T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİMSEL HESAPLAMA ALGORİTMALARININ YAZILIM GERÇEKLEME VE PERFORMANS KARŞILAŞTIRMA ODAKLI TASARIMLARI

MUHAMMED TORUN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2021

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİMSEL HESAPLAMA ALGORİTMALARININ YAZILIM GERÇEKLEME VE PERFORMANS KARŞILAŞTIRMA ODAKLI TASARIMLARI

MUHAMMED TORUN
YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
DR. ÖĞR.ÜYESİ ÖNDER ŞUVAK

GEBZE
2021

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SOFTWARE IMPLEMENTATION AND PERFORMANCE COMPARISON ORIENTED DESIGNS OF SCIENTIFIC COMPUTATIONAL ALGORITHMS

MUHAMMED TORUN

A THESIS SUBMITTED FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRONICS ENGINEERING

THESIS SUPERVISOR

ASSIST. PROF. DR. ÖNDER ŞUVAK

GEBZE

2021

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

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 01/07/2021 tarih ve 2021/30 sayılı kararıyla oluşturulan jüri tarafından 07/07/2021 tarihinde tez savunma sınavı yapılan Muhammed TORUN'un tez çalışması Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	: Dr. Öğr. Üyesi Önder ŞUVAK
ÜYE	: Doç. Dr. Koray KAYABOL
ÜYE	: Doç. Dr. Mustafa Berke YELTEN

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

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

ÖZET

Tez kapsamında, simülasyon yazılımları alanında kullanılabilecek bir araç tasarlanmıştır. Bu konuda literatür araştırmaları, kod tasarımları ve kodlama işlemleri yapılmıştır. Çeşitli simülasyonlar içerisinde sisteme yükü fazla olan nümerik hesaplamaların düşük seviyeli diller arasında yer alan C++ içerisinde gerçekleşmesi ve simülasyon test sonuçlarının yüksek seviyeli bir dil olan Python dilinde görüntülenmesini konu alan proje içerisinde birçok yenilikçi yöntem kullanılmıştır. Tez kapsamında nümerik hesaplamalar ve simülasyon yazılımları için tasarlanmış olan IT++, Armadillo, Boost, ACML (AMD Core Math Library) ve OpenMP gibi kütüphaneler araştırılmıştır ve proje kapsamında kullanılmıştır. Yazılım dünyasında kendilerini kanıtlamış olan bu kütüphaneler, nümerik hesaplamalar yapılırken karşılaşılan birçok probleme cevap oluşturabilmiştir. Proje, temelinde aile dosya format yapısını kurulmuştur. Devamında tasarımı yapılan her bir test için özel dosyalar kullanmıştır. Linux işletim sisteminde tasarlanan projede açık kaynak kodlu kütüphaneler kullanılmıştır. İşletim sistemi içerisinde önemli araçlardan biri olan Makefile yapısı, projenin temelini oluşturmaktadır. Kullanılacak olan kütüphaneler, tasarlanan simülasyon test dosyaları ve son olarak oluşturulan obje dosyaları Makefile aracı sayesinde birbirine bağlanmaktadır. Tasarlanan araç iki ayrı test ile incelenmiştir. Bu testler; iki boyutlu, ZUSF (Zaman Uzayında Sonlu Farklar) yöntemi ile tasarlanmış; “Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi” ve iki boyutlu, ZUSF yöntemi ile tasarlanmış “Fotonik Kristal Doksan Derece Eviren Dalga Kılavuzu Testidir”.

Anahtar Kelimeler: C++, Python, IT++, Armadillo, Boost, ACML.

SUMMARY

Regarding the scope of the thesis, after the literature review and probe in the field of simulation software, code designs and coding processes, a tool that can be used in the simulation software has been designed for this study. Many innovative methods have been used in the project. The project deals with the realization of numerical calculations, which are overloaded on the system in various simulations, through using C ++, which is one of the low-level languages; and the display of simulation test results in Python that is a high-level language. In this thesis project, libraries designed for numerical calculations and simulation software such as IT ++, Armadillo, Boost, ACML (AMD Core Math Library) and OpenMP- have been examined. These libraries could respond to many problems faced while performing numerical calculations, so that they have displayed sound and successful outcomes. The basis of the project relies on the family file format structure and it used special files for each designed tests. Open source libraries have been used in the project whose operating system is Linux. Therefore, the problems to be encountered in using the vehicle designed within the scope of the project in other projects have been prevented. The "Makefile Structure" forms the basis of the project. Because, the libraries to be used, the designed simulation test files and the created object files are linked each other through the Makefile tool. This designed tool has been examined in two different tests. These tests: “2D FDTD (Finite Difference Time Domain) of Waveguide Splitter/Combiner” and “2D FDTD of Photonic Crystal 90 Degree Bent Waveguide”.

Key Words: C++, Python, IT++, Armadillo, Boost, ACML.

TEŞEKKÜR

Başta, yüksek lisans eğitimim boyunca her konuda desteğini benden esirgemeyen değerli hocam Dr. Öğr. Üyesi Önder ŞUVAK'a

Tez sürecimin başından sonuna kadar yanımda olan ve çalışmalarıma kendi doktora tezinde yer veren Arş.Gör. Mecit Emre DUMANA'a

Bünyesinde çalışmakta olduğum TUSAŞ (Türk Havacılık ve Uzay Sanayii) şirketine, en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		**Açıklamalar**
$ACML$	:	AMD core math library
$ZUSF$	:	Zaman Uzayında Sonlu Farklar
$FDTD$	:	Finite Difference Time Domain
MKL	:	Math Kernel Library
2B	:	İki Boyutlu
GCC	:	GNU Compiler Collection
DLL	:	Dynamic Link Library
GTÜ	:	Gebze Teknik Üniversitesi
MB	:	Megabyte
GHz	:	Gigahertz
PML	:	Perfectly Mached Layer
ABC	:	Absorbing Boundary Conditions
MEK	:	Mükemmel Eşlenmiş Katman
ESK	:	Emici Sınır Koşulu
SWIG	:	Simplified Wrapper and Interface Generator

ŞEKİLLER DİZİNİ

1. GİRİŞ

İnsanlık için simülasyonların önemi günümüz dünyasında daha da artmıştır. Simülasyon araçlarının geliştirilmesi ve yeni yöntemlerin keşfedilmesi amacıyla akademi ve iş çevrelerinde çokça çalışmalar mevcuttur. Bu çalışmalar genellikle daha hızlı ve gerçeğe daha yakın simülasyonlar üretmek için yapılmaktadır. Yapılan bu tez çalışmasında simülasyon alanında yeni bir araç geliştirerek daha hızlı ve gerçeğe daha yakın simülasyonlar ortaya konmaya çalışılmıştır. Karşılaştırma yöntemi ile çeşitli kütüphaneler bu tez kapsamında değerlendirilmiştir.

Modern simülasyon programlarının temelinde nümerik hesaplama yöntemleri, efektif lineer cebir klasları ve başka birçok matematiksel alt yapıya sahip araçlar bulunmaktadır. Bu tarz sistemlerde en temel birimler genel olarak vektör ve matrisler olarak görülmektedir. Matlab [28], Octave [27], Python [4] gibi programların temel çıkış noktaları vektör ve matris operasyonlarıdır. Matris ve vektör işlemleri, büyük veri setlerinde çok yavaş gerçekleşebilmektedir. Paket program olarak hazırlanan yazılım araçları, yapıları gereği yazılımcıya esnek bir çalışma ortamı sunamamaktadırlar. Düşük seviyede yazılım iyileştirmesi yapmak isteyen kod geliştiricileri modern simülasyon programlarında istedikleri ortamları oluşturamamaktadırlar. Tez kapsamında tasarlanan yazılım aracı sayesinde, geliştirilen simülasyon kodları istenilen her seviyede müdahale edilebilir hale getirilmiştir. Vektör ve matris işlemlerine farklı yaklaşımlar gösteren birçok kütüphanenin birlikte veya ayrı şekilde kullanabilmesine imkân sağlanmıştır. Proje oluşturduğu "Aile Dosya Yapısı" sayesinde kendi ağaç sistemi oluşturmuştur. Aile dosya yapısı içerisinde kullanıcıya kendi C++ kütüphanesini oluşturabilmesi için sunulan alan ile esnek bir yazılım ortamı oluşturulmuştur. Matris ve vektör kopyalamalarını hızlandırmak için paralel programlama araçları projeye eklenmiştir. Projenin temelinde birden fazla lineer cebir altyapısına sahip kütüphane denenerek en iyi sonuçlar aranmıştır.

Matlab üzerinde yazılmış olan Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi ve iki boyutlu, ZUSF yöntemi ile tasarlanmış Fotonik Kristal Doksan Derece Eviren Dalga Kılavuzu Testi [3] oluşturulan platform üzerinde test edilmiştir.

1.1. Tezin Amacı, Katkısı ve İçeriği

Tez kapsamında yapılan çalışmada daha çok simülasyonlarda yaşanan yavaş çalışma ve doğruluğa yakınlık problemlerine odaklanılmıştır. Temel olarak Python ve Matlab gibi programlar ile gerçekleştirilen simülasyonlar, yenilikçi ve özgürce kod yazımına izin vermemesi dolayısıyla yeni araçlar üretilmesi ihtiyacını doğurmuştur. Tez konusu oluşturulan bu projede Linux araçlarını efektif bir şekilde kullanılmıştır. Projenin ilk kısmında yeni tasarlanacak olan simülasyon aracının temelleri ince ayrıntılarıyla belirlenmiştir. İkinci kısımda ise temelleri belirlenen projenin gerçekleştirme aşaması başlamıştır. Gerçekleştirme sürecinin ilk etabında, kullanılacak temel kütüphaneler kurulmuştur. Bu kütüphaneler IT++ [5], Boost Python [25], Armadillo [19] gibi kütüphanelerdir. Aile dosya formatı altında oluşturulan proje dosyalarının içerisinde makefile [26] araçları kullanılmıştır. Tasarlanan simülasyonlarda kullanılacak olan kütüphaneler birbirine makefile dosyaları ile bağlanmıştır. Aile dosya formatı altında oluşturulan kullanıcının simülasyonlarda kullanmak amacıyla oluşturabileceği kütüphane bölümleri de mevcuttur. Bu kütüphaneler de kendi içeresinde ayrı makefile dosyaları içermektedir. Kütüphanelerin derlenmesi sonucunda oluşturulan paylaşılabilir objeler sayesinde bütün proje uyum içerisinde çalışabilmektedir. Son aşamada derlenen ana projeden oluşan paylaşılabilir obje Python aracına aktarılarak simülasyonlar gerçekleştirilmektedir.

2. YARDIMCI KÜTÜPHANELER VE YAZILIM ARAÇLARI

Tez kapsamında çok sayıda yardımcı kütüphane ve işletim sisteminin sunmuş olduğu yazılım araçları kullanılmıştır. Bu araç ve kütüphaneler, bu projenin daha iyi çalışabilmesi için detaylı literatür araştırmasının sonucunda özenle seçilmiştir. Kendi alanındaki yetkinliklerine göre sınıflandırılan kütüphaneler tez kapsamında karşılaştırılmıştır. Testlerin sonucunda kullanılmaması gereken kütüphaneler ile ilgili bilgiler verilmiştir ve projeden çıkarılmıştır.

2.1. Linux İşletim Sistemi

Linux [23], en çok bilinen ve en çok kullanılan açık kaynak kodlu bir işletim sistemidir. Bilgisayar da çalışan programların en alt katmanında çalıştırılan Linux, üst katmanda çalıştırılan programlar ile donanım arasında köprü görevi görmektedir. Linux işletim sistemini diğer işletim sistemlerinden ayıran en büyük özelliği açık kaynak kodlu olmasıdır. Tez kapsamında, alt yapısında Linux 3.16.7 kullanan Opensuse 13.2 [24] sanal işletim sistemi kullanılmıştır. İşlemler 32 bitte yapılmıştır. Derleme işlemlerinde GCC 4.8 kullanılmıştır.

2.2. IT++ Kütüphanesi

C++ dilinin kütüphanesi olan IT++ [5], nümerik hesaplamalara uygun fonksiyonlar içeren, sinyal işleme ve haberleşme alanlarında kullanılabilecek klaslara ve fonksiyonlara sahip bir kütüphanedir. Haberleşme sistemlerinin simülasyonlarında ve haberleşme alanında yapılan araştırmalarda kullanılmaktadır. Kütüphanenin temelinde matris ve vektör klasları bulunmaktadır. Bu özelliği ile Matlab, GNU Octave ve Scipy [30] programlarının kütüphaneleriyle benzerlik göstermektedir. Kütüphane, açık kaynak kodlu ve ticari amaçla yazılmış kütüphanelerin fonksiyonelliklerini, hızlarını ve doğruluklarını daha iyi hale getirebilmek için oluşturulmuştur. IT++; Atlas [29], MKL (Math Kernel Library) [31] ve ACML [32] kütüphaneleriyle beraber daha efektif bir şekilde kullanılabilir.

2.2.1. IT++ Kütüphanesi Performans Değerlendirmesi

IT++ kütüphanesi yapısı gereği MATLAB ile benzerlik göstermektedir. Kütüphane araçlarının doğru kullanılmasıyla MATLAB benzeri bir ortam oluşturulabilir. IT++ sayesinde MATLAB tan C++ kaynak koduna geçiş yapılabilmektedir [11].

IT++ kütüphanesinin performansı üzerine birçok araştırma mevcuttur. Nitekim yapılan bir araştırmada Matlab, Octave, Newmat ve IT++ çeşitli matris işlemleri üzerinden karşılaştırılmıştır. Matris işlemleri ile hız testi uygulanmıştır. NxN boyutlarına sahip A, B, C, D ve Q matrisi N = 50 önbellek boyutları sınırları içinde kalacak şekilde ve N = 500 ön bellek sınırlılarını aşacak şekilde test edilmiştir [12].

Test için 2 MB önbelleğe sahip Intel Core2 Duo işlemcisi kullanılmıştır. Sistem 2 GHz ve 64 bitte çalıştırılmıştır. Linux çekirdeği v2.6.32, GCC v4.4.4 kullanılmıştır. Kütüphaneler; Armadillo 0.9.80, Matlab v7.1.0.183, Octave v.3.2.3, Newmat 11 beta, IT++ v4.0.6 olarak belirlenmiştir [12].

2.3. Armadillo Kütüphanesi

Armadillo, C++ dilinin geniş kapsamlı lineer cebir alt yapısına sahip bir kütüphanesidir. Kütüphane kullanım kolaylığı ve hızlı çalışma prensiplerini kendine hedef olarak belirlemiştir. Kullanım açısından yüksek seviyeli kod yapısı ve fonksiyonelliği ile MATLAB kütüphanelerine benzemektedir. C++ ortamında direkt olarak algoritma geliştirme ve araştırma kodlarının hızlıca proje ortamında aktarılabilmesini sağlamaktadır. Armadillo, nümerik hesaplamalarda matris ve vektör alt yapısına sahip klas ve fonksiyonlarıyla araştırma süreçlerine destek vermektedir. Alt yapısında tam sayılar, rasyonel sayılar ve karmaşık sayılar içeren algoritmaları desteklemektedir. Kütüphane otomatik olarak alt işlemlerinde OpenMP [9] kütüphanesini paralelleştirme için kullanmaktadır. Özelleştirilmiş ifade tanıyıcıları ve diğer otomatik çalışan araçları sayesinde daha hızlı ve efektif çalışan kodların yazılmasına destek vermektedir. Kütüphane makine öğrenmesi, örüntü tanıma, sinyal işleme ve finans alanlarında kullanılabilmektedir [19].

2.3.1. Sayısal Hesaplama Yöntemleri ve Armadillo

Sayısal hesaplama yöntemlerinin yazılım alanında gerçekleştirilmesi ve bu yazılımların daha hızlı çalışabilmesi için yapılan çalışmalar için çeşitli kütüphaneler tasarlanmıştır. Bunlardan birisi olan Armadillo hesaplama algoritmalarının gerçekleştirilmesinde ve daha hızlı çalıştırılabilmesi kullanılabilmektedir.

En temel hesaplama yöntemlerinden birisi olan Gauss Eleme Yöntemi için lineer cebir işlemlerinin daha hızlı gerçekleştirilebilmesi amacıyla Armadillo Kütüphanesi test içinde kullanılmıştır [13].

2.3.2. Matlab Kodlarının Armadillo ile C++ Kaynak Kodlarına Dönüştürülmesi

Matlab kodlarının C++ kaynak kodlarına dönüştürülmesi işlemi Armadillo Kütüphanesinin lineer cebir operasyonları için tasarlanmış olan araçlarının Matlab'ın araçlarıyla benzerlik göstermesi dolayısıyla efektif bir şekilde yapılabilmektedir. Armadillo; LAPACK, Intel MKL, ACML ve OpenBLAS gibi kütüphaneleri daha performanslı çalışabilmek için kullanmaktadır [14]. Kod dönüşümünü ve oluşturulan ortamın test edilmesi için çeşitli simülasyon testleri çalıştırılmıştır.

Sonuçların incelenmesi halinde görüleceği üzere yüksek seviyeli dil olan Matlab içerisinde daha sade ve okunabilirliği kolay kod yazılabilmektedir. Performansı arttırmak için kaynak kodların C++ içerisinde yazılması kod uzunluğu açısından dezavantajlı olarak görülse de düşük seviyeli bir dilde kod geliştirmek yazılımcıya yazmış olduğu koda her noktada müdahale edebilme imkânı sağlamaktadır [14].

2.4. Ctypes Kütüphanesi

Ctypes, Python için yazılmış üçüncü parti bir kütüphanedir. Kütüphane C diliyle uyumlu fonksiyonlara ve veri türlerine sahiptir. Ctypes, DLL ve "Shared Library" aracılığıyla fonksiyon çağrılarına izin vermektedir. Tez kapsamında yazılan C++ kodlarının derlenmesi sonucunda oluşturulan objeler, "Shared Object" dosya

türü ile python tarafına aktarılmaktadır. Kütüphanenin alt yapısında çokça C dili ile yazılmış fonksiyon ve yapı mevcuttur. Bu yapı ve fonksiyonlar "Pointer" araçlarını sıklıkla kullanmaktadırlar. "Pointer" araçları genel yapıları gereği anlaşılması zor ve karmaşık yapılar ortaya çıkarmaktadırlar. Oluşan bu karmaşık ortam, tezin amacıyla uyumluluk göstermediği için tez kapsamında kullanılmaması gerektiği düşünülüp aynı amaçla kullanımı daha rahat olan başka kütüphaneler araştırılmıştır.

2.5. Boost Python Kütüphanesi

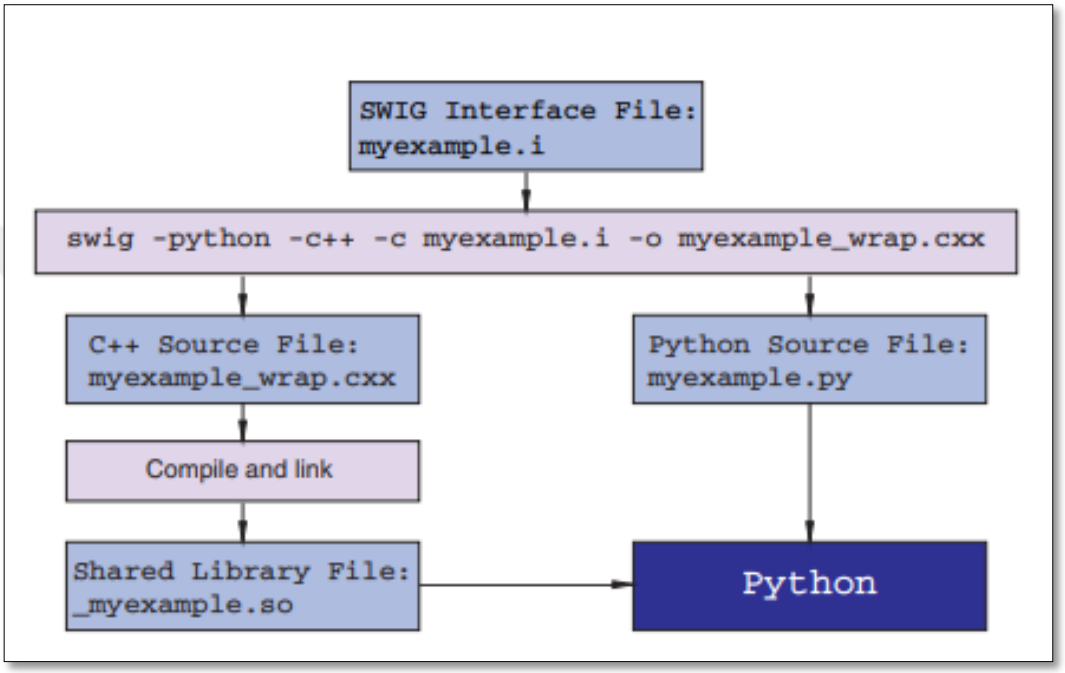
Boost, açık kaynak kodlu bir C++ kütüphanesidir. Kütüphane, kullanılabilirlik açısından geniş uygulama alanlarına destek sağlamak amacıyla oluşturulmuştur. Boost kütüphanesi, temel standartlara uygun hazırlanmıştır ve uygulamalar için referans olacak yapılar oluşturmayı hedeflemektedir. Tez kapsamında kütüphanenin temel bölümlerinden olan Boost Python kapsamlı şekilde kullanılmıştır.

Boost Python, mevcut C/C++ modülleriyle ara yüz oluşturmamıza ve hızın kritik olduğu yerlerde C++'da yeni modüller yazmamıza olanak tanır [15]. Kütüphane, C++ ve Python dillerine sorunsuz şekilde beraber çalışabilme imkânı sağlamaktadır. Çeşitli konularda ara yüz destekleri vermektedir. Karşılıklı uyumlaştırılmış işaretçi ve referanslar kütüphanenin sağladığı başlıca özelliklerdendir. Kütüphane, otomatik tip dönüşümlerine imkân sağlamaktadır ayrıca fonksiyon aşırı yüklemesi efektif şekilde kullanılabilmektedir. Python objelerinin C++ içerisinde işlenmesi kütüphanenin en önemli özelliklerindendir. C++ yineleyicileri ve Python yineleyicileri karşılıklı olarak aktarılabilmektedir.

Tez kapsamında, Boost Python kütüphanesinin sağladığı Python dilinin Numpy kütüphanesine ait "ndarray" dizinlerinin C++ tarafından üretilip nümerik hesaplamaların C++ tarafında tamamlanması sağlanmıştır. Kütüphanenin temel özelliklerden olan işaretleyiciler ve referansların karşılıklı olarak aktarılabilmesi özelliği ile ağır nümerik hesaplamalar Boost Python sayesinde C++ tarafına taşınmıştır. Düşük seviyeli dil olan C++ dilinde yapılan işlemler daha hızlı şekilde tamamlanmıştır. Boost Python, C++ içerisinde oluşturulan fonksiyonların Python tarafından çağrılabilmesini sağlamaktadır [34].

2.5.1. Python C++ Etkileşiminin SWIG Aracıyla Oluşturulması

SWIG, yüksek seviyeli diller ile C/C++ dilleri arasında bağlantı oluşturabilen bir yazılım aracıdır. Python ve C++ arasında karşılıklı veri aktarımı için SWIG tercih edilebilmektedir. SWIG daha çok yüksek seviyeli bir yazılım ortamı oluşturmak için kullanılmaktadır. Araç genel olarak C/C++ kodlarını ayrıştırarak bir kod dizini oluşturur [8].



Şekil 2.1: SWIG ile oluşturulmuş C++, Python modülü [6].

Proje kapsamında paylaşımlı obje ile oluşturulmak istenen yapının SWIG ara yüzünde kullanıldığı ve C++ Python ilişkisinin sağlandığı gözlemlenmiştir. Paylaşımlı obje yapısının tez kapsamında makefile üzerinden kullanımına karar verilmiştir.

2.6. OpenMP Kütüphanesi

OpenMP, uygulama programlama ara yüzüne sahip aynı zamanda paralel programlamaya imkân sağlayan, C, C++ ve Fortran gibi platformlara destek veren bir kütüphanedir [9]. Solaris, AIX, Windows, Linux, HP-Ux, macOS işletim sistemlerinde çalışabilmektedir. Kütüphane çalışma anında işleyişe etki edecek

derleyici araçlarına, kütüphane rutinlerine ve ortam araçlarına sahiptir. Kütüphane, portatif ve uyum sağlayabilme özellikleri sayesinde yazılımcılara basit ve kullanışlı ara yüz sunmaktadır. Sunulan ara yüz, paralel programlama alanında yapılan çalışmaların kolay ve hızlı ilerleyebilmesini sağlamaktadır. Tez kapsamında kütüphane, büyük matris ve vektörlerin kopyalama işlemlerini hızlıca tamamlamak için kullanılmıştır.

OpenMP, diğer paralel programlama kütüphaneleriyle performansının karşılaştırılması için çalışmalar incelenmiştir. Π sayısının 10^9 aralıklarla hesaplanmasının OpenMP, MPI ve Pthread kütüphanesi ile hesaplama sonuçları karşılaştırılmıştır [16].

2.7. ACML (AMD Core Math Library) Kütüphanesi

AMD Core Math Library (ACML) yazılım geliştiricileri için AMD şirketi tarafından geliştirilmiş bir kütüphanedir. Kütüphane, optimize olmuş şekilde çalışarak matematiksel rutinlere yenilikçi çözümler sunmaktadır. Kütüphane daha çok AMD işlemciler için geliştirilmiştir. Kütüphanenin temelinde, seviye bir, seviye iki ve seviye üç temel lineer cebir alt yazılımları bulunmaktadır. Lineer cebir altyapısı BLAS olarak adlandırılmaktadır. Kütüphane, kapsamlı hızlı "Fourier" dönüşüm normlarına sahiptir. Sayısal ve vektörel işlemlerde üst düzeyde yazılmış fonksiyonları ile nümerik işlemleri hızlı şekilde yapabilmektedir. Hassas yapıda tasarlanmış rastgele sayı üreticileri içermektedir. Linux, Windows, Solaris gibi platformlara destek vermektedir. Tez kapsamında kütüphane, GNU Linux derleyicisiyle kullanılmıştır. ACML kütüphanesi temelde IT++ ile kullanımının getirdiği faydalar üzerine tez kapsamında kullanılması gerekliliği düşünülerek projeye eklenmiştir.

2.8. Makefile

Makefile, programın ne şekilde derleneceğini ve link edileceğini belirleyen bir otomasyon aracıdır. Unix sistemlerde kullanılmakta olan Makefile, içerisine yazılan kurallar sayesinde yapılmak istenen işlemler sırayla tamamlayabilmektedir.

Derlenmek istenen kodun bulunduğu klasörü, derleyicinin versiyonunu ve bağlanacak kütüphanelerin bulundukları yerlere kadar ayrıntılı şekilde tasarıma imkân sağlamaktadır. Makefile dosyası yazıldıktan sonra işletim sisteminin terminalinden make komutuyla çalıştırılabilir. Tez kapsamında Makefile aracı, çoklu kütüphane kullanımını kolaylaştırmak amacıyla kullanılmıştır. Makefile içerisinde kullanılan "patchelf [33]" komutu sayesinde çalıştırılan kütüphaneler için ortak bir çalışma alanı oluşturulmuştur. Oluşturulan bu alan testin çalıştırılması anında bütün test elemanlarının ve kütüphanelerinin birbirine bağlanmasını sağlamaktadır. Test amaçlı yazılan simülasyon kodlarının yardımcı kütüphanelerinin de ayrıca derlenip kütüphanelerin kendi içerisinde paylaşımlı nesne oluşturabilmesi Makefile araçları ile sağlanmaktadır.

2.9. Python

Python [4], yüksek seviyeli, modüler ve nesne yönelimli bir programlama dilidir. Modüler yapısı, kod yazımını kolaylaştırması ve karmaşıklığı önlemesi açısından C ve C++ dillerinin yerine tercih edilebilmesini sağlamaktadır. Unix, Linux, Mac, Windows gibi platformlarda kullanılabilmektedir. Numerik hesaplamalarda ve elektromanyetik dalga simülasyonlarında da çokça kullanılan Python, tez kapsamında C++ dilinde nümerik hesaplamaları yapılmış olan simülasyonların Pythonda grafiklere dökülmesi ve farklı kütüphanelerin performanslarının ölçülmesi amacıyla kullanılmıştır. Python için yazılan Boost Python kütüphanesinin ara yüzü sayesinde Python dilinin kütüphanesi olan Numpy kütüphanesinin araçları hem C++ hem de Python tarafında kullanılmıştır. Hesaplamalar sonucunda oluşan nümerik değerler "numpyarray" araçlarıyla C++ dan Python tarafına geçirilmiştir.

3. HESAPLAMALI ELEKTROMANYETİZMA YÖNTEMLERİ ve UYGULAMALARI

3.1. İki Boyutta Zaman Uzayında Sonlu Farklar Yöntemi

Zaman Uzayında Sonlu Farklar Yöntemi, ilk Kane Shee-Gong Yee tarafından 1966 yılında bulunmuştur. Yöntem, zaman uzayında Maxwell denklemlerini belirli bir düzlem üzerinde çözmek üzere tasarlanmıştır. Hesaplama verimliliği, doğru sonuçların üretilmesi ve fiziksel yorumlamalarının oluşturduğu faydalar sayesinde ZUSF yönteminin elektromanyetik dalgaların yayılma ve saçılma problemlerinde, elektromanyetik dalgaların biyolojik etkenlerinin araştırılmasında ve mikrodalga devrelerinin analizlerinde kullanımı giderek artmaktadır [18].

$$\frac{\partial D}{\partial t} = \frac{1}{\sqrt{\varepsilon\mu}} \nabla \times H \tag{3.1}$$

$$D(\omega) = \varepsilon(\omega) . E(\omega) \tag{3.2}$$

$$\frac{\partial H}{\partial t} = -\frac{1}{\sqrt{\varepsilon\mu}} \nabla \times E \tag{3.3}$$

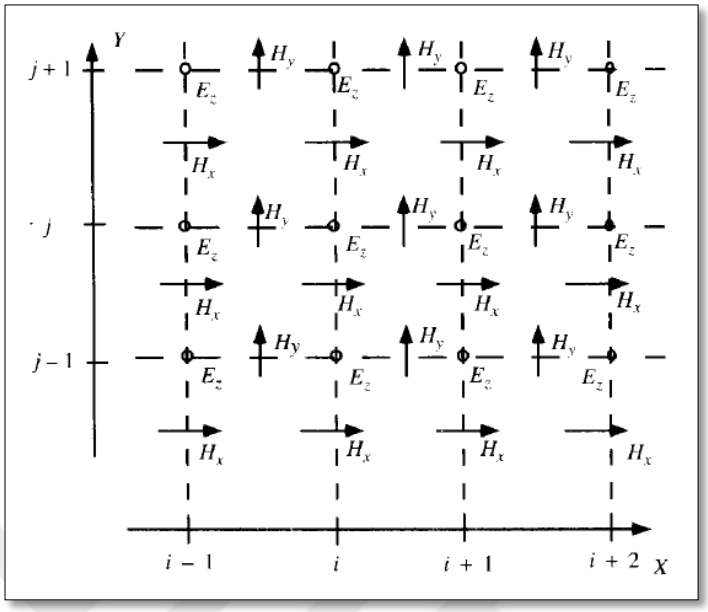
İki boyutlu hesaplamalarda Ex, Ey, Ez, Hx, Hy, Hz vektörlerinden üç adet vektör seçilmelidir. Enine manyetik mod için Ez, Hx ve Hy seçilir enine elektrik mod için ise Ex, Ey ve Hz seçilecektir [17].

$$\frac{\partial D}{\partial t} = \frac{1}{\sqrt{\varepsilon\mu}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right) \tag{3.4}$$

$$D_z(\omega) = \varepsilon_r(\omega) . E_z(\omega) \tag{3.5}$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\sqrt{\varepsilon\mu}} \frac{\partial E_z}{\partial y} \tag{3.6}$$

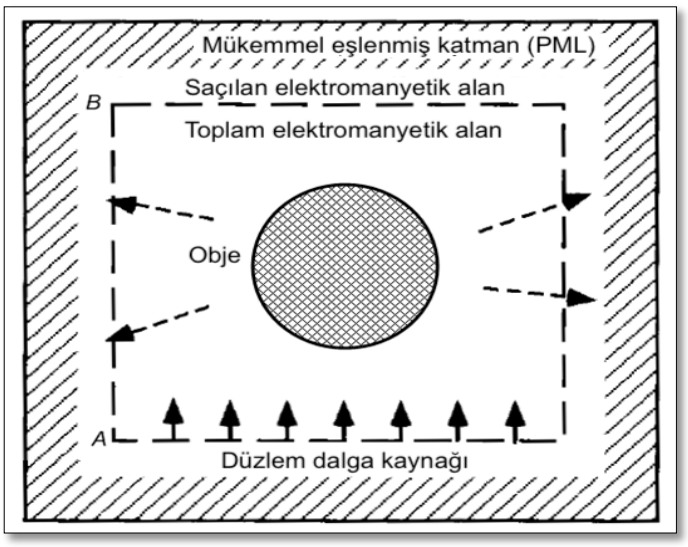
$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\varepsilon\mu}} \frac{\partial E_z}{\partial x} \tag{3.7}$$



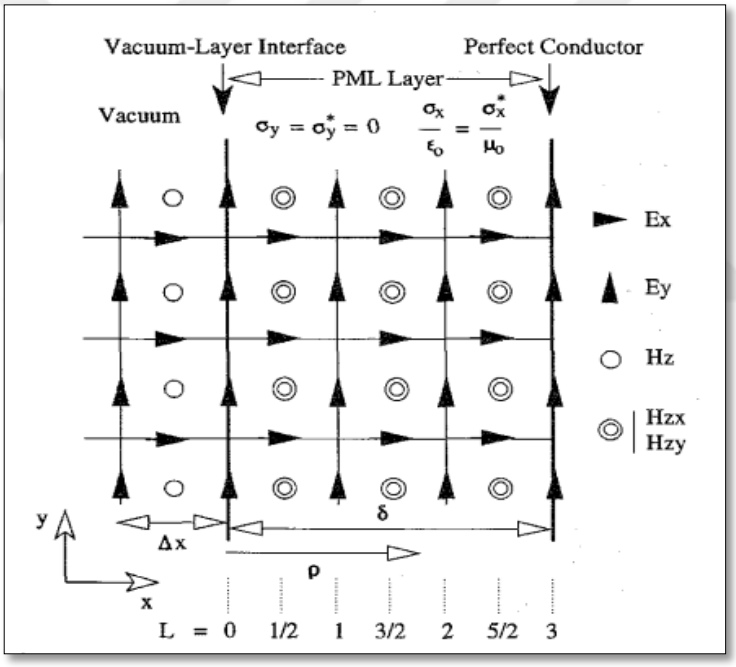
Şekil 3.1: İki boyutlu alanda enine manyetik modu E ve H binişimi.

3.2. Mükemmel Eşlenmiş Katman (MEK)

Mükemmel eşlenmiş katman, ZUFS yöntemi ile kullanılarak sınırlandırılmamış elektromanyetik dalga problemlerinin çözümü için kullanılan bir çözümleme tekniğidir. MEK tekniği elektromanyetik dalgaların yansıma olmadan absorbe edilebilmesi için oluşturulan katmanın kullanılması üzerine tasarlanmıştır [20].



Şekil 3.2: Mükemmel eşlenmiş katman (PML).



Şekil 3.3: Mükemmel eşlenmiş katman uygulaması.

3.3. Mur Emici Sınır Koşulları

Zaman uzayında sınırlandırılmamış alan içinde çözümlenen elektromanyetik alan denklemlerinin çözümlerinin tamamlanabilmesi için çözümleme alanını sınırlayan bir yönteme ihtiyaç vardır. Bu amaçla emici sınır koşulları (ESK)

sınırlandırılmamış problemlerin benzetimlerinin oluşturulabilmesi için kullanılmıştır [21].

ZUFS yöntemi içerisinde birinci dereceden Mur ESK kullanılırken y ve z eksenlerindeki türev terimleri yok edilir [22].

$$W|_{0,j,k}^{n+1} = W|_{1,j,k}^{n} + \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\left(W|_{1,j,k}^{n+1} - W|_{0,j,k}^{n}\right) \tag{3.8}$$

İkinci dereceden Mur ESK kullanıldığında y ve z eksenlerindeki türev terimleri korunarak alan dönüştürme metodu kullanılarak çözümlenmesi sağlanır [22].

$$\frac{\partial^2 W}{\partial x \partial t} - \frac{1}{c}\left(1 - \frac{k_y^2}{2} - \frac{k_z^2}{2}\right)\frac{\partial^2 W}{\partial t^2} + \frac{c}{2}\left(\frac{\partial^2 W}{\partial y^2} + \frac{\partial^2 W}{\partial z^2}\right) - \frac{1}{\partial t}\left(k_y \frac{\partial W}{\partial y} + k_z \frac{\partial W}{\partial z}\right) = 0 \tag{3.9}$$

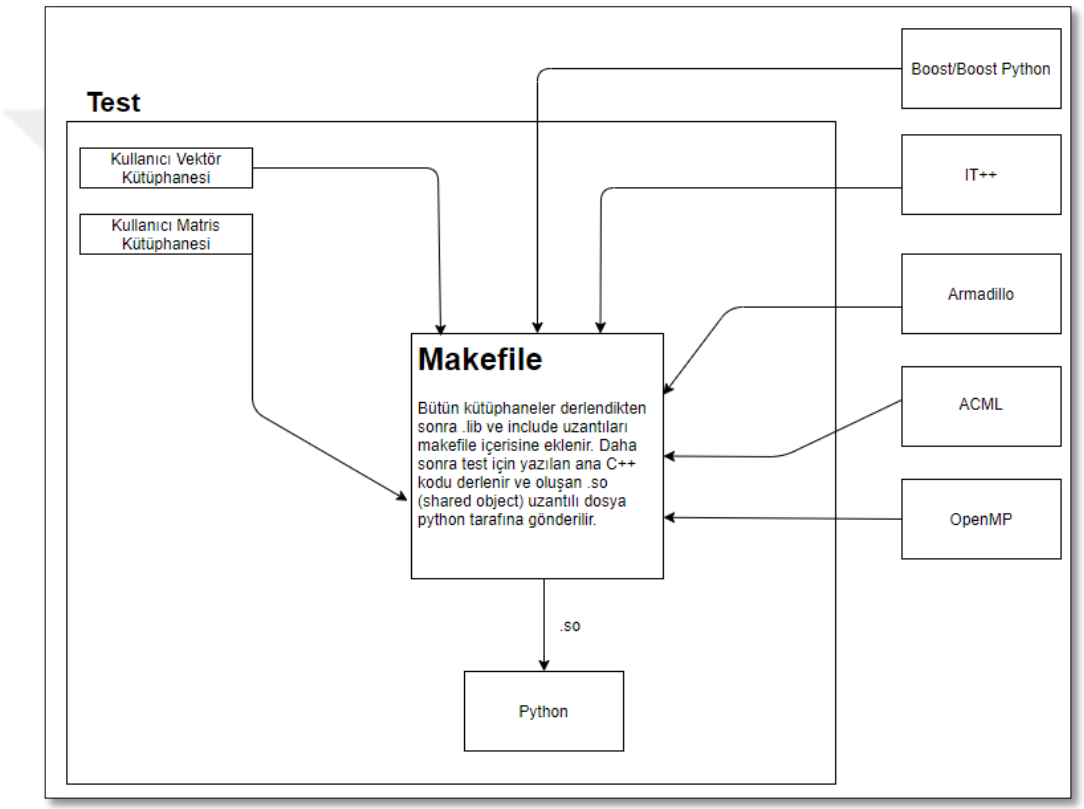
3.4. Uygulama

Proje kapsamında belirlenen tasarım için iki adet test kodu yazılmıştır. Bu testler; iki boyutlu, zamanda sonlu farklar yöntemi ile tasarlanmış; "Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi" ve iki boyutlu, zamanda sonlu farklar yöntemi ile tasarlanmış "Fotonik Kristal Doksan Derece Eviren Dalga Kılavuzu Testidir". Test kodları içerisinde MEK ve ESK ile dalga kılavuzlarının simülasyonlarının gerçekleştirilebileceği ortam hazırlanmıştır.

Tasarlanan sistemde çalıştırılacak testlerin belirlenmesinden sonra kaynak kod ve yan kütüphane kodlarında kullanılmak üzere yardımcı kütüphaneler kurulup derlenmiştir. Bu kütüphaneler; IT++, Armadillo, Boost, ACML ve OpenMP gibi kütüphanelerdir. İşletim sisteminde yerel bir alana kurulan kütüphanelerin derlenmesiyle oluşan ".lib" dosyalarının dosya uzantıları projenin ilerleyen aşamalarında yazılacak Makefile içerisinde kullanılmak üzere not edilmiştir.

Kütüphaneler ortak bir alanda tutulsa da test dosyaları ayrı klasörlerde tutulmuştur. Her test için, aile dosya yapısı kullanılarak projenin okunabilirliği arttırılmış ve sabit düzene sahip bir proje alt yapısı oluşturulmuştur. Oluşturulan aile

dosya yapısı içerisinde; kaynak C++ kodu, kullanıcının vektör ve matris işlemlerinde ayrı şekilde kullanabileceği kütüphane ortamı, ana kaynak kodun derlenmesi için Makefile dosyası ve son aşamada oluşturulan nümerik değerlerin grafiksel ifadelerinin oluşturulabilmesi için Python dosyaları mevcuttur. Test oluşturacak yazılımcının kendi fonksiyon altyapısını oluşturabilmesi için sunulan dosyalar içerisinde kendi kütüphanelerini derleyip ".lib" dosyalarını oluşturabilmektedir. Oluşan ".lib" dosyaları ana kaynak kodunu derleyecek olan Makefile ile ana test projesine eklenebilmektedir.



Şekil 3.4: Aile dosya yapısı.

Proje kapsamında kullanılan yardımcı üçüncü parti kütüphaneler sayesinde hem yazılımcının kendi oluşturacağı kütüphanede hem de ana kaynak kodu içerisinde nümerik hesaplamalar kolayca yapılabilmektedir. IT++ ve Armadillo kütüphaneleri yazılımcıya gerek oluşturtulduğu matris ve vektörler işlemlerinde gerekse içerisinde bulundurduğu efektif kopyalama fonksiyonlarıyla projeye portatif ve kullanılabilirliği kolay bir yapıda olma özelliği kazandırmıştır. Yazılımcı, C++ dilinde kod yazarken kullanacak olduğu bu kütüphaneler sayesinde yüksek seviyeli

bir dilde kod yazıyormuş gibi hissedebilmektedir. Kaynak kod içerisinde ve yazılımcının kendi yazacağı kütüphane içerisinde kullanılmak üzere OpenMP Kütüphanesi donanım paralelleştirmesi için eklenmiştir. Bu sayede işletim sisteminin çalıştırıldığı sistemdeki maksimum çekirdek sayısı kullanılarak çalıştırılmak istenen test hızlıca tamamlanabilmektedir.

Kaynak kodun destekleyicisi olan bir diğer kütüphanede Boost Kütüphanesi'dir. Boost Kütüphanesi içerisinde önemli bir yere sahip olan Boost Python efektif bir şekilde projede kullanılmıştır. Boost Python yapısı, hazırlanan ana kod dosyasının temelini teşkil etmektedir. Python tarafına taşınacak fonksiyonun isminin ve içeriğinin belirlemesini sağlamaktadır. Python tarafına taşınacak olan veriler hangi formatta taşınacağını da yine Boost Python sayesinde belirlenebilmektedir. Proje kapsamında oluşturulan nümerik veriler Python tarafına "numpyarray" olarak taşınmıştır. Projenin bu kısmında karşılaşılan çalışma zamanı obje dosyalarının birleştirilmesi problemi Makefile aracının "patchelf" komutu sayesinde giderilmiştir. Boost Python, ana fonksiyona girdilerin Python tarafından verilerek yürütme işleminin C++ dilinde tamamlanmasına imkân sağlamaktadır. Python tarafına taşınan nümerik değerler, Python dilinin efektif şekilde kullanılabilen grafiksel araçlarının testlerin değerlendirilmesini kolaylaştırmak amacıyla kullanılmıştır.

```
#include <boost/python.hpp>

BOOST_PYTHON_MODULE(ninety_degree_bend)
{
        using namespace boost::python;
        Py_Initialize();
        np::initialize();
        def("compute", compute);
}
```

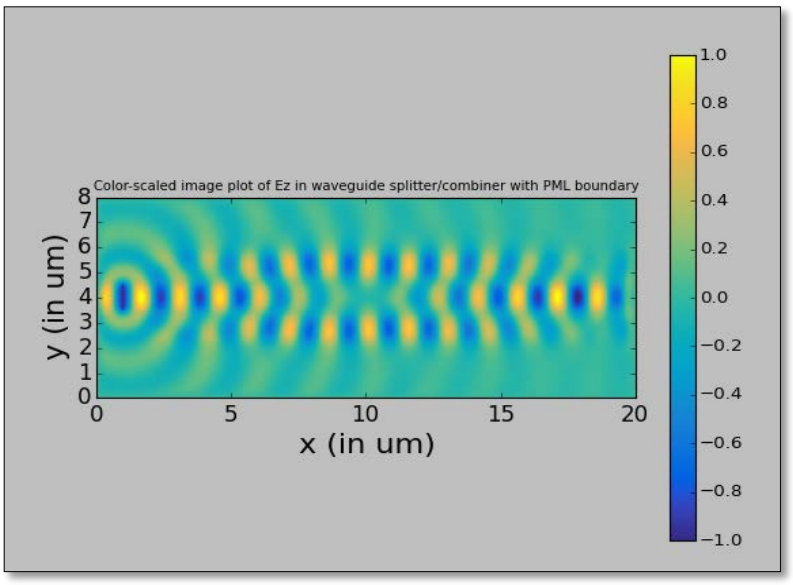
Şekil 3.5: Boost python yapısı.

4. SONUÇ

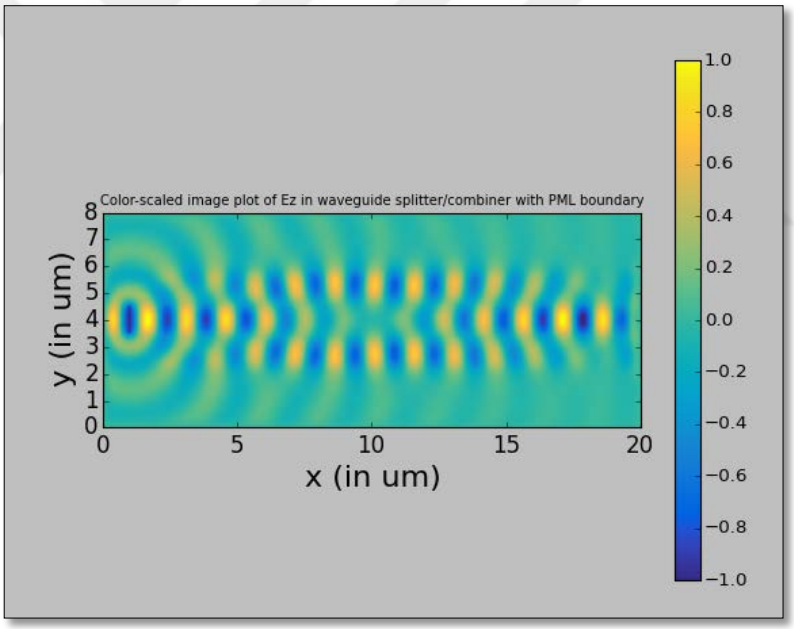
Matlabda hazırlanmış olan [3] kaynak test kodlarının C++ ortamına geçirilmesi işlemleri adım adım gerçekleştirilmiştir. İlk aşamada Matlab, C++ ve Pythonda matris ve vektörler için eleman numaralandırmaları farklı sayı ile başladıkları için yazılan bütün fonksiyonların test edilebilmesi amacıyla "Hata Ayıklama Modu" C++ ortamında oluşturulmuştur. Hata Ayıklama Modu için Linux terminali tarafından çalıştırılabilir dosya formatını oluşturacak hazır makefile kullanılmıştır. Hata Ayıklama Modu ile test kodları içerisindeki bütün kritik nümerik hesaplamaların sonuçları Matlab, Python ve C++ için karşılaştırılmış ve sonuçlar uyuşana kadar hatalar düzenlenmiştir. Sayısal olarak bütün değerlerin eşit olduğu gözlemlenmiştir.

Numaralandırma işlemlerinin başarıyla tamamlanmasından sonra Matlab bağımlılığı ortadan kaldırılmıştır. Projenin ana gövdesi olan Python ile C++ arasındaki bağlantıyı oluşturacak makefile oluşturulmuştur. Armadillo ve IT++ alt yapısıyla oluşturulan iki versiyon için de hesaplanan değerleri Python tarafına aktaracak kısım test edilmiştir. Bu aşamada taşıma ve satır kopyalama yöntemleri denenmiştir. Taşıma yöntemi riskli görülerek satır kopyalama yöntemi tercih edilmiştir. Son aşamada oluşturulan matrisin içerdiği değerler bir kez daha orijinal değerler ile karşılaştırılarak Python tarafına geçilmiştir. C++ tarafından gelen veriler Python tarafında oluşturulan "numpyarray" içerisine kopyalanmıştır. Sayısal değerlerin eşit olduğu gözlenmiş ve oluşturulan sistemin sağlıklı çalıştığı kanıtlanmıştır.

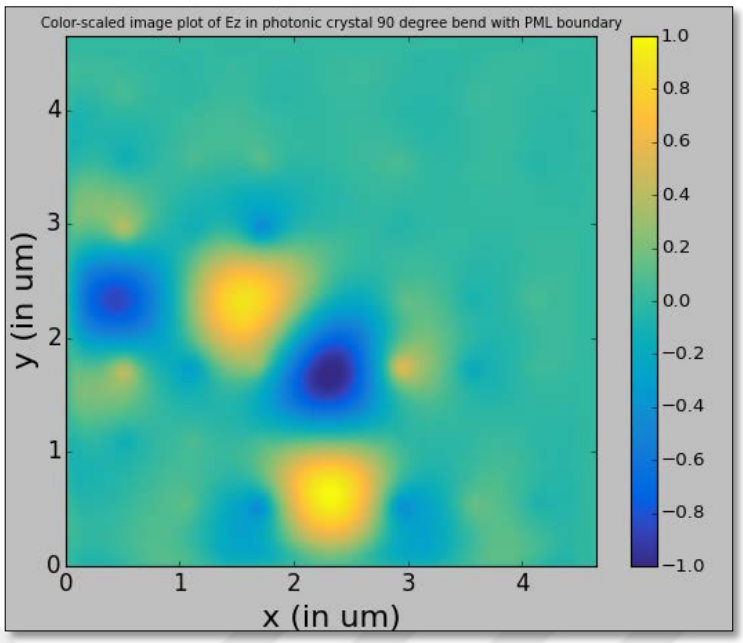
Başlangıçta Matlab ortamında oluşturulan hesaplanmış değerlerin grafiklerinin oluşturulduğu mekanizmanın bir benzeri de Python tarafında oluşturulmuştur. Renk haritasının sayısal değerleri ayrı bir Python dosyası ile eklenmiştir. Matlab ve Python grafiksel sonuçları karşılaştırılmış ve grafiksel ara yüzün de hatasız çalıştığı kanıtlanmıştır.



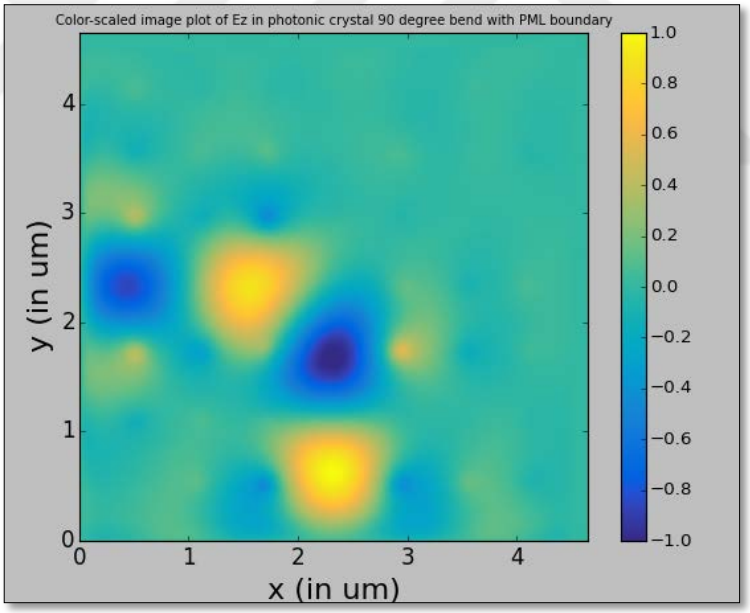
Şekil 4.1: 2B ZUSF ayrıştırıcı ve birleştirici dalga kılavuzu testi armadillo benzetim sonuçları.



Şekil 4.2: 2B ZUSF ayrıştırıcı ve birleştirici dalga kılavuzu testi IT++ benzetim sonuçları.



Şekil 4.3: 2B ZUSF fotonik kristal doksan derece eviren dalga kılavuzu testi Armadillo benzetim sonuçları.



Şekil 4.4: 2B ZUSF fotonik kristal doksan derece eviren dalga kılavuzu testi IT++ benzetim sonuçları.

Her iki dalga kılavuzu testi için sonuçlar incelenmiştir ve simülasyon sonuçlarının bire bir aynı olduğu gözlemlenmiştir. Armadillo ve IT++ kütüphanelerinin test ortamına aktarımı başarıyla gerçekleştirildiği anlaşılmıştır.

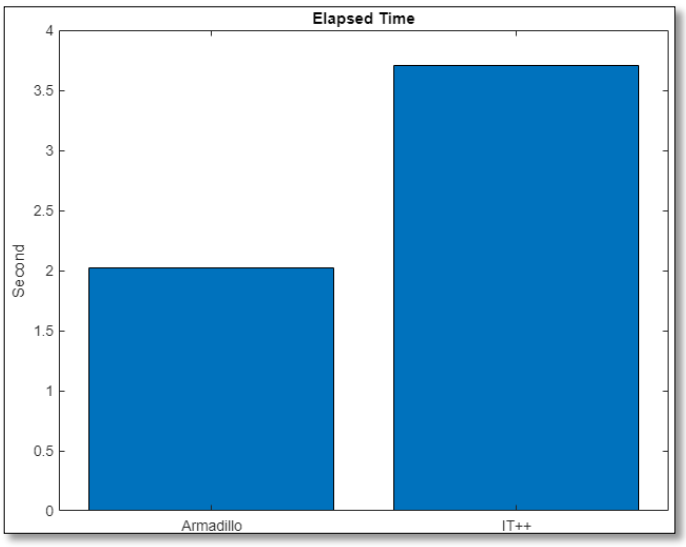
4.1. Armadillo ve IT++ Kütüphanelerinin Karşılaştırılması

Armadillo kütüphanesinin kendi kaynaklarında paylaştığı üzere yapılan karşılaştırma test sonuçları bu bölümde paylaşılmıştır. Armadillo, IT++ ve Newmat kütüphanelerinin karşılaştırıldığı bu çalışmada Armadillo ve IT++ arasındaki bölüme odaklanılmıştır.

İki adet 100x100 boyutlarında matrisin toplanmasını konu alan, dört adet 100x100 boyutlarında matrisin toplanmasını konu alan, dört adet 100x100 boyutlarında matrisin çarpımını konu alan, bir 100x100 matrisin alt matris operasyonlarında yapılan matrisin ilk satırını başka bir matrisin son satırına kopyalama işlemini konu alan ve son olarak bir devrik kolon vektörü ile bir köşegen matrisinin tersiyle çarpımı ve sonucunda bir kolon vektörünün çarpım işlemini konu alan testler kütüphaneyi oluşturan ekip tarafından yapılmıştır [19].

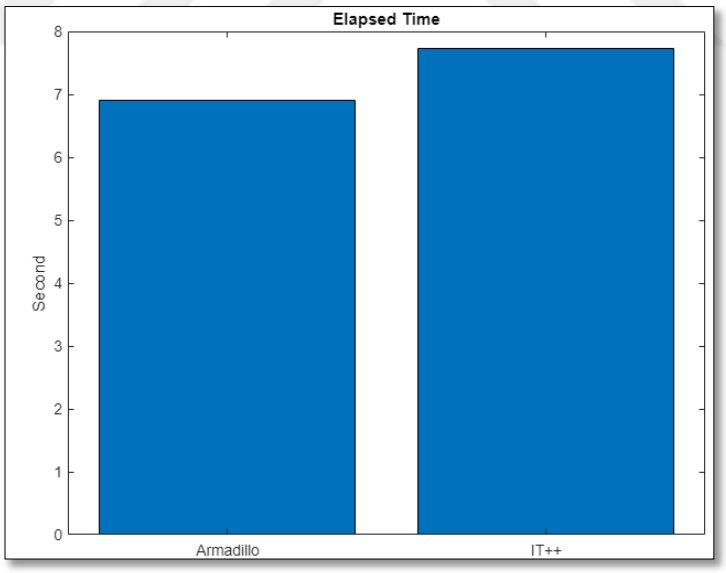
4.2. Simülasyon Sonuçlarına Göre Armadillo ve IT++ Kütüphanelerinin Karşılaştırılması

Tez kapsamında tamamlanan simülasyon test sonuçlarının testlerin gerçekleştirilme süreleri göre karşılaştırılması Şekil 4.5 ve Şekil 4.6 de görülmektedir. Şekil 4.5 2B ZUSF Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi için Pyton üzerinde çalıştırılmasından sonra ölçülen gerçekleşme süreleri görülmektedir.



Şekil 4.5: 2B ZUSF ayrıştırıcı ve birleştirici dalga kılavuzu testi Armadillo ve IT++ gerçekleşme süreleri.

Şekil 4.6 2B ZUFS Fotonik Kristal Doksan Derece Eviren Dalga Kılavuzu Testi için Python üzerinde çalıştırılmasından sonra ölçülen gerçekleşme süresi görülmektedir.



Şekil 4.6: 2B ZUFS fotonik kristal doksan derece eviren dalga kılavuzu testi Armadillo ve IT++ gerçekleşme süreleri.

KAYNAKLAR

[1] Reed G. T., Knights A., (2004), "Silicon Photonics - An Introduction 1st edition", 1st edition, John Wiley & Sons,Ltd .

[2] Taflove A. and Hagness S. C., (1995), "Computational Electrodynamics: The Finite-Difference Time-Domain Method-2D FDTD", 3rd edition, Artech House, Inc

[3] Web 1, (2021), https://www.mathworks.com/matlabcentral/profile/authors/3079258 , (Erişim Tarihi: 04/07/2021)

[4] Beazley D., (2013), "Python Cookbook Chapter 16. Extending and Embedding", 3rd edition, O'Reilly Media, Inc.

[5] Web 2, (2021),http://itpp.sourceforge.net/4.3.1/, (Erişim Tarihi: 04/07/2021)

[6] Cottom T. L., (2003), "Using Swig To Bind C++ To Python", Computing in Science and Engineering, 5(2), 88-97.

[7] Koranne S., (2010), "Handbook of Open Source Tools Chapter 6 Boost C++ Libraries", 2011th edition, Springer.

[8] Beazley D. M., (1996), "SWIG: An Easy to Use Tool for Integrating Scripting Languages with C and C++", 4th Annual Tcl/Tk Workshop, University of Utah, Salt Lake City, Utah, July 6-10.

[9] Mattson T., He H., Koniges A., (2019), "OpenMP Common Core: Making OpenMP Simple Again", 1st edition,The MIT Press.

[10] Chandra R., (2000), "Parallel Programming in OpenMP", 1st edition , Morgan Kaufmann.

[11] Tran D. N., (2017), "An Implementation Of Diabetic Retinopathy Grading System On Android Device Using Fully Automated Algorithm", 2017 International Conference on Advanced Technologies for Communications (ATC), Quy Nhon, Vietnam, 18-20 October.

[12] Sanderson C., (2010), "Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments", The University of Queensland, Sydney, Australia.

[13] Kużelewski A., Zieniuk E., (2016), "OpenMP For 3D Potential Boundary Value Problems Solved By PIES", AIP Conference Proceedings, 23 June, Białystok, Poland.

[14] Paulsen G. Y., (2016), "Matlab2cpp: a Matlab-to-C++ Code Translator", 2016 11th System of Systems Engineering Conference, Kongsberg, Norway, 12-16 June.

[15] Prasad D., (2010), “Python for Prototyping Computer Vision Applications”, Raphael Grasset New Zealand Computer Science Research Student, New Zeland, January.

[16] Dagum L., (1998), “Openmp: An Industry Standard API For Shared-Memory Programming”, IEEE Computational Science and Engineering, 5(1) , 46-55.

[17] Sullivan D. M., (2013), “Electromagnetic Simulation Using the FDTD Method ”, 2000th edition, Wiley-IEEE Press.

[18] Sui W., (1992), “Extending the Two-Dimensional FDTD Method To Hybrid Electromagnetic Systems With Active And Passive Lumped Elements”, IEEE Transactions on Microwave Theory and Techniques, 40(4) , 724-730.

[19] Web 3, (2021), http://arma.sourceforge.net/ , (Erişim Tarihi 04/07/2021).

[20] Berenger J. P, (1996), “Three-Dimensional Perfectly Matched Layer for the Absorption of Electromagnetic Waves”, Journal of Computational Physics, 127(2) , 363-379.

[21] Mur G., (1981), “Absorbing Boundary Conditions for the Finite-Difference Approximation of the Time-Domain Electromagnetic-Field Equations”, IEEE Transactions on Electromagnetic Compatibility, 23(4) , 377-382.

[22] Zheng A., Kishk A., Glisson A.W., (2006) “Implementatıon Of Mur’s Absorbing Boundaries With Periodic Structures to Speed Up The Design Process Using Finite-Difference Time-Domain Method”, Progress In Electromagnetics Research, 58, 101-114.

[23] Rusling D. A., (1999), “The Linux Kernel”, UNIX is a trademark of Univel.

[24] Castro J. D., (2016), “Introducing Linux Distros”, 1st. Edition, Apress.

[25] Web 4, (2021), https://www.boost.org/doc/libs/1_75_0/libs/python/doc/htlm /index.html, (Erişim Tarihi: 13/07/2021).

[26] Al-Kofani J. M., (2012), “Detecting semantic changes in Makefile build code”, 2012 28th IEEE International Conference on Software Maintenance, Trento, Italy, 23-28 September.

[27] Eaton J. W., (1997), “GNU Octave 4.2 Reference Manual”, Samurai Media Limited.

[28] Attaway S., (2013), “Matlab: A Practical Introduction to Programming and Problem Solving”, 3rd edition, Butterworth-Heinemann.

[29] Deconinck W., (2017), “Atlas: A library for numerical weather prediction and, climate modelling”, Computer Physics Communications,220, 188-204.

[30] Silva F. J., (2013), “Learning SciPy for Numerical and Scientific Computing”, 2nd Revised ed. Edition, Packt Publishing.

[31] Belonosov M., (2012), “Parallel Numerical Simulation of Seismic Waves Propagation with Intel Math Kernel Library”, Proceedings of the 11th International conference on Applied Parallel and Scientific Computing, Novosibirsk, Russia, 153-167, June.

[32] Yu J. K., Hwang J., Tang C. Y. , (2004), “Numerical Performance and Throughput Benchmark for Electronic Structure Calculations in PC−Linux Systems with New Architectures, Updated Compiler and Libraries”, Journal of Chemical Information and Computer Sciences, 44(2), 633-642.

[33] Web 5, (2021), https://nixos.org/patchelf.html, (Erişim Tarihi: 16/07/2021).

[34] Karssenberg D., (2007), “Modelling landscape dynamics with Python”, International Journal of Geographical Information Science, 21(5), 483-495.

ÖZGEÇMİŞ

Muhammed TORUN 2013 yılında başladığı Gebze Teknik Üniversitesi Mühendislik Fakültesi Elektronik Mühendisliği Bölümünü 2018 yılında başarıyla tamamlayarak aynı yıl yüksek lisans eğitimine Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik Mühendisliği Anabilim Dalında başladı. 2018 yılından bu yana TUSAŞ (Türk Havacılık ve Uzay Sanayii) şirketinde yazılım tasarım mühendisi olarak çalışmaktadır.

EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Torun M., Duman M. E., Şuvak Ö., "A Python/C++ Toolbox for the Uncertainty Quantification of 2D Waveguides", ICAR 5 th International Congress of Academic Research, Online, 19-21 April 2021.

Ek B: 2B ZUSF Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi IT++ C++ Kaynak Kodları

```
#include<iostream>
#include<itpp/itbase.h>
#include <omp.h>
#include <chrono>
#include <boost/python/numpy.hpp>
#include <boost/python/numpy/ndarray.hpp>
#include"help_functions_mat.hh"
#include"help_functions_vec.hh"
//#include<armadillo>
namespace p = boost::python;
namespace np = boost::python::numpy;

np::ndarray compute(int pml,int abc,int factor,double S,double epsilon0,double
mu0,double c,double delta,double deltat,int xdim,int ydim,int time_tot,double
wav,double index)
{

   using namespace std;
   using namespace itpp;

   // Initialization of permittivity and permeability matrices
   mat epsilon=epsilon0*ones(xdim,ydim);
   mat mu=mu0*ones(xdim,ydim);

   Transport buffer;
   Permittivity_profile pp;
   p::tuple shape = p::make_tuple(xdim,ydim);
   np::dtype dtype = np::dtype::get_builtin<double>();
   np::ndarray converted = np::zeros(shape,dtype);

   pp.permit_with_itpp_2d(epsilon,0,xdim-1,9*factor,13*factor-
1,index*index*epsilon0);
   pp.permit_with_itpp_2d(epsilon,0,xdim-1,14*factor,18*factor-
1,index*index*epsilon0);
```

```
   pp.permit_with_itpp_2d(epsilon,0,xdim-1,19*factor,23*factor-
1,index*index*epsilon0);


   if(pml==1)
   {
     mat Ez=zeros(xdim,ydim);
     mat Ezx=zeros(xdim,ydim);
     mat Ezy=zeros(xdim,ydim);
     mat Hy=zeros(xdim,ydim);
     mat Hx=zeros(xdim,ydim);


     // Initializing electric conductivity matrices in x and y directions
     mat sigmax=zeros(xdim,ydim);
     mat sigmay=zeros(xdim,ydim);


     //Boundary width of PML in all directions
     int bound_width=factor*4;

     //Order of polynomial on which sigma is modeled
     double gradingorder=6;

     //Required reflection co-efficient
     double refl_coeff=1e-6;

     //Polynomial model for sigma
     double sigmamax=(-
log10(refl_coeff)*(gradingorder+1)*epsilon0*c)/(2*bound_width*delta);


     Boundry bound1(xdim/2-1,bound_width-
1,epsilon0,sigmamax,bound_width,gradingorder);
     Boundry bound2(xdim/2-1,ydim-bound_width-
1,epsilon0,sigmamax,bound_width,gradingorder);
     Boundry bound3(bound_width-1,ydim/2-
1,epsilon0,sigmamax,bound_width,gradingorder);
     Boundry bound4(xdim-bound_width-1,ydim/2-
1,epsilon0,sigmamax,bound_width,gradingorder);

     vec val_1=bound1.baund_vector(epsilon);
     vec val_2=bound2.baund_vector(epsilon);
     vec val_3=bound3.baund_vector(epsilon);
     vec val_4=bound4.baund_vector(epsilon);
     sigmax=matrix_opt_inv_x(sigmax,0,xdim,0,bound_width+1,val_1);
     sigmax=matrix_opt_x(sigmax,0,xdim,ydim-bound_width-1,ydim,val_2);

     sigmay=matrix_opt_inv_y(sigmay,0,bound_width,0,ydim,val_3);
     sigmay=matrix_opt_y(sigmay,xdim-bound_width-1,xdim,0,ydim,val_4);
```

```
    //Magnetic conductivity matrix obtained by Perfectly Matched Layer condition
    //This is also split into x and y directions in Berenger's model
    mat sigma_starx=elem_div((elem_mult(sigmax,mu)),epsilon);
    mat sigma_stary=elem_div((elem_mult(sigmay,mu)),epsilon);

    //Multiplication factor matrices for H matrix update to avoid being calculated
many times
    //in the time update loop so as to increase computation speed
    mat G=(elem_div((mu-0.5*deltat*sigma_starx),(mu+0.5*deltat*sigma_starx)));
    mat H=(deltat/delta)/(mu+0.5*deltat*sigma_starx);
    mat A=elem_div((mu-0.5*deltat*sigma_stary),(mu+0.5*deltat*sigma_stary));
    mat B=(deltat/delta)/(mu+0.5*deltat*sigma_stary);

    //Multiplication factor matrices for E matrix update to avoid being calculated
many times
    //in the time update loop so as to increase computation speed
    mat C=elem_div((epsilon-0.5*deltat*sigmax),(epsilon+0.5*deltat*sigmax));
    mat D=(deltat/delta)/(epsilon+0.5*deltat*sigmax);
    mat E=elem_div((epsilon-0.5*deltat*sigmay),(epsilon+0.5*deltat*sigmay));
    mat F=(deltat/delta)/(epsilon+0.5*deltat*sigmay);

    for(int n=1;n<time_tot+1;++n)
     {
      #pragma omp parallel
      {
       #pragma omp for schedule(static) nowait
       for(int i=0;i<xdim-1;++i){
        for(int j=0;j<ydim-1;++j)
        {
         Hy(i,j)=A(i,j)*Hy(i,j)+B(i,j)*(Ezx(i+1,j)-Ezx(i,j)+Ezy(i+1,j)-Ezy(i,j));
         Hx(i,j)=G(i,j)*Hx(i,j)-H(i,j)*(Ezx(i,j+1)-Ezx(i,j)+Ezy(i,j+1)-Ezy(i,j));
        }
       }
       #pragma omp for schedule(static) nowait
        for(int i=1;i<xdim;++i){
         for(int j=1;j<ydim;++j)
         {
          Ezx(i,j)=C(i,j)*Ezx(i,j)+D(i,j)*(-Hx(i,j)+Hx(i,j-1));
          Ezy(i,j)=E(i,j)*Ezy(i,j)+F(i,j)*(Hy(i,j)-Hy(i-1,j));
         }
        }
      }
     int tstart=1;
     double N_lambda=wav*1e-6/delta;
     double* a;
     double val_11=0.5*sin(((2*pi*(c/(delta*N_lambda))*(n-tstart)*deltat)));
     double val_12=0.5*sin(((2*pi*(c/(delta*N_lambda))*(n-tstart)*deltat)));
     for(int i=14*factor;i<18*factor;++i)
     {
```

```
        Ezx(bound_width-1,i)=val_11;
       }

       for(int i=14*factor;i<18*factor;++i)
       {
        Ezy(bound_width-1,i)=val_12;
       }
       Ez=Ezx+Ezy;

       }//n timetot !!

    buffer.cpp_to_ndarray_with_strides_2d(converted,Ez._data(),xdim,ydim);
    return converted;
   }//pm ==1

if(abc==1)
  {
      //Initialization of field matrices
     mat Ezz=zeros(xdim,ydim);
     mat Hyy=zeros(xdim,ydim);
     mat Hxx=zeros(xdim,ydim);

     //Initializing electric and magnetic conductivity matrices
     mat sigma=4e-4*ones(xdim,ydim);
     mat sigma_star=4e-4*ones(xdim,ydim);

     //Multiplication factor matrices for H matrix update to avoid being calculated
many times
     //in the time update loop so as to increase computation speed
     mat AA=elem_div((mu-0.5*deltat*sigma_star),(mu+0.5*deltat*sigma_star));
     mat BB=(deltat/delta)/(mu+0.5*deltat*sigma_star);

     //Multiplication factor matrices for E matrix update to avoid being calculated
many times
     //in the time update loop so as to increase computation speed
     mat CC=elem_div((epsilon-0.5*deltat*sigma),(epsilon+0.5*deltat*sigma));
     mat DD=(deltat/delta)/(epsilon+0.5*deltat*sigma);

     //Mur's absorbing boundary condition parameters
     double p0=1;
     double p2=-0.5;

     //Co-efficients of present and previous (regarding time-step) boundary Ezz values
     //in boundary update equation for forward/up and backward/down boundaries in
     //the domain (as given in [1])
     double c0=(c/(2*S))*(1-(p0/S));
     double c1=-(c/(2*S))*(1+(p0/S));
     double c2=(c/(pow(S,2)))*(p0+(p2*S*S));
     double c3=-(p2*c)/2;
     double c0efffor=-(c0/c1);
```

```
double c2efffor=-(c2/c1);
double c3efffor=-(c3/c1);
c0=(c/(2*S))*(1+(p0/S));
c1=-(c/(2*S))*(1-(p0/S));
c2=-(c/pow(S,2))*(p0+(p2*S*S));
c3=(p2*c)/2;
double c1effrev=-(c1/c0);
double c2effrev=-(c2/c0);
double c3effrev=-(c3/c0);

 //Storage vectors for Ez boundary and boundary-1 values of previous and its
//previous timesteps
mat prev_xfor=zeros(1,ydim);
mat prev_x_minus_1for=zeros(1,ydim);
mat prev_yfor=zeros(xdim,1);
mat prev_y_minus_1for=zeros(xdim,1);
mat prev_xrev=zeros(1,ydim);
mat prev_x_minus_1rev=zeros(1,ydim);
mat prev_yrev=zeros(xdim,1);
mat prev_y_minus_1rev=zeros(xdim,1);

mat prev_prev_xfor = zeros(1,ydim);
mat prev_prev_x_minus_1for = zeros(1,ydim);
mat prev_prev_xrev = zeros(1,ydim);
mat prev_prev_x_minus_1rev = zeros(1,ydim);
mat prev_prev_yfor = zeros(xdim,1);
mat prev_prev_y_minus_1for = zeros(xdim,1);
mat prev_prev_yrev = zeros(xdim,1);
mat prev_prev_y_minus_1rev = zeros(xdim,1);


for(int n=1;n<time_tot+1;++n)
{
 #pragma omp parallel
 {
  #pragma omp for schedule(static) nowait
  //Vector update instead of for-loop for Hy and Hx fields
  for(int i=1;i<xdim-3;++i){
    for(int j=1;j<ydim-3;++j)
      {
        Hxx(i,j)=AA(i,j)*Hxx(i,j)-BB(i,j)*(Ezz(i,j+1)-Ezz(i,j));
        Hyy(i,j)=AA(i,j)*Hyy(i,j)+BB(i,j)*(Ezz(i+1,j)-Ezz(i,j));
      }
  }

 #pragma omp for schedule(static) nowait
  //Vector update instead of for-loop for Ez field
  for(int i=2;i<xdim-3;++i){
     for(int j=2;j<ydim-3;++j)
       {
```

```
            Ezz(i,j)=CC(i,j)*Ezz(i,j)+(Hyy(i,j)-Hyy(i-1,j)-Hxx(i,j)+Hxx(i,j-
1))*DD(i,j);
              }
          }
        }
      ////////////////////////////////////////////////////////////////
      //Mur's abc conditions obtained from Mur's difference equation for
      //forward boundary

      if(n>0)
      {
        for(int j=2;j<ydim-3;++j)
        {
          Ezz(xdim-3,j)=c0efffor*(Ezz(xdim-4,j)+prev_prev_xfor(0,j))\
          -
prev_prev_x_minus_1for(0,j)+c2efffor*(prev_xfor(0,j)+prev_x_minus_1for(0,j))\
          +c3efffor*(prev_x_minus_1for(0,j-
1)+prev_x_minus_1for(0,j+1)+prev_xfor(0,j-1)+prev_xfor(0,j+1));
        }

      }
      //Storage vectors for boundary and boundary-1 values of previous and its
      //previous time steps updated at forward boundary
      prev_prev_xfor=prev_xfor;
      prev_prev_x_minus_1for=prev_x_minus_1for;

      for(int j=0;j<ydim;++j)
        {
          prev_xfor(0,j)=Ezz(xdim-3,j);
        }

      for(int j=0;j<ydim;++j)
        {
          prev_x_minus_1for(0,j)=Ezz(xdim-4,j);
        }

      ////////////////////////////////////////////////////////////////
      //Mur's abc conditions obtained from Mur's difference equation for
      //upward boundary
      if (n>0)
      {

        for(int i=2;i<ydim-3;++i)
        {
          Ezz(1,i)=-
prev_prev_xrev(0,i)+c1effrev*(Ezz(2,i)+prev_prev_x_minus_1rev(0,i))\

+c2effrev*(prev_xrev(0,i)+prev_x_minus_1rev(0,i))+c3effrev*(prev_x_minus_1rev(
0,i-1)\
         +prev_x_minus_1rev(0,i+1)+prev_xrev(0,i-1)+prev_xrev(0,i+1));
```

```
        }

      }

      //Storage vectors for boundary and boundary-1 values of previous and its
      //previous time steps updated at backward boundary
      prev_prev_xrev=prev_xrev;
      prev_prev_x_minus_1rev=prev_x_minus_1rev;

      for(int j=0;j<ydim;++j)
        {
         prev_xrev(0,j)=Ezz(2,j);
        }

      for(int j=0;j<ydim;++j)
        {
         prev_x_minus_1rev(0,j)=Ezz(1,j);
        }

      ////////////////////////////////////////////////////////////////
      //Mur's abc conditions obtained from Mur's difference equation for
      //upward boundary


      if (n>0)
      {

         for(int i=2;i<xdim-3;++i)
         {
            Ezz(i,ydim-3)=c0efffor*(Ezz(i,ydim-4)+prev_prev_yfor(i,0))-
prev_prev_y_minus_1for(i,0)\
            +c2efffor*(prev_yfor(i,0)+prev_y_minus_1for(i,0))\
            +c3efffor*(prev_y_minus_1for(i-
1,0)+prev_y_minus_1for(i+1,0)+prev_yfor(i-1,0)+prev_yfor(i+1,0));
         }
      }

      //Storage vectors for boundary and boundary-1 values of previous and its
      //previous time steps updated at upward boundary
      prev_prev_yfor=prev_yfor;
      prev_prev_y_minus_1for=prev_y_minus_1for;

      for(int i=0;i<xdim;++i)
        {
         prev_yfor(i,0)=Ezz(i,ydim-3);
        }

      for(int i=0;i<xdim;++i)
        {
         prev_y_minus_1for(i,0)=Ezz(i,ydim-4);
```

```
        }
      ////////////////////////////////////////////////////////////
      //Mur's abc conditions obtained from Mur's difference equation for
      //downward boundary

     if(n>0)
      {
        for(int i=2;i<xdim-3;++i)
          {
             Ezz(i,1)=-
prev_prev_yrev(i,0)+c1effrev*(Ezz(i,2)+prev_prev_y_minus_1rev(i,0))\
            +c2effrev*(prev_yrev(i,0)+prev_y_minus_1rev(i,0))\
            +c3effrev*(prev_y_minus_1rev(i-
1,0)+prev_y_minus_1rev(i+1,0)+prev_yrev(i-1,0)+prev_yrev(i+1,0));
          }

      }

      //Storage vectors for boundary and boundary-1 values of previous and its
      //previous time steps updated at downward boundary
      prev_prev_yrev=prev_yrev;
      prev_prev_y_minus_1rev=prev_y_minus_1rev;

      for(int i=0;i<xdim;++i)
        {
         prev_yrev(i,0)=Ezz(i,2);
        }

      for(int i=0;i<xdim;++i)
        {
         prev_y_minus_1rev(i,0)=Ezz(i,1);
        }

      //Mirroring of corner values taking the fact that corners are reached by the fields
from the previous corners
      //in two time steps as S=1/sqrt(2) viz. sqrt(2)*delta(distance between two
corners) is reached in 2 time steps
      Ezz(1,1)=prev_prev_xrev(0,2);
      Ezz(1,ydim-3)=prev_prev_xrev(0,ydim-4);
      Ezz(xdim-3,1)=prev_prev_x_minus_1for(0,2);
      Ezz(xdim-3,ydim-3)=prev_prev_x_minus_1for(0,ydim-4);

      // Source condition incorporating given free space wavelength 'wav'
      // and having a location at the left end of central waveguide just
      //% after left boundary
      double tstart=1;
      double N_lambda=wav*1e-6/delta;

      for(int j=14*factor;j<18*factor;++j)
        {
```

```
          Ezz(3,j)=sin(((2*pi*(c/(delta*N_lambda))*(n-tstart)*deltat)));
        }

    }//time_tot
    buffer.cpp_to_ndarray_with_strides_2d(converted,Ezz._data(),xdim,ydim);
    return converted;
  }//abc==1
}
#include <boost/python.hpp>

BOOST_PYTHON_MODULE(splitter)
{
      using namespace boost::python;
      Py_Initialize();
      np::initialize();
      def("compute", compute);

}
```

Ek C: 2B ZUSF Ayrıştırıcı ve Birleştirici Dalga Kılavuzu Testi IT++ Python Simülasyon Kodu

```
import sys
import math as mt
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.image as mpimg
import time
import splitter
import ctypes
import matplotlib as mpl
from ctypes import cdll
import parula
# Conditions for simulation
pml=1;
abc=0;
factor=6;
S=1/2**0.5;
epsilon0 = (1/(36*np.pi))*1e-9;
mu0=4*np.pi*1e-7;
c=3e+8;
delta=0.25e-6/factor;
deltat=S*delta/c;
time_tot=1000;
epsilon0 = (1/(36*np.pi))*1e-9;
ydim=32*factor;
```

```
xdim=80*factor;
wav=2;
index=1.5;
t=time.time();
# Boost Python function
np_arr_2d=splitter.compute(pml,abc,factor,S,epsilon0,mu0,c,delta,deltat,xdim,ydim,time_tot,wav,index);
elapsed=time.time()-t;
# Plotting options
if pml==1:
   plt.title('Color-scaled image plot of Ez in waveguide splitter/combiner with PML boundary',fontsize=9);

plt.imshow(np.transpose(np_arr_2d),extent=[0,delta*1e+6*xdim,0,delta*1e+6*ydim],cmap=parula.parula_map,aspect='equal',vmin=-1,vmax=1);
   plt.xlabel('x (in um)',fontsize=20);
   plt.ylabel('y (in um)',fontsize=20);
   plt.xticks(fontsize=15);
   plt.yticks(fontsize=15);
   plt.colorbar();
elif abc==1:
   plt.title('Color-scaled image plot of Ez in waveguide splitter/combiner with Mur ABC boundary',fontsize=9);

plt.imshow(np.transpose(np_arr_2d),extent=[0,delta*1e+6*xdim,0,delta*1e+6*ydim],cmap=parula.parula_map,aspect='equal',vmin=-1,vmax=1);
   plt.xlabel('x (in um)',fontsize=20);
   plt.ylabel('y (in um)',fontsize=20);
   plt.xticks(fontsize=15);
   plt.yticks(fontsize=15);
   plt.colorbar();
else:
   print('Invalid baundry condition for simulation');
```

Ek D: Makefile

```
TMP_DIR        := $(HOME)/sw_temp/itpp_04.03.01_20170302_1603/itpp-4.3.1_install
TMP_DIR_INCLUDE := $(TMP_DIR)/include
TMP_DIR_LIB     := $(TMP_DIR)/lib
# location of the Python header files
PYTHON_VERSION = 2.7
PYTHON_INCLUDE = /usr/include/python2.7
# location of the Boost Python include fliles and library
BOOST_INC = /home/gtuee/waveguides/boost_lib/include
BOOST_LIB = /home/gtuee/waveguides/boost_lib/lib

LLIB = /home/gtuee/os_software/software_temp/acml_04_04_00_b32/gfortran32/lib
```

```
#Helper Funcs Include Dir
HELPER0_INC = /home/gtuee/waveguides/tests/splitter_combiner_itpp/vec/
HELPER1_INC = /home/gtuee/waveguides/tests/splitter_combiner_itpp/mat/
HELPER0_LIB = /home/gtuee/waveguides/tests/splitter_combiner_itpp/vec/lib
HELPER1_LIB = /home/gtuee/waveguides/tests/splitter_combiner_itpp/mat/lib

ARMADILLO_INC = /home/gtuee/waveguides/arma_lib/include
ARMADILLO_LIB = /home/gtuee/waveguides/arma_lib



PATH_LINK:=$(LLIB)/:$(BOOST_LIB)/:$(TMP_DIR_LIB)/:$(HELPER0_LIB)/:$
(HELPER1_LIB)/:$(ARMADILLO_LIB)
# compile mesh classes
TARGET = splitter
MODULE_NAME = splitter

$(TARGET).so: $(TARGET).o
	g++ -std=c++11 -shared $(TARGET).o -L$(BOOST_LIB) -lboost_python27 -
lboost_numpy27  -L$(TMP_DIR_LIB)  -litpp -L$(LLIB) -L$(ARMADILLO_LIB) -
larmadillo -lacml -L$(HELPER0_LIB) -lhelp_functions_vec      -
L$(HELPER1_LIB) -lhelp_functions_mat -o $(TARGET).so
	patchelf --set-rpath $(PATH_LINK) $@
$(TARGET).o: $(TARGET).cpp
	g++ -std=c++11 -I$(PYTHON_INCLUDE) -I$(TMP_DIR_INCLUDE) -
I$(BOOST_INC)  -I$(HELPER0_INC) -I$(HELPER1_INC) -
I$(ARMADILLO_INC) -fopenmp  -fPIC -c $(TARGET).cpp
clean:
	rm  $(TARGET).o $(MODULE_NAME).so
```