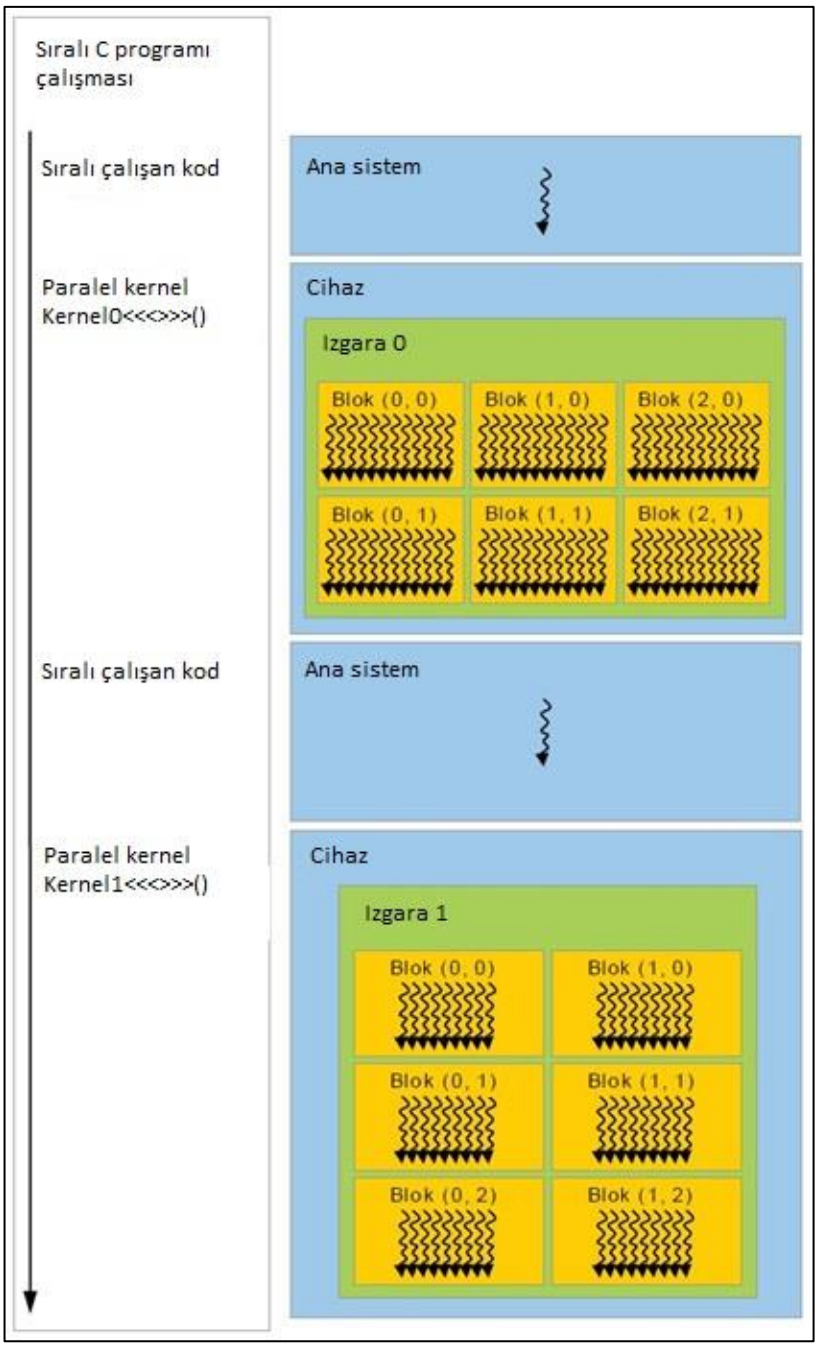
Şekil 3.5: Örnek CUDA kodu.

Burada *__global__* ön adıyla *MatAdd* fonksiyonu kernel fonksiyonu olarak tanımlanmıştır. Fonksiyona $N \times N$ boyutlarında A ve B matrisleri parametre olarak verilmiştir. Burada işlemi paralel hale getiren en önemli unsur i ve j indeks değerleridir. A ve B matrislerinin Şekil 3.4'teki gibi iki boyutlu düşünülüp her bir noktayı i ve j değerleriyle ifade edilmiştir. Matrisin i'inci elemanında çalışacak kerneli bulmak için x yönündeki $blok\ sayısı * blok\ indeksi + iş\ parçacığı\ indeksi$ toplamından bulunur. Benzer formül y yönündeki j'inci eleman için uygulandığında bir iş parçacığı C matrisi üzerindeki bir noktanın hesabını yapmaktadır. Bu iş parçacığı sayısı her bir blok için x ve y yönlerinde 16 seçilirse tüm iş parçacıklarının paralel olarak çalışabilmesi için bir yöndeki hesaplanacak $matris\ boyu\ /\ 16$ yapılarak toplam blok sayısı elde edilmiş olur. Böylece paralel olarak matris toplama işlemi gerçekteştirilir.

Bloktaki tüm iş parçacıkları paralel olarak işlenir. Eğer paylaşımlı bellek kullanılıyorsa bloktaki iş parçacıkları bir sonraki hesaplamaya geçmeden önce *__syncthreads()* fonksiyonu kullanılıp veri değiştirme işlemi yapılarak bir sonraki çevrimde diğer iş parçacıklarının da aynı veri üzerinde çalışması sağlanır.

3.1.2. CUDA Program Akışı

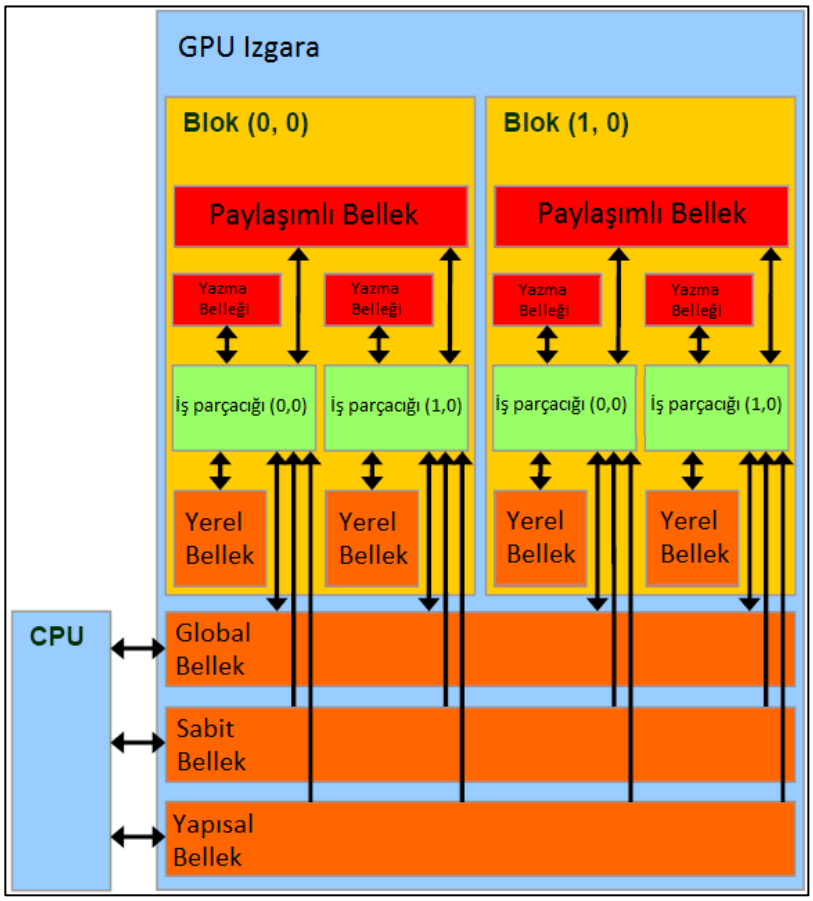
CUDA programı işlemci tarafında çalışan ana sistem (host) kodları ve ekran kartında çalışan cihaz (device) kodlarından oluşmaktadır. Program işlemci tarafından başlatılır. Hesaplanması istenen veriler cihaza gönderilip hesaplama sonucu alınır. Şekil 3.6'da örnek bir iş akışı gösterilmiştir.



Şekil 3.6: CUDA'da örnek iş akışı.

3.1.3. Hafıza Türleri

CUDA'da bulunan bellek türleri hızlarına ve kapasitelerine göre çeşitlilik göstermektedir. Şekil 3.7'de cihazdaki bellek erişim yapısı gösterilmiştir.



Şekil 3.7: CUDA bellek erişim yapısı.

Cihazda bulunan global bellek (global memory) ekran kartının tanımında yazan VRAM (Video RAM) değerinde olan bellektir. Bu tezde 4 GB VRAM'i olan ekran kartı kullanılmıştır. Bu bellek ana sistem ile cihazın ortak kullanabildiği bir alandır. Erişim hızı yavaş olup yüksek hafıza kapasitesine sahiptir.

Sabit bellek (constant memory) cihazda sadece okuma yapılabilen bellektir. Programın çalışma süresince ihtiyaç duyacağı sabitlerin bu alana yazılabilir. Erişim hızı global bellekten hızlı olup, tüm iş parçacıkları tarafından kullanılabilir. 64 KB'lık bir bellek alanına sahiptir.

Yapısal belleği (texture memory), sabit bellek gibi sadece okuma yapılabilen bellektir. Özellikle grafik uygulamaları için tasarlanmıştır. Çip üzerinde önbellekte (cache) bulunmaktadır.

Yerel bellek (local memory) program çalışırken her bir iş parçacığının sadece kendisinin erişebildiği bellek türüdür. Burada tutulan veriler iş parçacığı çalışması bitene kadar saklanır.

Yazma belleği (register) her bir iş parçacığının program esnasında kullandığı en küçük veri saklama birimidir. Diğer hafıza türlerinden hızlı çalışır.

Paylaşımlı bellek (shared memory) aynı blok içerisinde yer alan iş parçacıklarının ortak olarak erişebildiği bellek türüdür. İş parçacıklarının organizasyonu doğru yapılıp, senkronizasyon sağlanırsa çok hızlı çalışmaktadır.

3.2. GPU ile ZUSF Uygulamaları

3.2.1. İki Boyutlu Uygulamalar

Deneysel alan ayrıştırma tekniğinin birden çok GPU'da CUDA ve MPI ile programlanması yöntemiyle yarı sonsuz iki boyutlu homojen olmayan ortamda yayılan akustik dalgaların ZUSF çözümü yapılmıştır. Çalışma sonucunda GPU blok düzeni için blok başına düşen iş parçası (thread) sayısının 32×16 olarak düzenlenmesinin, 16 x 16 ve 16 x 32'lik blok yapılarına göre işlem süresi ve işlenen veri büyüklüğü bakımından deneysel olarak en iyi sonucu verdiği görülerek 32×16 blok yapısı için sistemin ölçeklenebilirlik analizi yapılmıştır. CUDA yerine OpenCL ile programlama yapılırsa, diğer marka GPU'ların da programlanabileceği belirtilmiştir. Sistemde birden çok bilgisayarın (küme tabanlı) kullanımı tek bir GPU için hafıza problemini çözülmüştür [Zamith et al., 2010].

Bölümlenmiş (tiled) 16×16 boyutlarında iş parçacığı blokları ile alan ayrıklaştırma yöntemi kullanılarak iki boyutlu oda akustiği problemi çözülmüştür. Düğüm sayısına göre hesaplama süresi analiz edilmiştir. CPU ve farklı modelde GPU tabanlı çözümler ile sonuçlar karşılaştırılmıştır. Buna göre düğüm sayısının az olduğu problemlerde CPU, düşük güçteki GPU'lardan daha iyi çalışmakta, problem boyutu büyüğünde ise kullanılan bütün GPU'lar CPU'ya göre daha hızlı çalışmaktadır. İncelenen örnekler farklı GPU'larda çalışabilecek şekilde ölçeklenebilir olarak geliştirilmiştir [Southern et al., 2010].

Yapısal bellek ve paylaşımlı bellek kullanan kernel yöntemiyle iki boyutlu homojen olmayan ortamda akustik dalga yayılımı problemi çözülmüştür. Problem büyüklüğüne göre hafıza kullanımı, 100.000 benzetim adımından sonra geçen süre ile hesaplama süresi ve hafıza kopyalama işlemleri için geçen toplam süre bakımından sonuçlar elde edilmiştir. Sonuçlar yapısal bellek ile paylaşımlı bellek için de karşılaştırılmıştır. Buna göre paylaşımlı belleğin yapısal belleğe göre daha verimli çalıştığı görülmüştür. Ayrıca çözülen problem büyüklüğü arttıkça kullanılan hafıza miktarının da doğrusal olarak arttığı tespit edilmiştir [Brandao et al., 2010].

16×16 boyutlarındaki iş parçacığı blokları kullanılarak iki boyutlu elektromanyetik dalga yayılımı problemi çözümü için çalışma yapılmıştır. CPU ve GPU hesaplama süresi türünden elde edilen sonuçlara göre karşılaştırmalar yapılmıştır. Buna göre GPU yaklaşık olarak 10 ile 20 kat arasında daha hızlı çalışmaktadır [Balevic et al., 2008].

3.2.2. Üç Boyutlu Uygulamalar

Veri döşeme (data tiling) ile tek geçiş (single pass) yaklaşımını kullanarak üç boyutlu sismik dalgaların hesaplamasını yapılmıştır. Hesaplanan nokta sayısı Mpoints/s (millions of output points per second) türünden elde edilen sonuçlarda değişen boyutlara göre; uzaydaki mertebe (order in space), döşeme boyutu (tile dimension) ve GPU sayılarının etkisi karşılaştırılmıştır. 16×16 iş parcacıklarının kullanıldığı tek şema hesaplamasında, yüksek mertebelerdeki halelerin (halo, 4 tane 4×16'lık bölge) okuma bantgenişliğini daha yüksek oranda kullandığı tespit edilmiştir. Sabit hacim boyutları durumunda, uzaydaki mertebe arttıkça okuma fazlalığından dolayı toplam yapılan işin azaldığı gözlenmiştir. Tek GPU'nun kullanıldığı, uzayda sekizinci mertebe ve zamanda ikinci mertebedeki hesaplamada GPU'nun CPU'ya göre 10 kat daha performanslı olduğu, ayrıca; küçük ölçekli problem boyutlarında 16×16'lık döşeme boyutlarının 32x32'lik boyutlara göre daha iyi olduğu, büyük ölçekli problemde ise 32×32 döşeme boyutlarının daha iyi olduğu sonuçları çıkarılmıştır. Çoklu GPU hesaplamasında GPU başında düşen dilimler 200 veya daha fazla oldukça, haberleşme gereksinimleri gizlenebilir hale gelmektedir. Büyük boyutlarda daha çok GPU kullanımının performans ve ölçeklemeyi arttırdığı sonucu ortaya çıkmıştır [Micikevicius, 2009].

Doğrusal ayrıştırma tekniğinin bölümlenmiş (tiled) temelli 16x16'lık iş parçacığı blokları kullanılmasıyla yapılan çalışmada 44.1 kHz'de 40 m^3 hacmindeki üç boyutlu odada sınır kayıplı akustik dalga denkleminin ZUSF ile çözümü yapılmıştır. Matlab, C ve CUDA ile aynı hesaplamalar yapılmış, sonlu hassasiyette farklılıklar oluştuğu görülmüştür. Paralel GPU iş parçacıkları ile yapılan hesaplamanın seri hesaplamaya göre 80 kat daha hızlı olduğu tespit edilmiştir. Birden çok GPU'nun MPI ile programlanmasıyla hızlanma sağlanarak fiziksel olarak daha büyük problemlerin çözülebileceği belirtilmiştir [Webb and Bilbao, 2011].

İç Çarpım (Dot Product) yaklaşımı ile global bellek dilimleme (global memory slicing) yöntemi kullanılarak üç boyutlu oda akustiği problemini sınır olmayan, frekans bağımsız sınır ve frekans bağımlı sınır modelleri için çözmüştür. İç Çarpım ve Sade (Naive) yaklaşımları için, saniyede işlem yapılan mega voksel sayısının (MV/s); sınır düğümlerinin (node) tüm düğümlere oranı (nB/nT), hesaplanan eleman sayısı, filtre derecesi, iş parçacığı (thread) başına düşen blok/çekirdek sayısına göre değişim sonuçları bulunmuştur. İç Çarpım yaklaşımı ve sade yaklaşım için farklı sınır, farklı şablon (stencil) ve farklı oda boyutları için karşılaştırma yapılmıştır. Sade yaklaşımda, artan oda boyutu ile artan boş sınır düğümleri için performansın arttığı gözükmüştür. Oda boyutlarının; $64 \times 64 \times 128$, $128 \times 128 \times 128$, $256 \times 256 \times 128$, $512 \times 512 \times 128$, $1024 \times 1024 \times 128$ olması durumlarında sırasıyla blok başına düşen optimum iş parçacığı (x, y); 128×4, 128×4, 256×2, 512×2, 1024×1 olarak bulunmuştur. Sınır düğümleri için kullanılan sayısal empedans filtresi (DIF) yaklaşımında, filtre derecesinin artması ve nB/nT oranının düşmesiyle nispi performans artışı görülmüştür. Şablon (stencil) sayısının artması performansı düşürmüştür. nB/nT oranı arttıkça Sade yaklaşımda performans düşmüş, İç Çarpım yaklaşımında performans yaklaşık olarak aynı kalmıştır. Sınır düğümü olmadığı koşulda Sade yaklaşım, diğer durumlarda İç Çarpım yaklaşımı en iyi performansı göstermiştir. Çalışmada ayrıca, simulasyona rastgele yerleştirilen farklı geometri ve akustik özelliklerdeki engeller olabildiği durumda, İç Çarpım yaklaşımının iş parçacıklarına (thread) eşit oranda iş yükü getirdiği belirtilmiştir [Spa et al., 2015].

Izgara üzerindeki her bir düğüm için GPU'da bir iş parçacığı atayarak üç boyutlu oda ve salon geometrileri için 7 kHz'e kadar akustik benzetim ve ses oluşturma (auralization) üzerinde çalışmıştır. Hesaplanan düğüm sayısı ve dijital empedans filtresi derecesine göre hesaplama süresi artışı sonuçlarını elde etmiştir. Standart doğrusal (SRL) ve aradeğerli genişbant şemalar karşılaştırılmıştır.

Aradeğerli genişbant şemanın standart doğrusal şemaya göre yaklaşık 3.8 kat daha verimli olduğu tespit edilmiştir [Savioja, 2010].

İki boyutlu genişletilmiş bölümleme (tile) bloklarının paylaşımlı bellek kullanılarak yedi nokta ZUSF yöntemi ile 44.1 kHz'de 6.5 m, 3.5, x 12 m boyutlarındaki oda akustiği problemi çözülmüştür. Farklı blok düzenlemeleri ve sürümleri kullanılarak hesaplama süresi bakımından sonuçlar elde edilmiştir. Buna göre 16×16 iş parçacığı blok yapısında genişletilmiş bölümleme kullanılarak en hızlı sonuçlar elde edilmiş, paylaşımlı bellek kullanıldığında ek olarak %3 daha hızlı sonuç elde edilmiştir. İki boyutlu yinelemeli yaklaşımda en yavaş hesaplama süresi elde edilmiş, bir boyutlu iş parçacığı bloğu kullanıldığı yapının ise sadece satır boyutunda verimli çalıştığı sonucu elde edilmiştir [Webb and Bilbao, 2011].

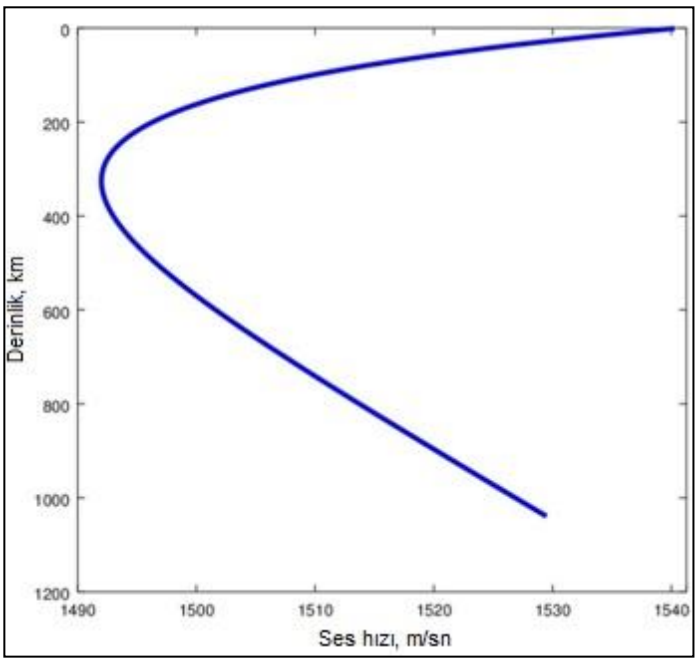
Tiled (bölümlenmiş) ve sliced (dilimlenmiş) temelli kernel modelleri ile yapılan GPU hesaplamasıyla üç boyutlu oda akustik dalga yayılımının benzetimi gerçekleştirilmiştir. Yapılan çalışma sonucunda kernelin uygulanma biçiminin performansa etki ettiği görülmüştür. Tek hassasiyetli sayılarda tiled (bölümlenmiş) kernel en iyi sonucu verirken, çift hassasiyetli sayılarda sliced (dilimlenmiş) kernel en iyi sonucu vermiştir. Merkezi fark sınır formülü kullanan kernel ile en kötü performansın elde edildiği görülmüştür. Kodlama sırasında const__restrict anahtar sözcüğü kullanılmasıyla daha verimli kod yazılabileceği belirtilmiştir [Saarelma and Savioja, 2014].

4. SAYISAL SONUÇLAR

Bu bölüm iki başlık halinde incelenmiştir. İlk başlıkta literatürdeki SOFAR kanalı ile ilgili çalışmalar temelinde MATLAB ve GPU üzerinde çözümler karşılaştırılarak doğru sonuç alınması amaçlanmıştır. İkinci başlıkta farklı kaynak frekansı değerleri için GPU'da problemin ihtiyaç duyduğu zaman ve RAM kullanımı değerleri gözlenerek performans ile ilgili analiz yapılabilmesi sağlanmıştır. ZUSF problem uzayında gerekli dış sınırlar birinci mertebeden Mur tipi soğurucu sınır koşulu uygulanarak sonlandırılmıştır.

4.1. MATLAB ve GPU Çözümünün Doğrulanması

Munk ses hızı profilinde $z_{minimum}$ ile ifade edilen derinlik ses hızının minimum olduğu nokta için kullanılmış ve okyanuslar için 1300 m olarak belirtilmiştir [Munk, 1974]. Ancak problem uzunluğu GPU'daki VRAM kısıtı nedeniyle x yönünde 10.000 m, z yönünde 160 m olarak seçilmiştir. Bu problem uzayında SOFAR kanalının gözükmesi için ses hızının minimum olduğu noktanın da değiştirilmesi gerekmektedir. Bu nedenle (1.1) eşitliğinde $z_{minimum}$ yerine 50 m seçilmiş ve Şekil 4.1'deki ses hızı profili elde edilmiştir.



Şekil 4.1: Tezde kullanılan ses hızı profili.

Kaynak işareti olarak

$$u(\vec{r},t) = u(\vec{r},t) + 2\cos(2\pi f t) \tag{4.1}$$

kullanılmıştır. Burada f kaynak işaretinin frekansını ifade etmektedir. Kaynak işareti frekansı öncelikle 1 kHz olarak belirlenmiştir. ZUSF benzetim süresi 8 sn olarak belirlenmiştir. Bu süre yaklaşık 1500 km/sn hızındaki işaret için 10 km menzilindeki problem uzayının sonuna ulaşabilmesi için yeterli bir süredir. Buna göre (2.18), (2.19), (2.20) ve (2.21) eşitlikleri kullanılarak $\Delta x = \Delta z = 0.154026225$ m, $\lambda = 1.54026222$ m, $N_x = 64925$, $N_z = 1039$ olarak hesaplanmıştır.

Problemin elektriksel uzunluğunu hesaplamak için ses hızının minimum ve maksimum olduğu noktalar hesaplanır. [Munk, 1974] makalesinde ses hızının minimum değerinin ifade eden $c_{minimum}$ hızı 1460 m/sn olarak verilmiştir. Buna göre x yönünde 10.000 m, z yönünde 160 m olan problem uzunlukları için (1.1) numaralı denklemde yerine konulduğunda 1540,3 m/sn maksimum ses hızı elde edilir. Buna göre (2.19), (2.20) ve (2.21) denklemleri kullanılarak problemin elektriksel uzunluğu hesaplanırsa $c_{minimum}$ - $c_{maksimum}$için

- x yönünde: 6492.2λ - 6849.3 λ,
- z yönünde: 103.8λ - 109.5λ

bulunur.

CUDA ZUSF kodu önce C++'da geliştirilmiş, sonra CUDA'ya aktarılmıştır. Kullanılan laboratuar bilgisayarının özellikleri aşağıda verilmiştir.

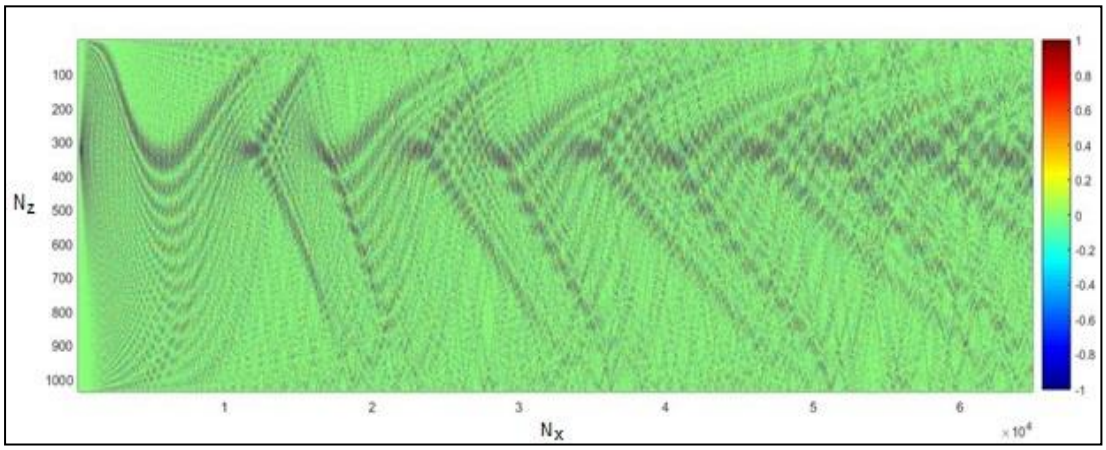
- İşlemci: Intel® Core™ i7-960 İşlemci 8M Önbellek, 3,20 GHz, 4.80 GT/s,
- RAM: 24 GB,
- Ekran Kartı: NVIDIA GeForce GTX 1050 TI 4GB VRAM, CUDA çekirdeği sayısı: 768, GPU maksimum saat frekansı: 1.49Ghz,
- İşletim Sistemi: Windows 7.

4.1.1. MATLAB Sonuçları

MATLAB kodunda döngü içerisinde yapılan hesaplamaların daha hızlı yapılabilmesini sağlayan vektörizasyon uygulanmıştır [Web, 7]. Bu durumda kodun çalışması yaklaşık 64 saat sürmüş, 1.7 GB RAM harcamıştır. MATLAB ile hesaplanan sonuçların doğruluğunun kontrol edilmesi için-alan dağılımı grafiği-çeşitli noktalardan alınan işaretlerin grafiği ve yayılım kaybı grafiği elde edilmiştir.

- Alan Dağılımı

Şekil 4.2'de literatürde Şekil 1.2'de görülen SOFAR kanalının MATLAB ile hesaplanan ZUSF sonucu bulunmaktadır. Beklendiği gibi ses hızının minimum olduğu derinlik kapsamında SOFAR kanalı açıkça gözlenmektedir.

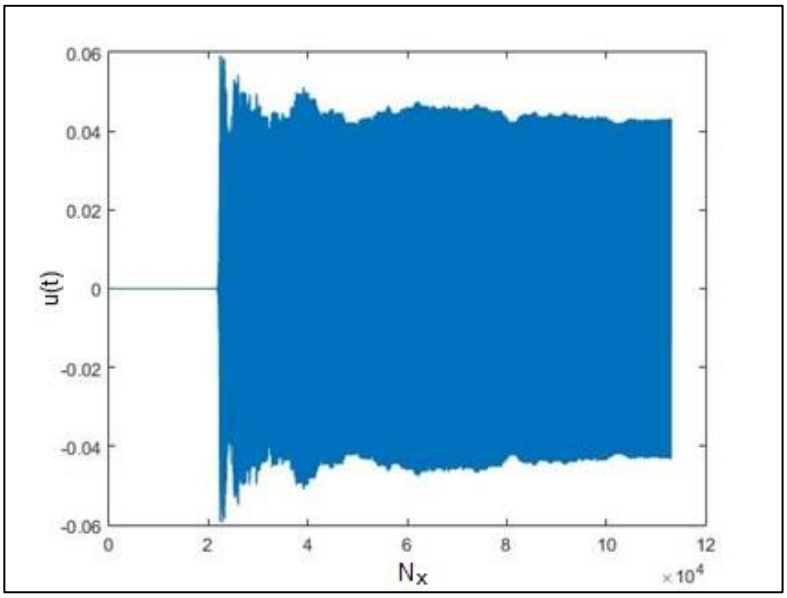


Şekil 4.2: MATLAB ile elde edilen SOFAR kanalı alan dağılımı grafiği.

- Çeşitli Noktalardan Alınan İşaret Örnekleri

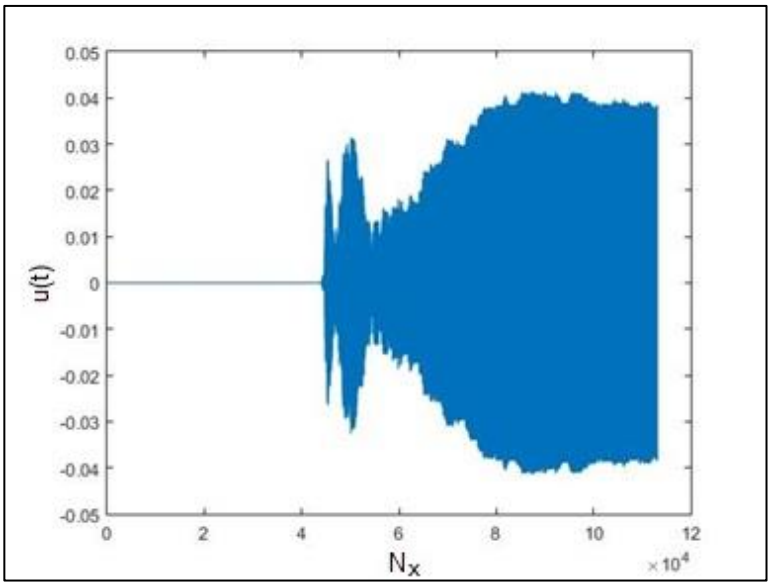
Kaynaktan iletilen işaret için problem uzayında belirli gözlem noktalarında işaretin zamana göre değişim grafiği hesaplanmıştır. Buna göre kaynağa yakın olan noktalarda işaret kısa süre sonra gözlenirken, kaynaktan uzak olan mesafelerde işaretin gözlenme süresi de artmaktadır. Bu durum nedensellik (causality) prensibi ile uyumludur. Ayrıca ses hızı profiline göre alınan işaretlerin şiddetleri de farklıdır. Bu durum yol kaybı kapsamında tutarlıdır. Ses hızının minimum olduğu derinlikte SOFAR kanalının merkez hizasından alınan örneklerde işaretin bütünlüğünü koruması, SOFAR kanalının başlangıç ve bitiş noktalarından alınan işaretlerde ise bir

miktar bozulma beklenmektedir. Buna göre 20 m derinlik ve 2311 m menzil için elde edilen zaman uzayı işareti Şekil 4.3'te verilmiştir.



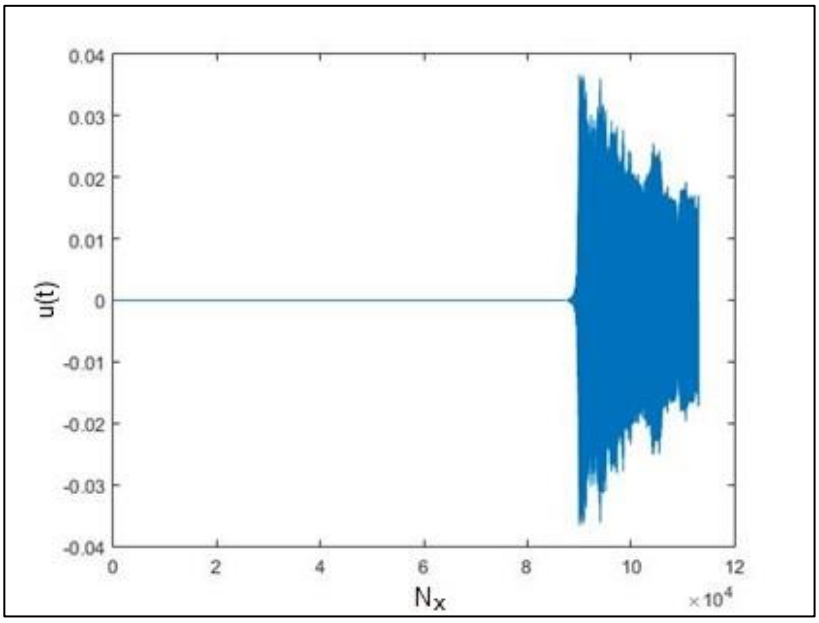
Şekil 4.3: MATLAB ile 20 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 20 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.4'te verilmiştir.



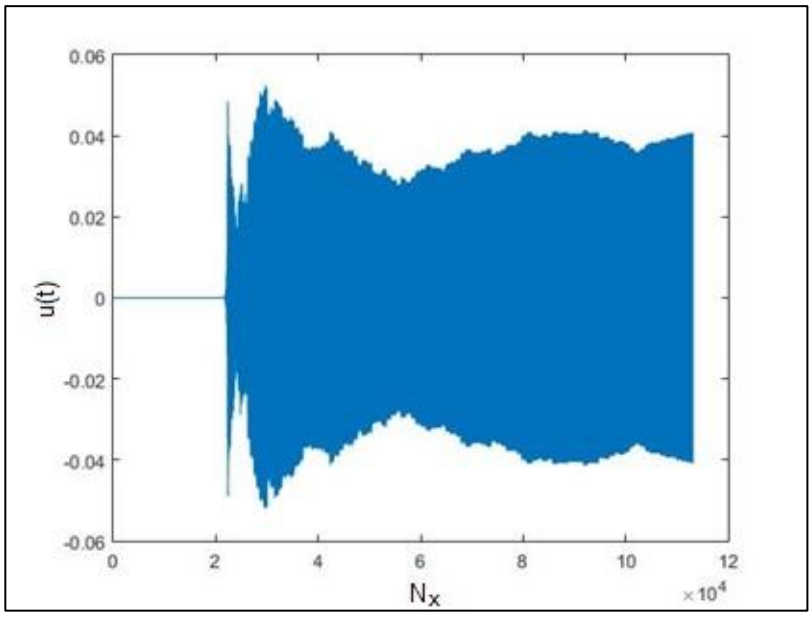
Şekil 4.4: MATLAB ile 20 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 20 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.5'te verilmiştir.



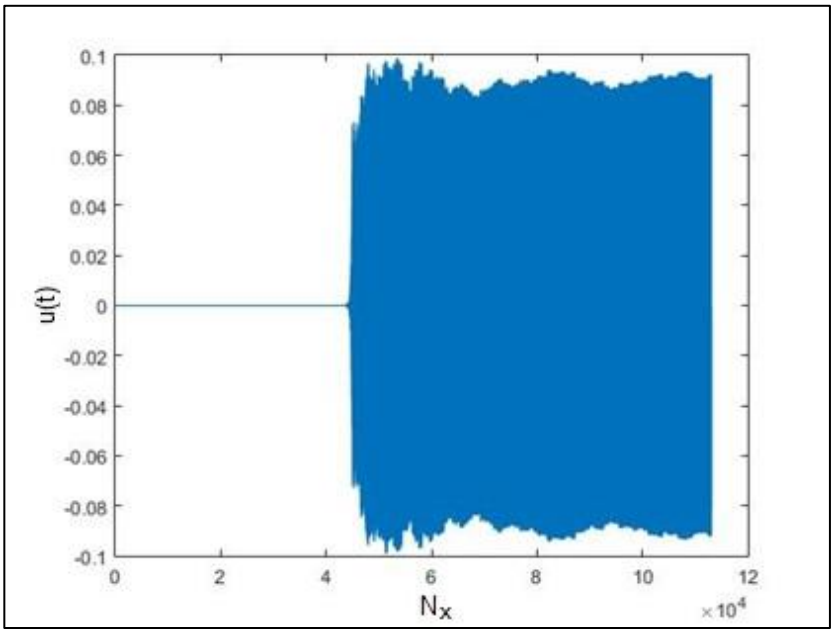
Şekil 4.5: MATLAB ile 20 m derinlik ve 9244 m menzil için hesaplanan işaret.

SOFAR kanalında ses hızının minimum olduğu derinlik olan 50 m alınan işaretler için sonuçları aşağıda incelenmiştir. Buna göre 50 m derinlik ve 2311 m menzil için hesaplanan işaret Şekil 4.6'da verilmiştir.



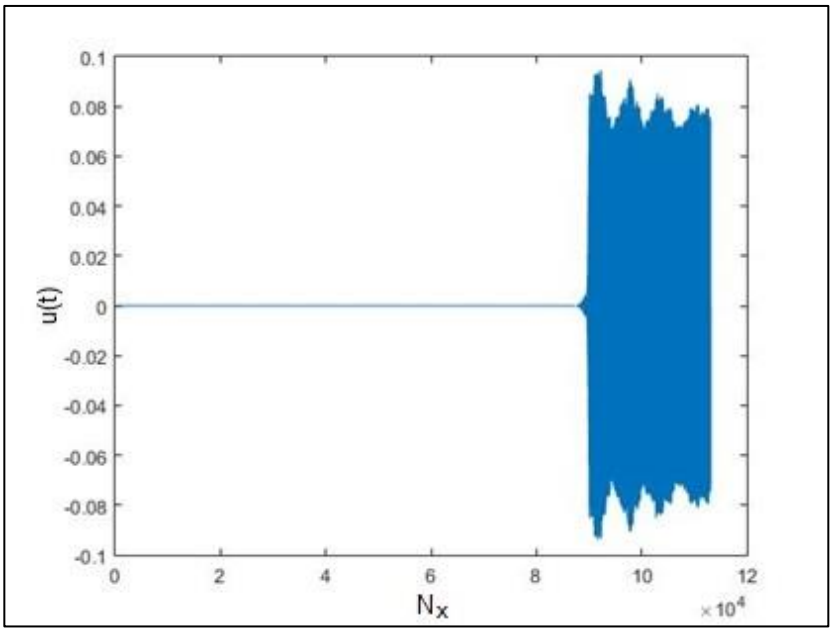
Şekil 4.6: MATLAB ile 50 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 50 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.7'de verilmiştir.



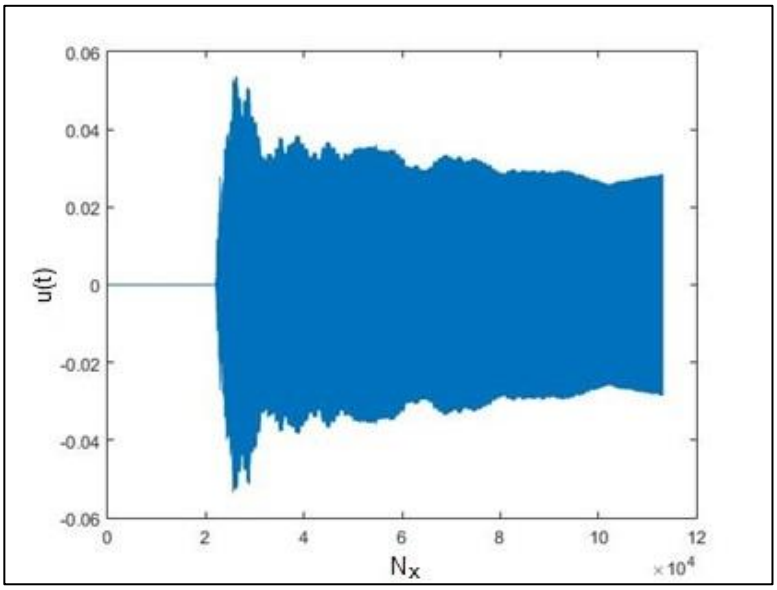
Şekil 4.7: MATLAB ile 50 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 50 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.8'de verilmiştir.



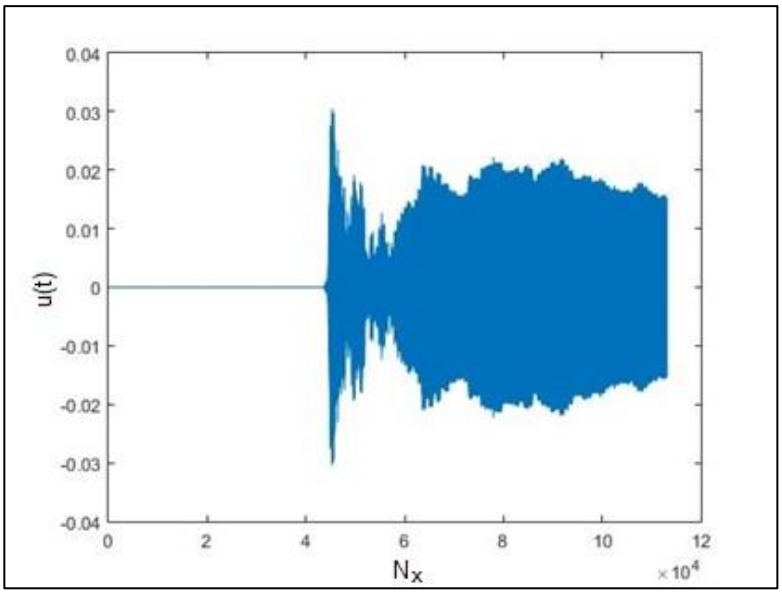
Şekil 4.8: MATLAB ile 50 m derinlik ve 9244 m menzil için hesaplanan işaret.

SOFAR kanalının alt taraftan son noktası olarak 100 m derinlik ve 2311 m menzil için hesaplanan işaret Şekil 4.9'da verilmiştir.



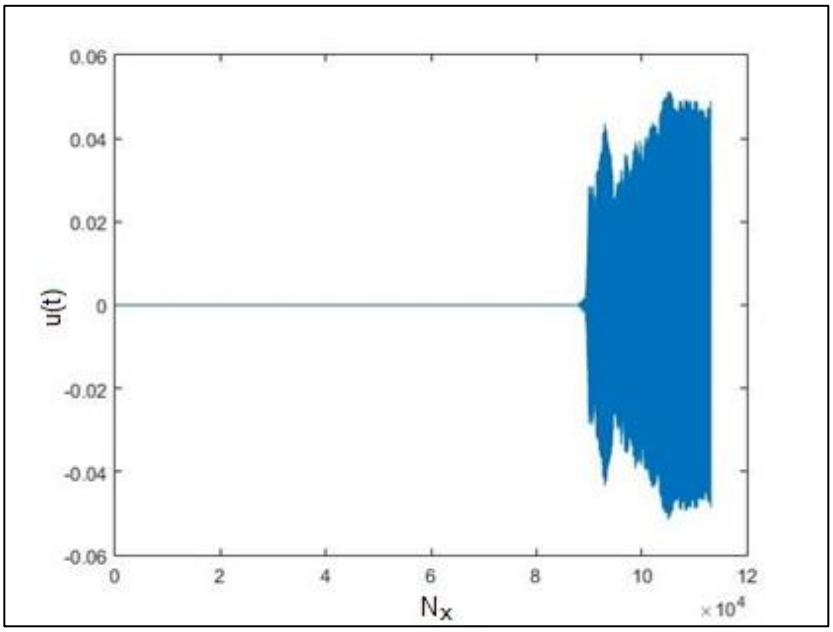
Şekil 4.9: MATLAB ile 100 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 100 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.10'da verilmiştir.



Şekil 4.10: MATLAB ile 100 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 100 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.11'de verilmiştir.



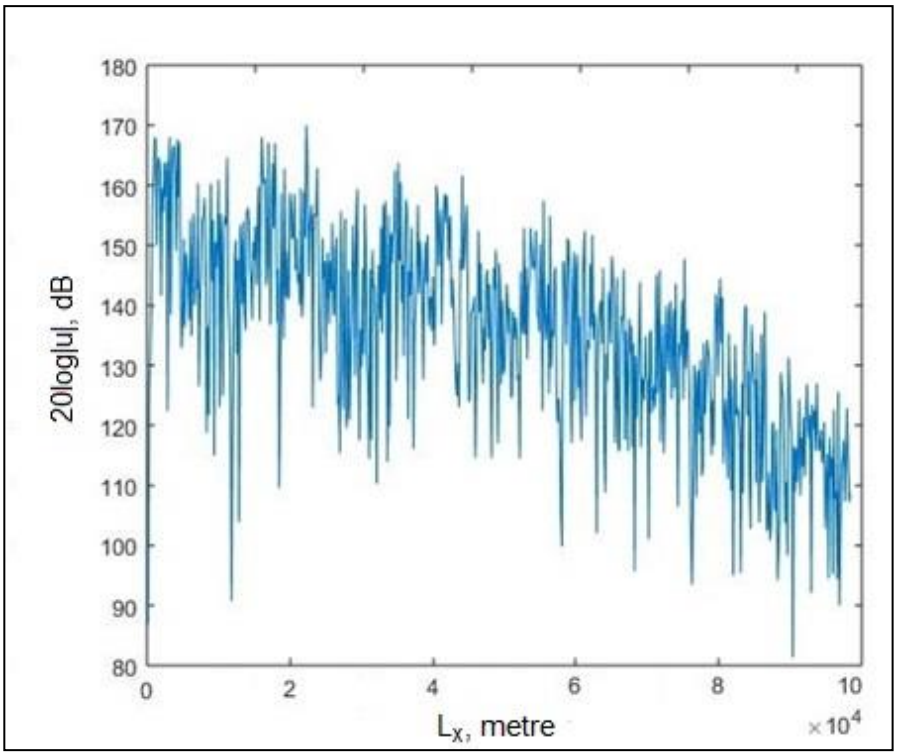
Şekil 4.11: MATLAB ile 100 m derinlik ve 9244 m menzil için hesaplanan işaret.

Çeşitli noktalardan hesaplanan işaretlerin zamana göre değişim grafikleri incelendiğinde beklenildiği gibi ses hızının minimum olduğu nokta hizasından alınan işaretlerin, SOFAR kanalının başlangıç ve bitiş noktalarından alınan işaretlere göre daha az kayba uğrayarak iletildiği gözlenmiştir.

- İletim Kaybı Sonuçları

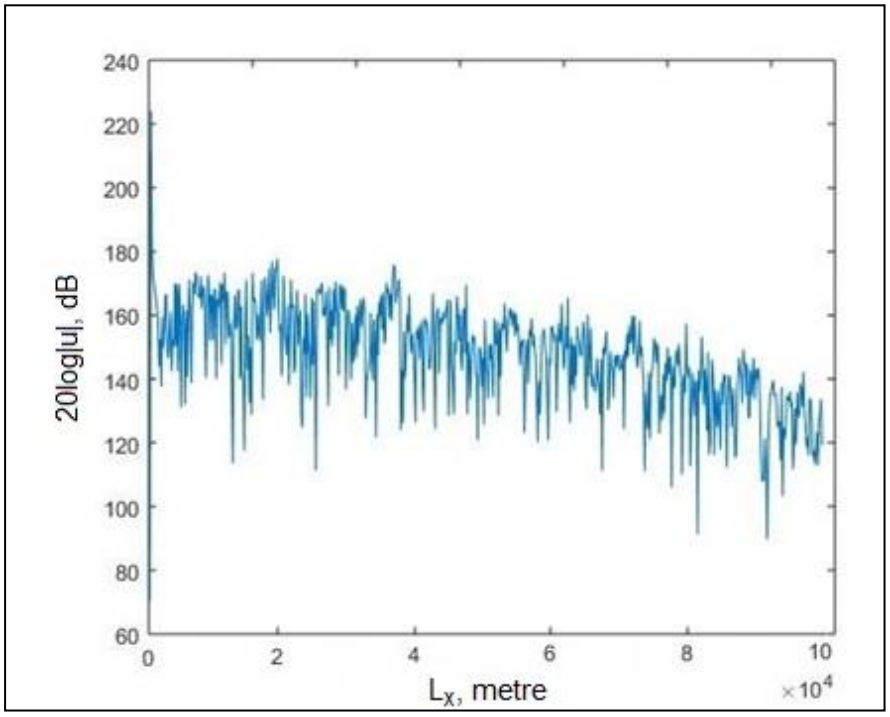
İletim kaybı grafiği SOFAR kanalının ZUSF'de doğru olarak uyarıldığının ispatı açısından önemlidir. Bunun için menzilde 100'er nokta aralıklar bırakılarak örnekleme yapılarak belirlenen noktalarda hesaplanan zaman uzayı işaretlerinin FFT'leri alınıp, daha sonra elde edilen sonucun mutlak değerinin maksimum olduğu değerin $20log$'sı alınarak yayılım kaybı grafikleri konuma bağlı olarak elde edilmiştir. İletim kaybı grafiği, SOFAR kanalının başlangıç noktasından, ses hızının minimum olduğu noktadan ve kanalın bitiş noktasından alınan derinliklerde hesaplanmıştır.

Buna göre tezde kullanılan SOFAR kanalının başlangıç noktası olan 20 m derinlikte menzile bağlı iletim kaybı grafiği Şekil 4.12'de verilmiştir.



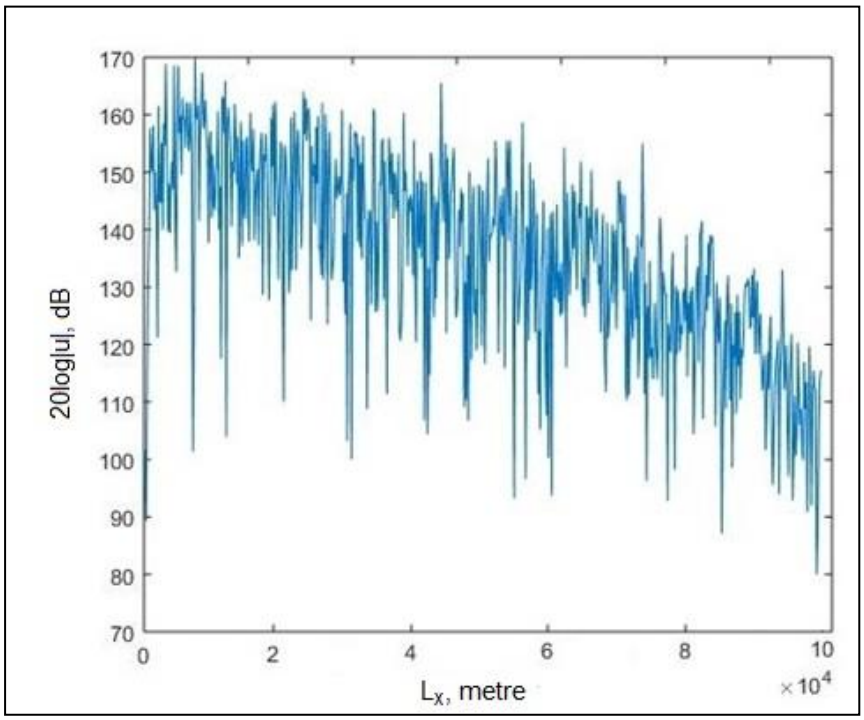
Şekil 4.12: MATLAB ile 20 m derinlikte iletim kaybı hesabı.

SOFAR kanalında ses hızının en düşük olduğu derinlik olan 50 m'den alınan iletim kaybı grafiği Şekil 4.13'de verilmiştir.



Şekil 4.13: MATLAB ile 50 m derinlikte iletim kaybı.

SOFAR kanalının bitişi olan 100 m derinlikte menzile bağlı iletim kaybı grafiği Şekil 4.14'de verilmiştir.



Şekil 4.14: MATLAB ile 100 m derinlikte iletim kaybı hesabı.

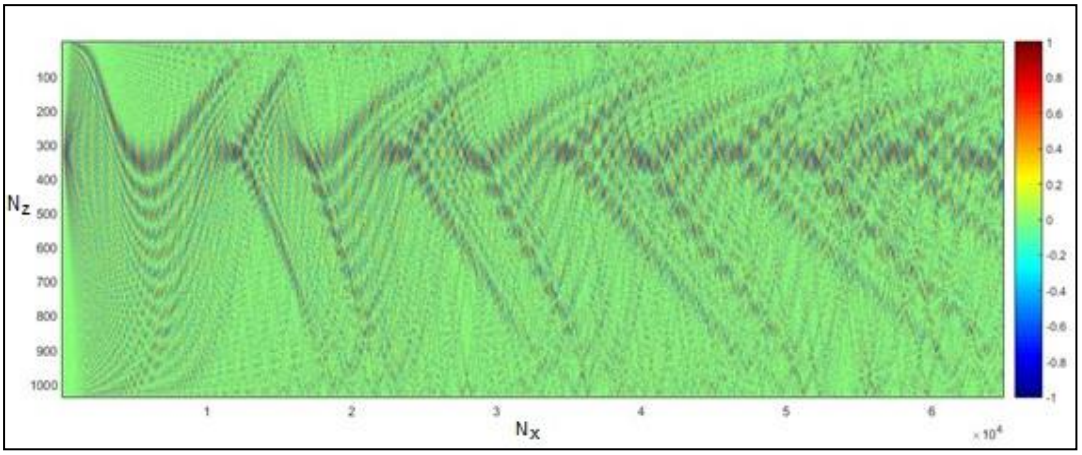
4.1.2. CUDA ile Paralelleştirme Sonuçları

CUDA ile problem uzayı çok parçaya bölünerek her bir parçanın bir CUDA çekirdeğinde hesabının yapılması ile hızlanma elde edilmektedir. Buna göre yapılan denemelerde problem uzayının 128×2 parçaya bölünmesiyle en yüksek hızın elde edildiği görülmüştür.

1 kHz'lik kaynak işareti için çözülen problem için geçen süre 155 dakika, kullanılan RAM miktarı ise 1.3 GB olmuştur.

- Alan Dağılımı

SOFAR kanalının CUDA paralelleştirilerek hesaplanan sonucu Şekil 4.15'te ses hızının minimum olduğu derinlikte gönderilen sürekli zamanlı sinüsoidal işaret sonucu olarak gösterilmiştir.

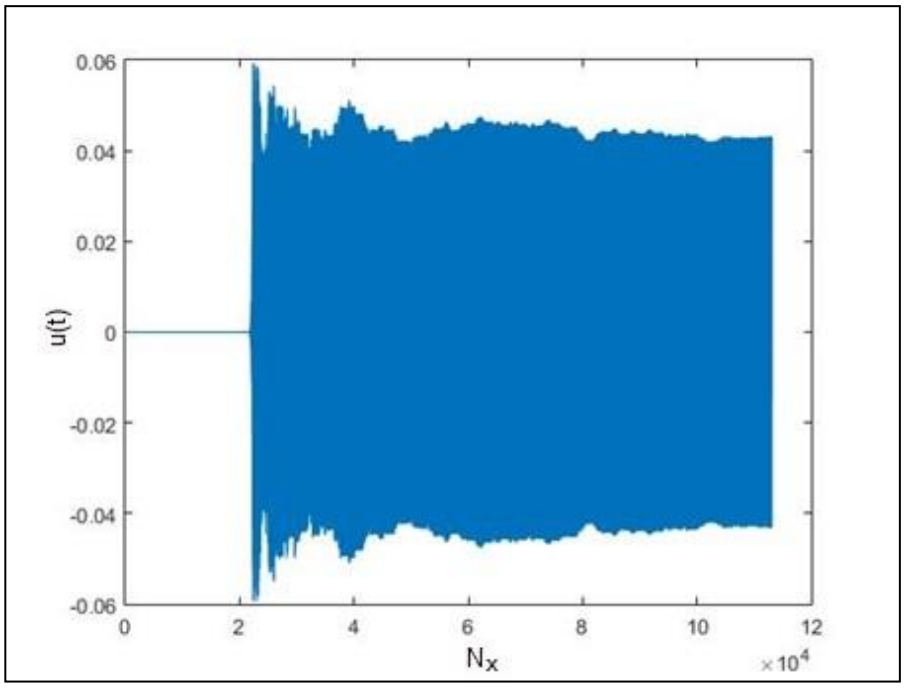


Şekil 4.15: CUDA ile paralel hesaplanan alan dağılımı grafiği.

Buradan gözükeceği üzere SOFAR kanalı davranışı genel olarak açıktır.

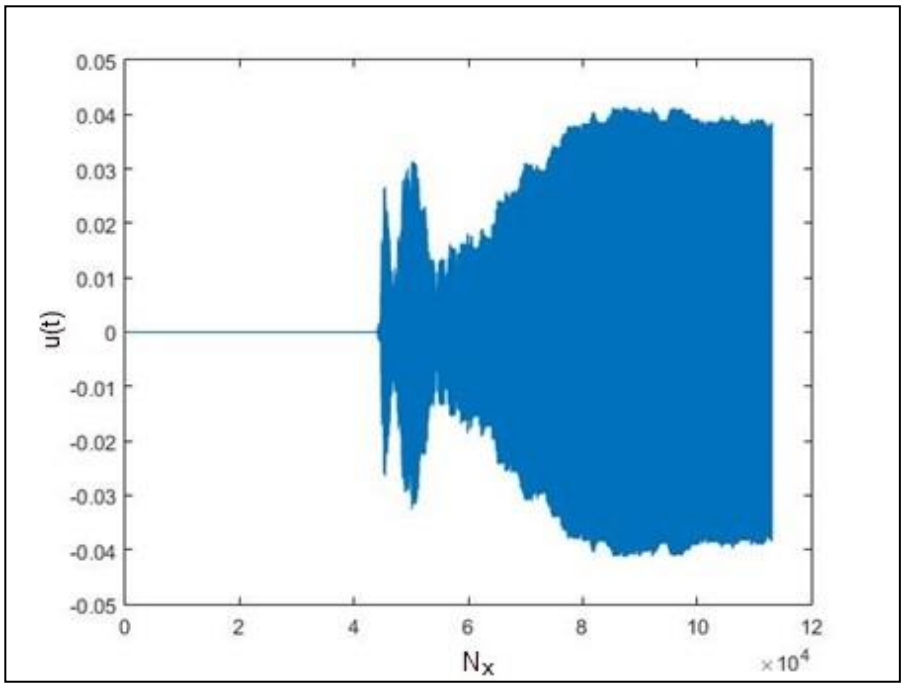
- Çeşitli Noktalardan Alınan İşaret Örnekleri

Kaynaktan iletilen işaret için problem uzayında alıcı noktanın konumuna göre alınan işaretin grafiği elde edilmiştir. Buna göre kaynağa yakın olan noktalarda işaret kısa süre sonra gözlenirken, kaynaktan uzak olan mesafelerde işaretin gözlenme süresi de artmaktadır. Ayrıca ses hızı profiline göre alınan işaretlerin şiddetleri de farklıdır. Ses hızının minimum olduğu SOFAR kanalının merkez hizasından alınan örneklerde işaretin bütünlüğünü koruması, SOFAR kanalının başlangıç ve bitiş noktalarından alınan işaretlerde ise bir miktar bozulma beklenmektedir. Buna göre 20 m derinlik ve 2311 m menzil için hesaplanan işaret Şekil 4.16'da verilmiştir.



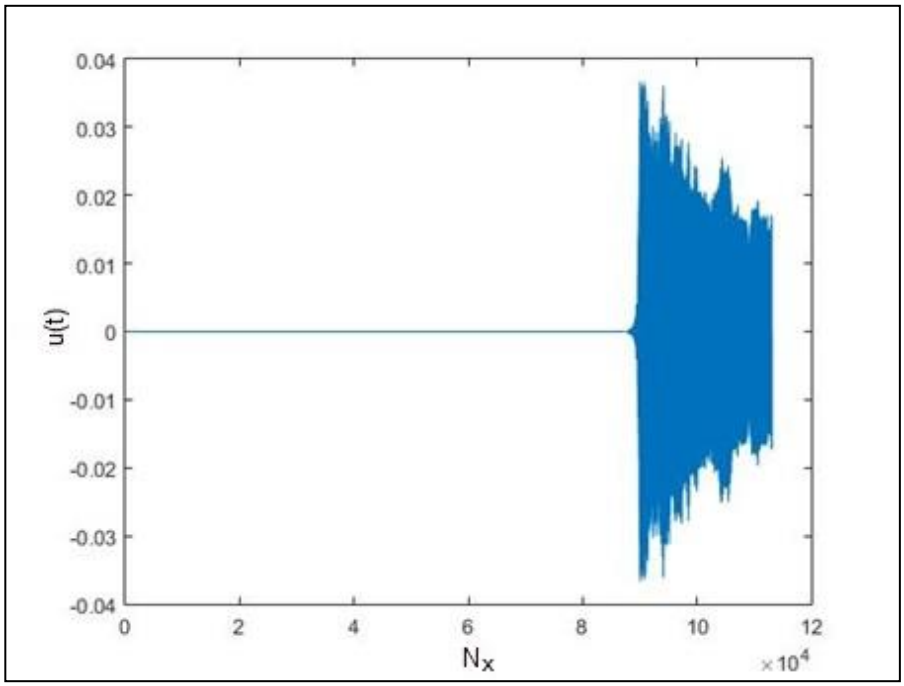
Şekil 4.16: CUDA ile 20 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 20 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.17'de verilmiştir.



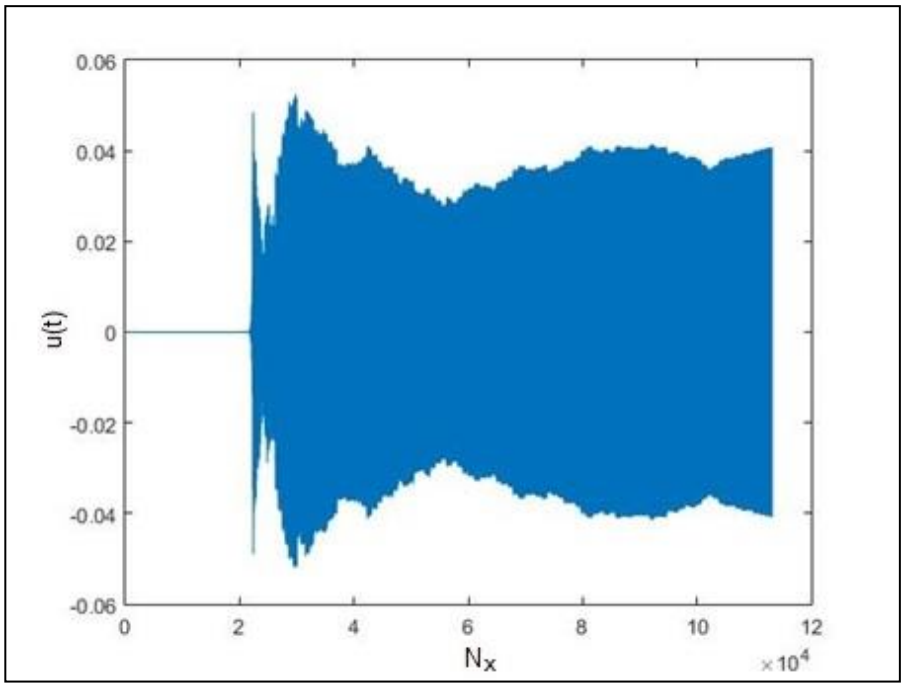
Şekil 4.17: CUDA ile 20 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 20 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.18'de verilmiştir.



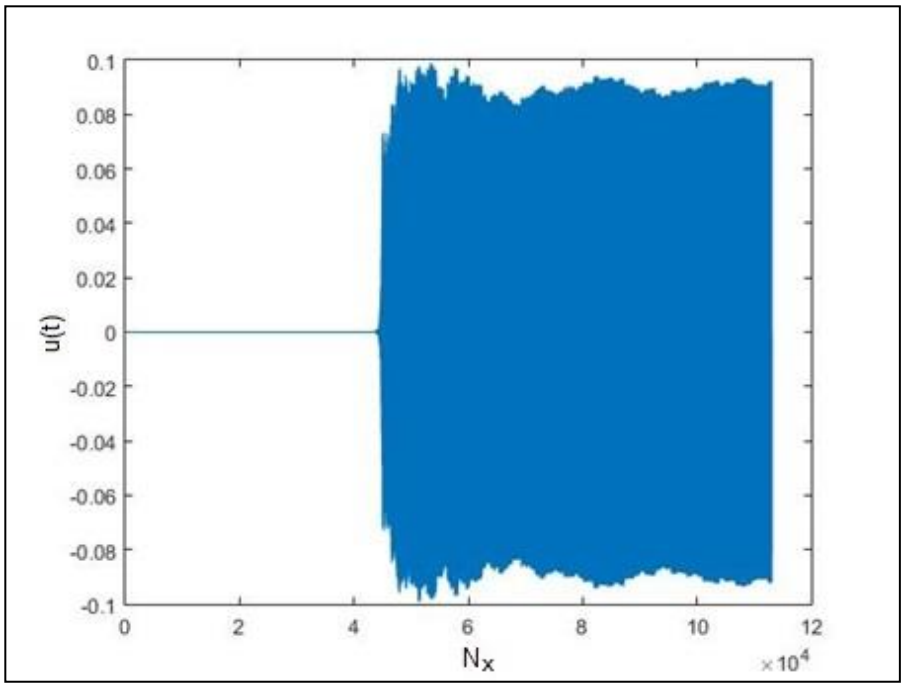
Şekil 4.18: CUDA ile 20 m derinlik ve 9244 m menzil için hesaplanan işaret.

SOFAR kanalında ses hızının minimum olduğu derinlik olan 50 m'de alınan işaretler için sonuçlar aşağıda verilmiştir. Buna göre 50 m derinlik ve 2311 m menzil için hesaplanan işaret Şekil 4.19'da verilmiştir.



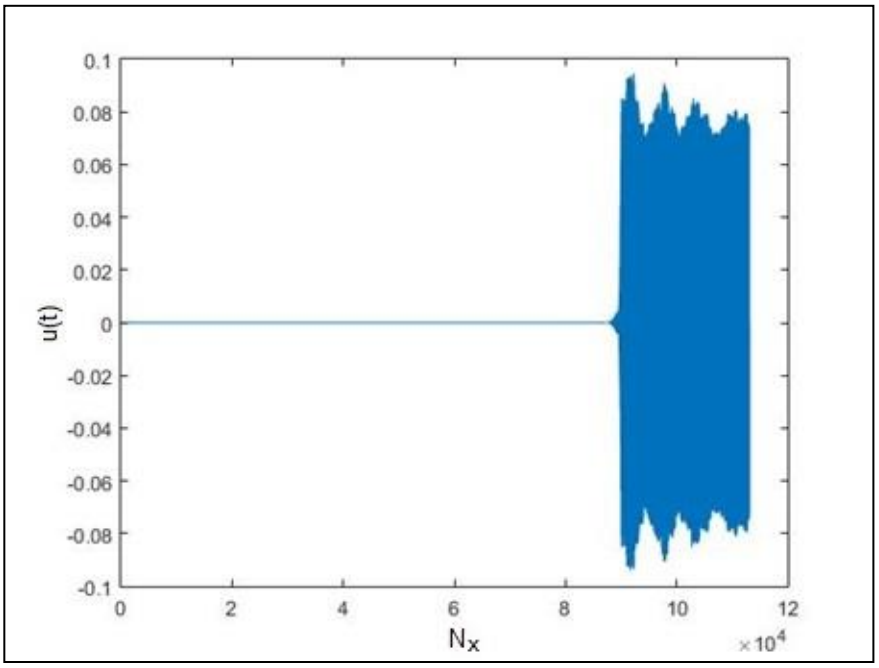
Şekil 4.19: CUDA ile 50 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 50 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.20'de verilmiştir.



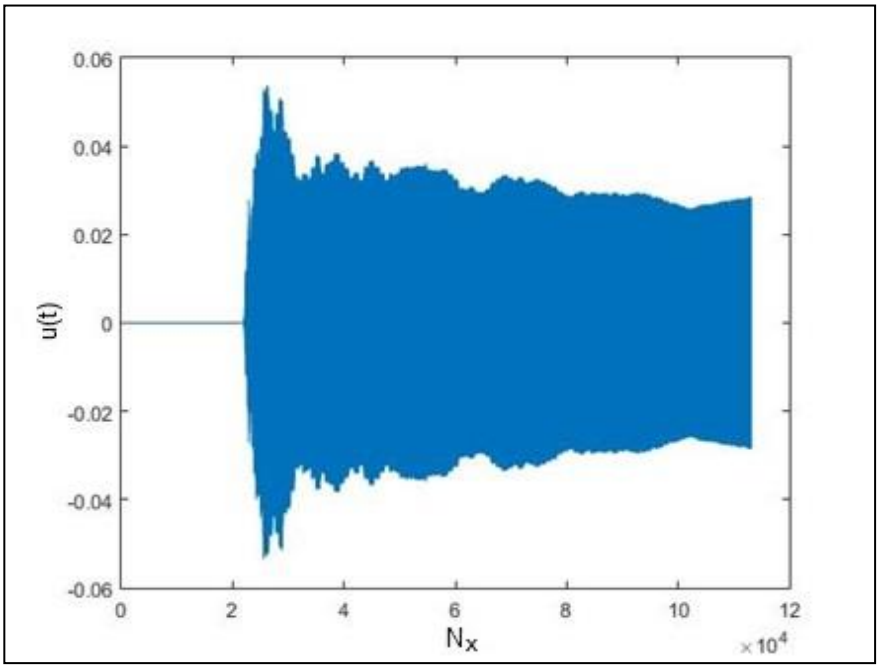
Şekil 4.20: CUDA ile 50 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 50 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.21'de verilmiştir.



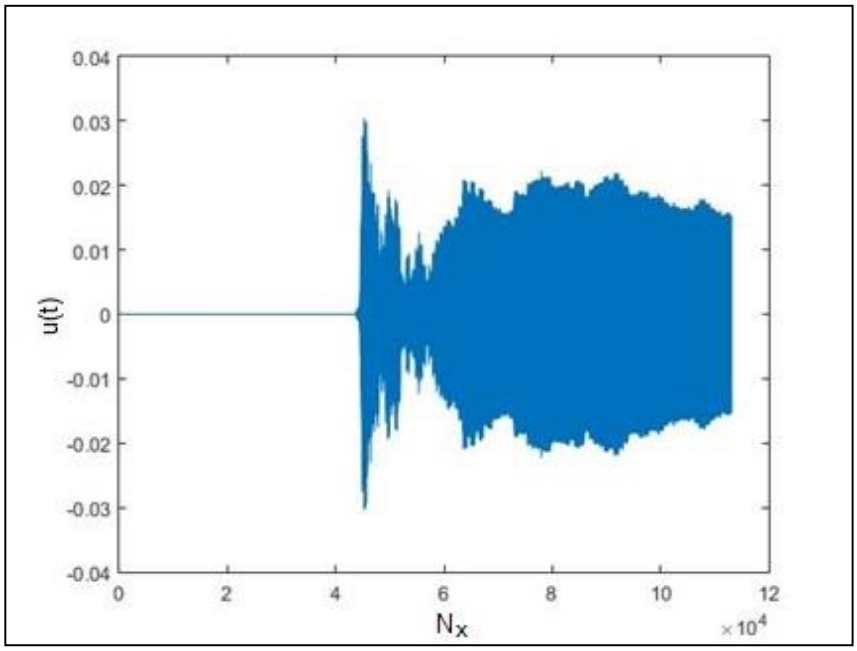
Şekil 4.21: CUDA ile 50 m derinlik ve 9244 m menzil için hesaplanan işaret.

SOFAR kanalının bitişinin derinliği olan 100 m derinlik ve 2311 m menzil için hesaplanan işaret Şekil 4.22'de verilmiştir.



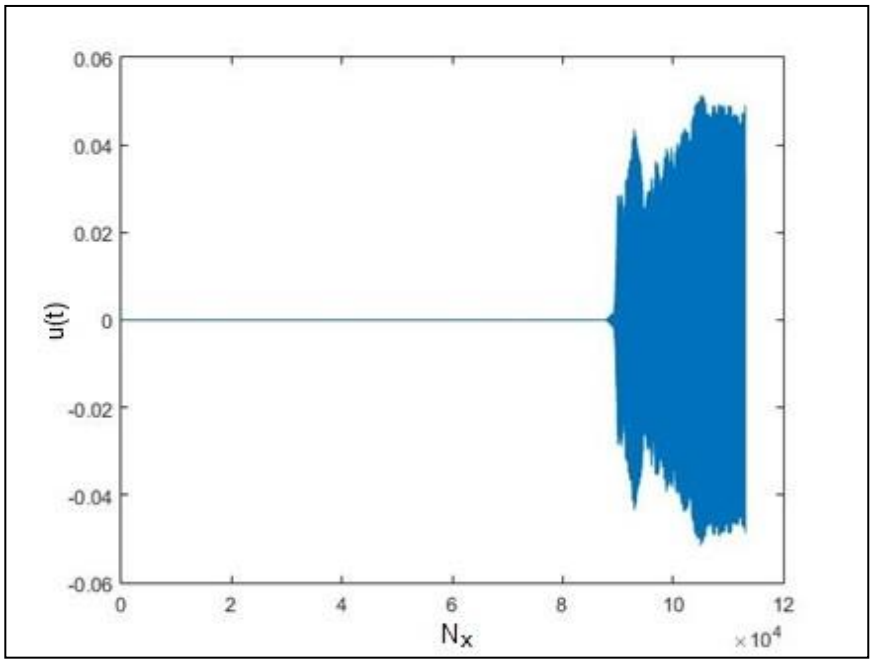
Şekil 4.22: CUDA ile 100 m derinlik ve 2311 m menzil için hesaplanan işaret.

Benzer şekilde 100 m derinlik ve 4622 m menzil için hesaplanan işaret Şekil 4.23'de verilmiştir.



Şekil 4.23: CUDA ile 100 m derinlik ve 4622 m menzil için hesaplanan işaret.

Benzer şekilde 100 m derinlik ve 9244 m menzil için hesaplanan işaret Şekil 4.24'de verilmiştir.

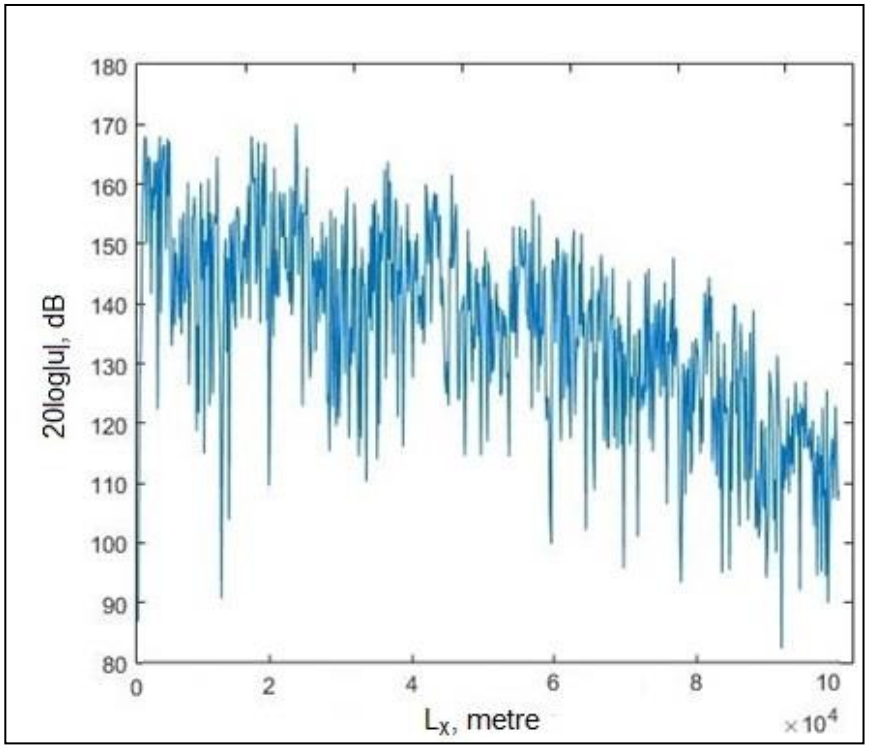


Şekil 4.24: CUDA ile 100 m derinlik ve 9244 m menzil için hesaplanan işaret.

CUDA ile çeşitli noktalardan hesaplanan işaretlerin zamana göre değişim grafikleri incelendiğinde MATLAB ile elde edilen sonuçlar ile aynı olduğu açıkça gözükmektedir.

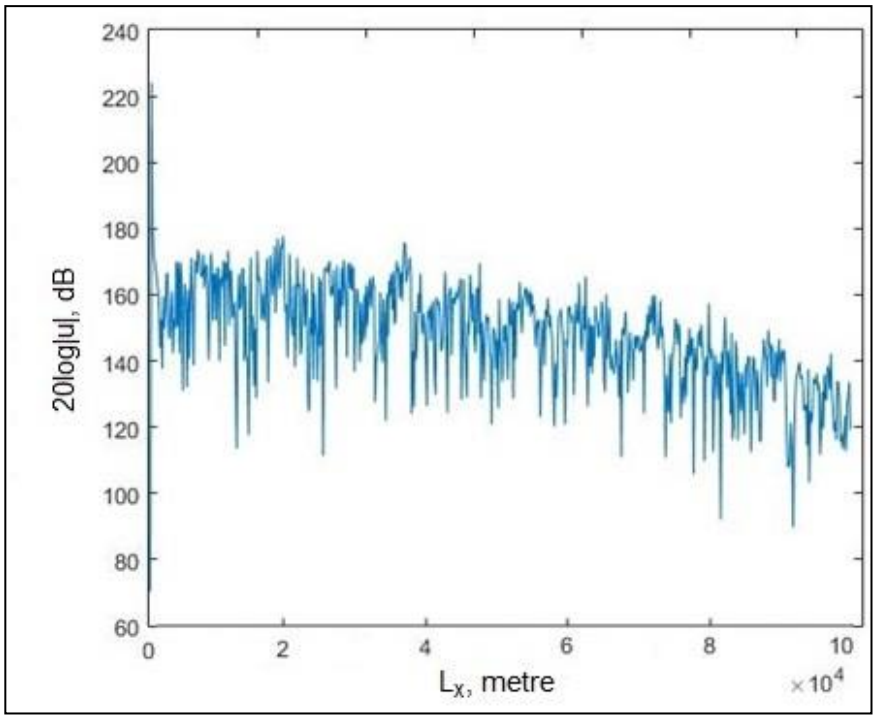
- İletim Kaybı Sonuçları

İletim kaybı grafiği, SOFAR kanalının başlangıç noktasından, ses hızının minimum olduğu derinlik ve kanalın bitiş noktasından alınan derinliklerde hesaplanmıştır. Buna göre bu tez için modifiye SOFAR kanalının başlangıç derinliği olan 20 m'den alınan iletim kaybı grafiği Şekil 4.25'de verilmiştir.



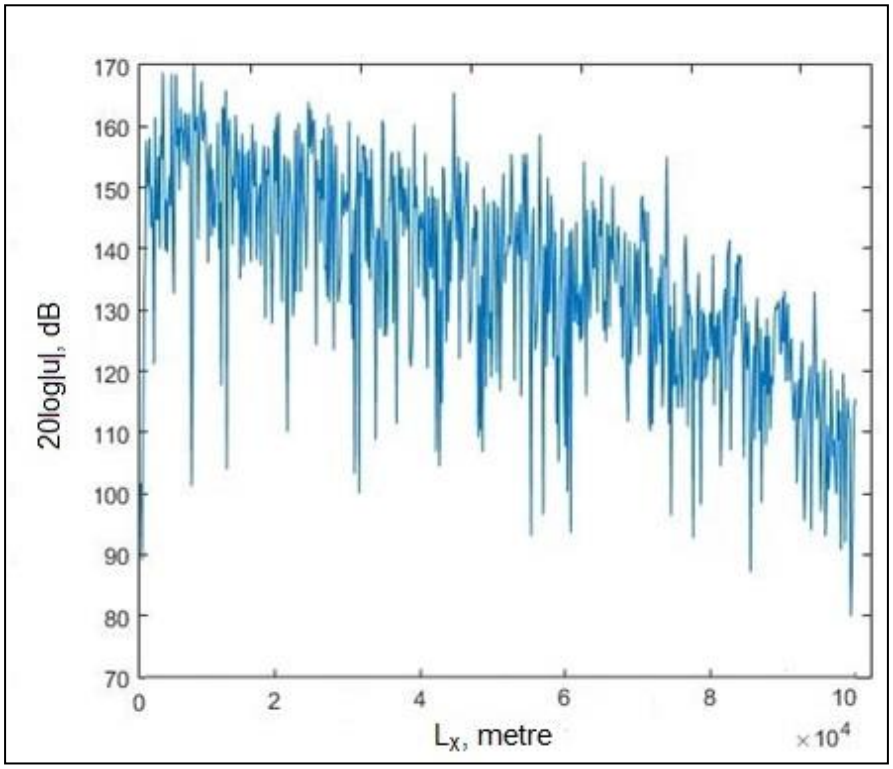
Şekil 4.25: CUDA ile 20 m derinlikte iletim kaybı hesabı.

SOFAR kanalında ses hızının en düşük olduğu derinlik olan 50 m'de hesaplanan iletim kaybı grafiği Şekil 4.26'da verilmiştir.



Şekil 4.26: CUDA ile 50 m derinlikte iletim kaybı hesabı.

SOFAR kanalının sonu olan 100 m derinlikte hesaplanan iletim kaybı grafiği Şekil 4.27'de verilmiştir.



Şekil 4.27: CUDA ile 100 m derinlikte yayılım kaybı hesabı.

4.1.3. Literatüre Göre Sonuçların Doğrulanması

MATLAB ile elde edilen Şekil 4.2'deki SOFAR kanalı alan dağılımı grafiğinin literatürde Şekil 1.2 ile gösterilen SOFAR kanalındaki alan dağılımı grafiği olduğu gözükmektedir. İletim kaybı grafiği ele alındığında literatürde Şekil 1.5, Şekil 1.6 ve Şekil 1.7 ile gösterilen SOFAR kanalındaki iletim kaybı grafikleri ile deneysel olarak elde edilmiş Şekil 4.12, Şekil 4.13 ve Şekil 4.14 grafiklerinin benzediği gözükmektedir. Bu grafiklere bakıldığında MATLAB çözümünün doğru çalıştığı sonucu ortaya çıkmıştır.

MATLAB ile CUDA arasındaki doğrulama aşamasında ise 4.1.1. ve 4.1.2 bölümlerinde elde edilen SOFAR alan dağılımı grafiği, çeşitli noktalardan hesaplanan zaman uzayı işaret örneği sonuçları ve iletim kaybı grafiği sonuçlarının birebir aynı olduğu görülmektedir. Böylelikle hem MATLAB ile hem CUDA ile çözümlerin doğru çalıştığı sonucu çıkmıştır. MATLAB ile CUDA arasında çalışma süresi ve RAM kullanımı bakımından farklılıklar bulunmaktadır. Tablo 4.1'de elde edilen sonuçlar çalışma süresi, RAM kullanımı ve hızlanma ölçütlerine göre verilmiştir.

Tablo 4.1: Süre ve hafıza kullanımı sonuçları.

Platform	Çalışma Süresi	RAM	Hızlanma
MATLAB	64,22 saat	1.7 GB	1
CUDA	2,58 saat	1.3 GB	24

Tablo 4.1'e göre CUDA ile 24 kat daha hızlı sonuç elde edilmiştir. Bunun yanı sıra CUDA çözümünün MATLAB ile elde edilen çözüme göre 400 MB daha az RAM kullandığı görülmüştür.

4.2. Farklı Frekanslar için GPU Çözümü Sonuçları

Bu bölümde farklı monokromatik kaynak frekansları için GPU çözümünün ihtiyaç duyduğu çalışma süresi ve RAM kullanımı incelenmiştir. 1000 Hz için önceki bölümde yapılan çözümlere ek olarak (4.1) denklemindeki kaynak işaretinin 1500 Hz ile 1940 Hz'lik frekanslar için hazırlanan program çalıştırılmış ve sonuçlar elde edilmiştir. 1940 Hz kullanılan GPU donanımında aynı fiziksel boyuttaki problemde

çözülebilecek en büyük frekans değeridir. Daha büyük frekans seçilmesi durumunda kullanılan GPU'nun VRAM'i yetersiz kaldığından, çözüm elde edilememektedir.

4.2.1. 1500 Hz'lik Kaynak İşareti

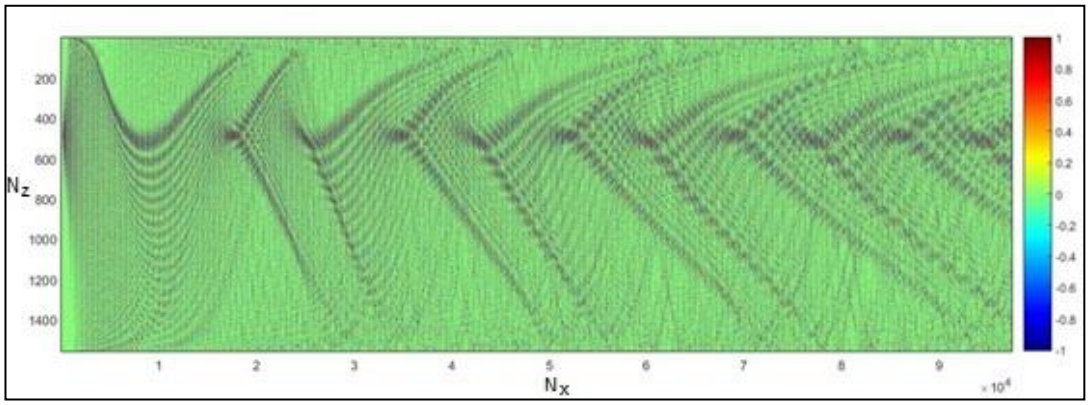
Kaynak işaretinde kullanılan dalganın frekansın 1500 Hz iken (2.18)-(2.21) eşitlikleri kullanılarak $\Delta x = \Delta z = 0.10268$ m, $\lambda = 1.0268$ m ve $N_x = 97387$, $N_z = 1559$ değerleri bulunmuştur. Buna göre problemin elektriksel uzunluğu

- x yönünde: 9738,3 λ - 10273,9 λ,
- z yönünde: 155.8 λ - 164.3 λ

olarak hesaplanmıştır. Buna göre CUDA kodu çalıştırıldığında 6.33 saatlik hesap süresi ve 2.4 GB VRAM kullanımı yeterli olmuştur. Sonuçlarla ilgili SOFAR alan dağılımı ve iletim kaybı grafiği aşağıda verilmiştir.

- Alan Dağılımı

1500 Hz frekansında çalışan kaynak işareti için elde edilen SOFAR kanalı Şekil 4.28'de gösterilmiştir.

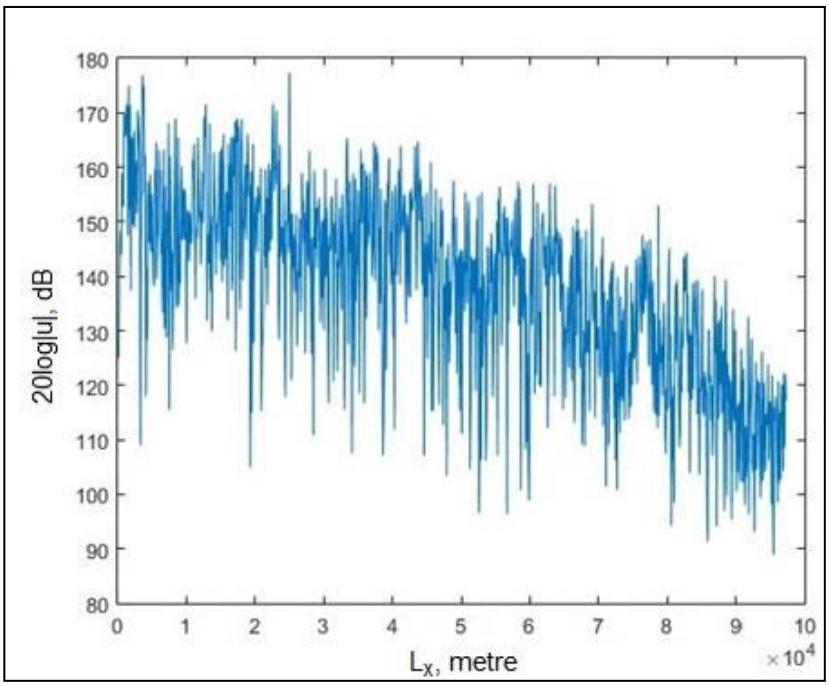


Şekil 4.28: CUDA ile 1500 Hz kaynak frekansı için SOFAR alan dağılımı.

• İletim Kaybı Sonuçları

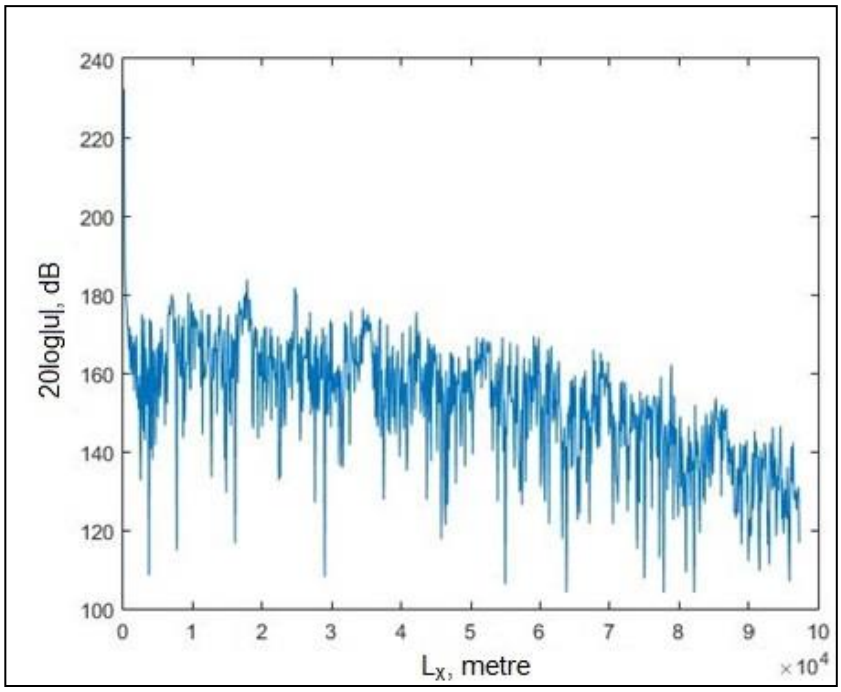
Menzile bağlı iletim kaybı değişimi 1500 Hz frekansında kaynak işaretinin olduğu problemde SOFAR kanalının başlangıç noktasından, ses hızının minimum olduğu derinlikten ve kanalın bitiş noktasından alınan örneklerle elde edilmiştir.

Buna göre bu tez çalışmasında modifiye edilmiş SOFAR kanalının başlangıç noktası olan 20 m derinlikte hesaplanan iletim kaybı grafiği Şekil 4.29'da verilmiştir.



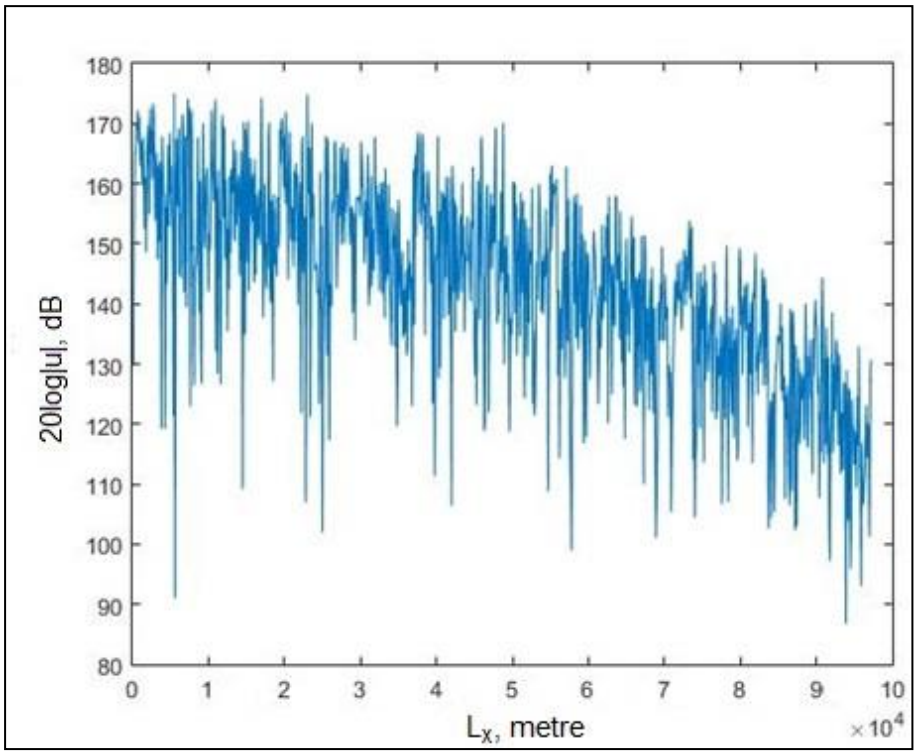
Şekil 4.29: CUDA ile 1500 Hz kaynak frekansı için 20 m derinlikte iletim kaybı.

SOFAR kanalında ses hızının en düşük olduğu derinlik olan 50 m'de hesaplanan iletim kaybı grafiği Şekil 4.30'de verilmiştir.



Şekil 4.30: CUDA ile 1500 Hz kaynak frekansı için 50 m derinlikte iletim kaybı.

SOFAR kanalının sonu olan 100 m derinlikte hesaplanan iletim kaybı grafiği Şekil 4.31'de verilmiştir.



Şekil 4.31: CUDA ile 1500 Hz kaynak frekansı için 100 m derinlikte iletim kaybı.

4.2.2. 1940 Hz'lik Kaynak işareti

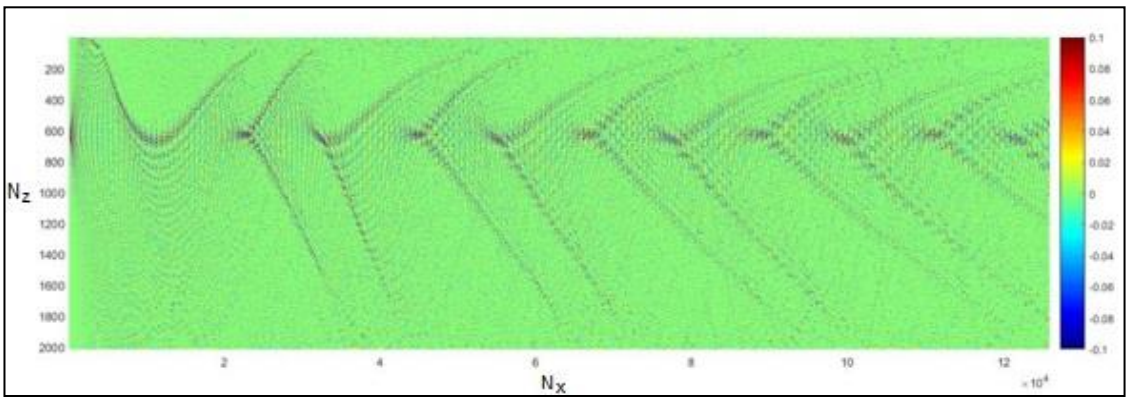
Kaynak için kullanılan frekansın 1940 Hz iken (2.18)-(2.21) eşitlikleri kullanılarak $\Delta x = \Delta z =$ 0.079395 m, $\lambda = 0.79395$ m ve $N_x = 125953$, $N_z = 2016$ değerleri bulunmuştur. Buna göre problemin elektriksel uzunluğu

- x yönünde: 12594,9 λ - 13287,6 λ,
- z yönünde: 201,.5 λ - 212.6 λ

olarak hesaplanmıştır. Buna göre CUDA kodu çalıştırıldığında 13.74 saatlik hesap süresi ve 3.9 GB VRAM kullanımını yeterli olmaktadır. Sonuçlarla ilgili SOFAR alan dağılımı ve iletim kaybı grafiği aşağıda verilmiştir.

• Alan Dağılımı

Şekil 4.32'de 1940 Hz frekansında çalışan kaynak işareti için elde edilen SOFAR kanalı gösterilmiştir.

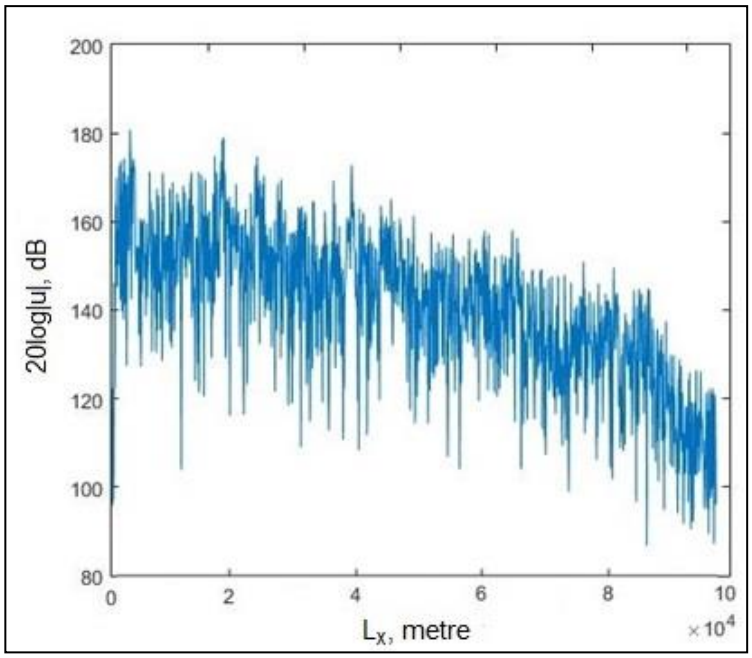


Şekil 4.32: CUDA ile 1940 Hz kaynak frekansı için SOFAR alan dağılımı.

• İletim Kaybı Sonuçları

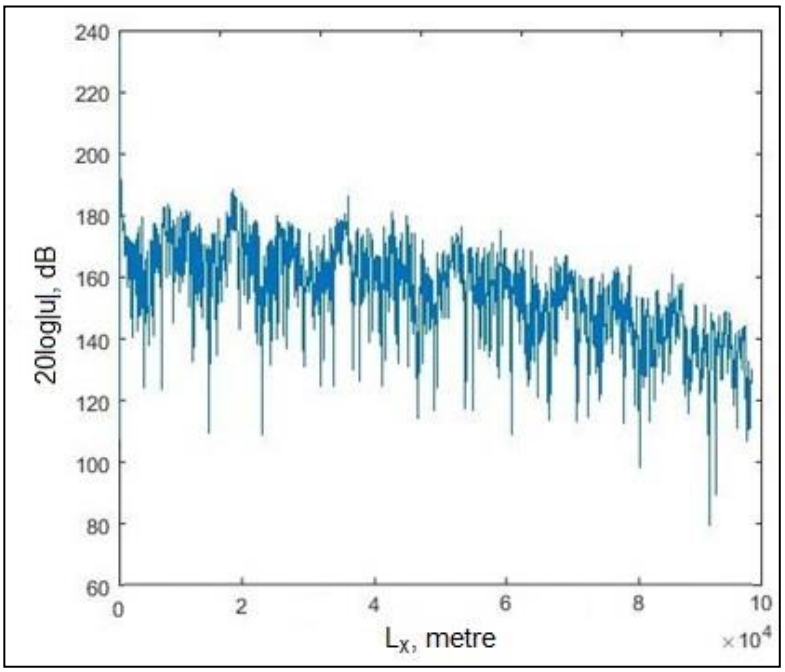
Menzile bağlı iletim kaybı değişimi, 1940 Hz frekansında kaynak işaretinin olduğu problemde SOFAR kanalının başlangıç noktasından, ses hızının minimum olduğu derinlik ve kanalın bitiş noktasından alınan örneklerle elde edilmiştir.

Buna göre bu tez çalışmasında modifiye edilmiş SOFAR kanalının başlangıç noktası olan 20 m derinlikte hesaplanan alınan iletim kaybı grafiği Şekil 4.33'de gösterilmiştir.



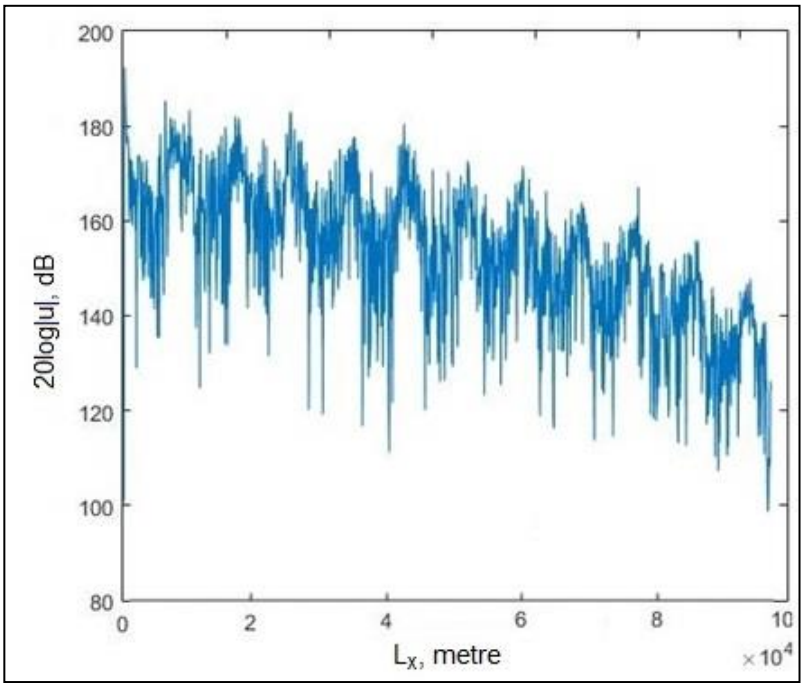
Şekil 4.33: CUDA ile 1940 Hz kaynak frekansı için 20 m derinlikte iletim kaybı.

SOFAR kanalında ses hızının en düşük olduğu nokta olan 50 m derinlikte hesaplanan iletim kaybı grafiği Şekil 4.34'de gösterilmiştir.



Şekil 4.34: CUDA ile 1940 Hz kaynak frekansı için 50 m derinlikte iletim kaybı.

SOFAR kanalının sonu olan 100 m derinlikte hesaplanan iletim kaybı grafiği Şekil 4.35'de gösterilmiştir.



Şekil 4.35: CUDA ile 1940 Hz kaynak frekansı için 100 m derinlikte iletim kaybı.

4.2.3. Performans Değerlerinin Karşılaştırılması

Bölüm 4.2.1 ve 4.2.2'de incelenen 1500 Hz ve 1940 Hz frekanslı kaynaklar ile bölüm 4.1.2'de incelenen 1000 Hz frekanslı kaynak için CUDA'da çözülen problemlere ait hesap süresi ve kullanılan RAM miktarı Tablo 4.2'de gösterilmiştir.

Tablo 4.2: Farklı frekans değerleri için ölçüm sonuçları

Frekans	Nokta Sayısı	Çalışma Süresi	RAM
1000 Hz	$N_x \times N_z = 64925 \times 1039$	2,58 saat	1,3 GB
1500 Hz	$N_x \times N_z = 97387 \times 1559$	6,33 saat	2,4 GB
1940 Hz	$N_x \times N_z = 125953 \times 2016$	13,74 saat	3,9 GB

Tablo 4.2'ye bakıldığında frekans artış oranı ile hesap yapılan nokta sayısının artış oranının aynı olduğu gözükmektedir. Çalışma süresi ve RAM kullanımı

sonuçları da frekans artışıyla beraber farklı oranlarda artış hızı göstermektedir. İncelenen problem iki boyutlu problem olduğu için artış hızı oranları da karesel olma eğilimi göstermektedir. Frekans değeri yükseldikçe çalışma süresinin arttığı ancak çalışma süresi artış hızının düştüğü görülmektedir. RAM kullanımı bakımından frekans değeri arttıkça hem RAM kullanımının hem de RAM kullanımı artış hızının arttığı görülmüştür.

5. SONUÇLAR ve GELECEK ÖNERİLERİ

Bu tez çalışmasında su altı akustik dalga yayılımı probleminin ZUSF yöntemiyle GPU üzerinde hızlandırma amacıyla çözümü yapılmıştır. SOFAR kanalı probleminin çözüldüğü bu çalışmada sonuçlar iki aşamada değerlendirilmiştir. İlk aşamada MATLAB ve CUDA ile elde edilen sonuçların literatürde yer alan SOFAR kanalı sonuçlarıyla olan tutarlılığı kontrol edilerek elde edilen hızlanmalar incelenmiştir. Buna göre CUDA paralelleştirme ile 24 kat daha hızlı hesap yapılabilmektedir.

Değerlendirmenin ikinci aşamasında farklı frekans değerleri için CUDA paralelleştirme ile hesaplama yapılarak problem çözüm süresinin ve kullanılan hafıza miktarının frekansa bağlı olarak değişimi incelenmiştir. Buna göre frekans değeri arttıkça problemin çözüm süresi ve kullanılan hafıza miktarının arttığı sonucu ortaya çıkmıştır.

Geleceğe yönelik çalışmalarda CUDA'da yer alan paylaşımlı bellek, yapısal bellek, yazma belleği gibi farklı bellek türlerinin kullanılmasıyla hızlı veri transferi yapılarak problemin daha da hızlı çözülmesi sağlanabilir. Donanımsal olarak sistemde birden fazla GPU'nun ortak kullanılmasıyla daha büyük problemlerin hızlı olarak çözümü yapılabilir. Ayrıca MATLAB ile GPU kullanımı sağlanarak CUDA ve MATLAB'ın GPU kullanım başarısının ölçülmesi çalışması da yapılabilir. NVIDIA ekran kartlarına olan bağımlılığı azaltmak için ise CUDA arayüzü yerine OpenCL programlama dili kullanılarak farklı üreticilerce üretilmiş ekran kartlarıyla da problem çözümü gerçekleştirilebilir.

Yine güvenilir akustik yol, yüzey kanal yayılımı, toplanma bölgesi yayılımı, dipten sekme yayılımı vb. farklı su altı akustik problemlerinin özellikle sonar tespit optimizasyonu amaçlı olarak bu yöntemle çözümü önemli bir açılım sağlayacaktır.

KAYNAKLAR

Aksoy S., (2019), "Zaman Uzayı Sonlu Farklar Yöntemi, Ders Notları", Elektronik Mühendisliği Bölümü, Gebze Teknik Üniversitesi, Gebze, Kocaeli.

Balevic A., Rockstroh L., Hillebrand W. L., Simon S., Tausendfreund A., Patzelt S., Goch G., (2008), "Acceleration of a finite-difference method with general purpose GPUs - Lesson learned", 2008 8th IEEE International Conference on Computer and Information Technology, 291-294, Sydney, NSW, 8-11 July.

Brandao D., Zamith M., Clua E., Montenegro A., Bulcao A., Madeira D., Kischinhevsky M., Leal-Toledo R., (2010), "Performance evaluation of optimized implementations of finite difference method for wave propagation problems on GPU architecture", 22nd International Symposium on Computer Architecture and High Performance Computing Workshops, 7-12.

Kutschale H., (1961), "Long range sound transmission in the Arctic Ocean", Journal of Geophysical Research, 66-7, 2189-2198.

Mellberg L. E., Johannessen O. M., Connors D. N., Botseas G., Browning D. G., (1991), "Acoustic propagation in the western Greenland Sea frontal zone", The Journal of the Acoustical Society of America, 89, 2144-2156.

Micikevicius P., (2009), "3D finite difference computation on GPUs using CUDA", GPGPU-2 Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, 79 - 84, Washington, D.C., USA, 08 March.

Munk W. H., (1974), "Sound channel in an exponentially stratified ocean, with application to SOFAR", The Journal of the Acoustical Society of America, 55, 220-226.

Nakai S., Ishii T., Tsuchiya T., (2011), "Numerical analysis of sound wave propagation in ocean by WE-FDTD method with GPU cluster system", Proceedings of Symposium on Ultrasonic Electronics, 32, 575-576.

NVIDIA, (2017), "CUDA C Programming Guide", Version 8, NVIDIA.

Raghuvanshi N., Narain R., Lin M. C., (2009), "Efficient and accurate sound propagation using adaptive rectangular decomposition", IEEE Transactions on Visualization and Computer Graphics, 15-5, 789-801.

Saarelma J., Savioja L., (2014), "An open source finite-difference time-domain solver for room acoustics using graphics processing units", Forum Acusticum, 11-8, Krakow, POLAND, 7-12 September.

Savioja L., (2010), "Real-time 3D finite-difference time-domain simulation of low- and mid-frequency room acoustics", Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10), Graz, AUSTRIA, 6-10 September.

Southern A., Murphy D., Campos G., Dias P., (2010), "Finite difference room acoustic modelling on a general purpose graphics processing unit", AES 128th Convention, 8028, 22-25 May.

Spa C., Rey A., Hernández E., (2015), "A GPU implementation of an explicit compact FDTD algorithm with a digital impedance filter for room acoustics applications", IEEE/ACM Transactions on audio, speech, and language processing, 23-8, 1368-1380.

Web 1, (2019) https://www.tomshardware.com/reviews/intel-core-i7-broadwell-e-6950x-6900k-6850k-6800k,4587.html, (Erişim Tarihi: 26.06.2019).

Web 2, (2019), https://devblogs.nvidia.com/parallelforall/inside-volta/, (Erişim Tarihi: 26.06.2019).

Web 3, (2019), https://computer.howstuffworks.com/graphics-card1.htm, (Erişim Tarihi: 26.06.2019).

Web 4, (2019), https://pcisig.com/faq?keys=3.0, (Erişim Tarihi: 26.06.2019).

Web 5, (2019), https://en.wikipedia.org/wiki/CUDA, , (Erişim Tarihi: 28.06.2019).

Web 6, (2019), https://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html, (Erişim Tarihi: 12.07.2019).

Web 7, (2019), https://www.mathworks.com/help/matlab/matlab_ prog/ vectorization.html, (Erişim Tarihi: 06.08.2019).

Webb C. J., Bilbao S., (2011), "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity", IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 317-320, Prague, CZECH REPUBLIC, 22-27 May.

Webb C. J., Bilbao S., (2011), "Virtual room acoustics a comparison of techniques for computing 3D-FDTD schemes using CUDA", AES 130th Convention, 8028, 13 May.

Zamith M. P. M., Brandão D. N., Kischinhesvky M., Leal-Toledo R. C. P., Filho O. T. S., Clua E. W. G., Montenegro A. A., Bulcão A., (2010), "Simulation of wave propagation in semi-infinite domains using the finite difference method on a GPU based on cluster", Asociación Argentina de Mecanica Computacional, 7147-7157.

ÖZGEÇMİŞ

Yunus Emre ALÇAKAKAN 1993 yılında İstanbul'da doğdu. 2011 yılında başladığı Gebze Teknik Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümünü 2015 yılında tamamladı. Aynı üniversitede 2013 yılında girmiş olduğu Bilgisayar Mühendisliği Çift Anadal programından 2016 yılında mezun oldu. 2016 yılında Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalında yüksek lisans eğitimine başladı. Aralık 2016 tarihinden bu yana TÜBİTAK BİLGEM'de yazılım geliştirme mühendisi olarak çalışmaktadır.