T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI DOKTORA PROGRAMI
2023-DR-017

KOMPLEKS GEOMETRİLERDE KISMİ TÜREVLİ DİFERANSİYEL DENKLEMLERİN DERİN ÖĞRENME YAKLAŞIMLARI İLE NÜMERİK ÇÖZÜMLERİ

ÖZCAN KOLYİĞİT
DOKTORA TEZİ

DANIŞMAN
Doç. Dr. Korhan GÜNEL

Bu tez Aydın Adnan Menderes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FEF-22026 proje numarası ile desteklenmiştir.

AYDIN-2023

KABUL VE ONAY

T.C. Aydın Adnan Menderes Üniversitesi Fen Bilimleri Enstitüsü Matematik Anabilim Dalı Doktora öğrencisi Özcan KOLYİĞİT tarafından hazırlanan "KOMPLEKS GEOMETRİLERDE KISMİ TÜREVLİ DİFERANSİYEL DENKLEMLERİN DERİN ÖĞRENME YAKLAŞIMLARI İLE NÜMERİK ÇÖZÜMLERİ" başlıklı tez, aşağıdaki jüri tarafından Doktora Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 04/08/2023

Üye :	Doç. Dr. Refet POLAT	Yaşar Üniversitesi	…….
Üye (T.D.) :	Doç. Dr. Korhan GÜNEL	Aydın Adnan Menderes Ü.	…….
Üye :	Dr. Öğr Üyesi Elgin KILIÇ	Ege Üniversitesi	…….
Üye :	Dr. Öğr. Üyesi Rıfat AŞLIYAN	Aydın Adnan Menderes Ü.	…….
Üye :	Dr. Öğr Üyesi Emrah YILDIRIM	Aydın Adnan Menderes Ü.	…….

ONAY:

Bu tez Aydın Adnan Menderes Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca yukarıdaki jüri tarafından uygun görülmüş ve Fen Bilimleri Enstitüsünün ……..… tarih ve …… sayılı oturumunda alınan …… numaralı Yönetim Kurulu kararıyla kabul edilmiştir.

Prof. Dr. Mustafa SÜRMEN

Enstitü Müdürü

TEŞEKKÜR

Başta bana her konuda destek ve yol gösterici olan danışman hocam Sayın Doç. Dr. Korhan GÜNEL olmak üzere doktora sürecinde benden desteğini esirgemeyen Sayın Dr. Öğr. Üyesi Rıfat AŞLIYAN'a, Sayın Doç. Dr. Refet POLAT'a ve diğer hocalarımıza sonsuz teşekkürlerimi sunarım.

Bu tezdeki uygulamalar, Aydın Adnan Menderes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından desteklenen FEF-22026 numaralı proje kapsamında kurulan ADÜ-BAP Fizik/Matematik HPC sistemi sunucuları üzerinde gerçekleştirilmiştir. Bu Yüksek Başarımlı Hesaplama sisteminin kurulumunda rol alan Fizik Bölümü öğretim üyeleri Sayın Prof. Dr. Ethem AKTÜRK, Sayın Doç. Dr. Fatih ERSAN ve Matematik Bölümü öğretim üyesi Doç. Dr. Korhan GÜNEL ile BAP komisyon üyeleri ve Bilgi İşlem Daire Başkan V. Sinan BALCI'ya teşekkürü borç bilirim.

Özcan KOLYİĞİT

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

GPS **:** Genelleştirilmiş Desen Arama (Generalised Pattern Search)

HPC **:** Yüksek Başarımlı Hesaplama (High Processing Computing)

IBC **:** İç Sınır Koşulları (Inner Boundary Conditions)

MADS **:** Örgü Uyarlamalı Doğrudan Arama (Mesh Adaptive Direct Search)

MC **:** İç Bölgesinde Kesişen Üç Dairesel Boşluğu Bulunan Çözüm Uzayı

MSE **:** Ortalama Karesel Hata (Mean Squared Error)

NOMAD **:** Örgü Uyarlamalı Doğrudan Arama ile Doğrusal Olmayan Optimizasyon (Nonlinear Optimization with the MADS)

OBC **:** Dış Sınır Koşulları (Outer Boundary Conditions)

PSO **:** Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)

ReLU **:** Doğrultulmuş Lineer Birim (Rectified Linear Unit) fonksiyonu

ResNet **:** Rezidü Yapay Sinir Ağı (Residual Neural Network)

STC : İç Bölgesinde Kesişmeyen İki Dairesel Boşluğu Bulunan Çözüm Uzayı (Separated Two Circles)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

ÖZET

KOMPLEKS GEOMETRİLERDE KISMİ TÜREVLİ DİFERANSİYEL DENKLEMLERİN DERİN ÖĞRENME YAKLAŞIMLARI İLE NÜMERİK ÇÖZÜMLERİ

Kolyiğit Ö. Aydın Adnan Menderes Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Programı, Doktora Tezi, Aydın, 2023.

Amaç: Bu tez, son yıllarda büyük ilerleme kaydeden derin öğrenme yaklaşımlarını kullanarak üzerinde çakışan veya çakışmayan deliklerin bulunduğu bölgelerde kısmi türevli diferansiyel denklemlerin sayısal çözümlerini elde etmeyi amaçlamaktadır.

Materyal ve Yöntem: Çalışmada öncelikle ilk yapılan çalışmalardan günümüze kadar bu alanda yapılmış yayınlar incelenmiş, literatür taraması yapılmıştır. Kısmi türevli diferansiyel denklemlerin çözümlerini elde edecek derin öğrenme ağı modeli ve uygun topoloji belirlenmeye çalışılmıştır. Çözüm uzayında DistMesh algoritması üçgenselleştirme yoluyla düğüm noktaları elde edilmiştir. Güncel çalışmalarda çoğunlukla ısı yayılım denklemi, ısı dağılım denklemi, Poisson denklemi ve Black-Scholes denklemi gibi Fizik ve Ekonomi alanlarında kullanılan denklemlerin üzerinde çalışmalar yapıldığı görülmüş ve tez çalışmasında üzerinde çalışılmak üzere iki farklı Poisson denklemi belirlenmiştir. Optimizasyon algoritmaları üzerine çalışılmış ve tez konusuna uygun optimizasyon algoritması belirlenmeye çalışılmıştır. Derin yapay sinir ağı modeli olarak Rezidü Yapay sinir ağı modeli, optimizasyon algoritması olarak da Örgü Uyarlamalı Doğrudan Arama ile Doğrusal Olmayan Optimizasyon (NOMAD) algoritması seçilmiştir.

Bulgular: Çalışmada geliştirilen model, her iki örnek için, iki farklı sınır koşulları ve iki farklı çözüm uzayı üzerinde test edilmiş; elde edilen yapay sinir ağı çözümlerinin gerçek çözümlere düşük hata miktarları ile yakınsadığı görülmüştür. Modelin farklı örnekler ve farklı çözüm uzayları ile kısmi diferansiyel denklemlerin üzerinde çakışan veya çakışmayan delikler bulunan bölgelerde nümerik çözümlerinin elde edilmesinde başarılı olduğu görülmüştür.

Sonuç: Bu çalışmada geliştirilen model ile Rezidü yapay sinir ağı modeli ve türev içermeyen bir optimizasyon (NOMAD) algoritması kullanılarak farklı yapıdaki bölgelerde kısmi türevli diferansiyel denklemlerin nümerik çözümlerinin elde edilebileceği gösterilmiştir.

Anahtar Kelimeler: Rezidü Derin Yapay Sinir Ağları, DistMesh Üçgenselleştirme, Örgü Uyarlamalı Doğrudan Arama Algoritması, Kısmi Diferansiyel Denklemler, Optimizasyon.

ABSTRACT

NUMERICAL SOLUTIONS OF PARTIAL DIFFERENTIAL EQUATIONS ON COMPLEX GEOMETRIES WITH DEEP LEARNING APPROACHES

Kolyiğit Ö. Aydın Adnan Menderes University, Graduate School of Natural and Applied Sciences, Mathematics Program, Doctorate Thesis, Aydın, 2023

Objective: This thesis aims to obtain numerical solutions of partial differential equations in regions with overlapping or non-overlapping holes by using deep learning approaches, which have made great progress in recent years.

Material and Methods: In this study, firstly, the publications made in this field from the first studies to the present day have been examined and a literature review has been made. The deep learning network model and the appropriate topology to obtain the solutions of partial differential equations are tried to be determined. Nodes were obtained by triangularization via DistMesh algorithm in the solution space. In the current studies, it has been observed that the equations used in the fields of Physics and Economics such as heat diffusion equation, heat dissipation equation, Poisson equation and Black-Scholes equation are mostly studied, and two different Poisson equations have been determined to be studied in the thesis. Optimization algorithms were studied and an optimization algorithm suitable for the thesis subject was tried to be determined. The residual artificial neural network model was chosen as the deep artificial neural network model and the Nonlinear Optimisation with Mesh Adaptive Direct Search (NOMAD) algorithm was chosen as the optimisation algorithm.

Results: The model developed in this study was tested on two different boundary conditions and two different solution spaces for both examples and it was observed that the neural network solutions converged to the real solutions with low error amounts. The model was found to be successful in obtaining numerical solutions of partial differential equations with different examples and different solution spaces in regions with overlapping or non-overlapping holes.

Conclusions: In this study, it is shown that numerical solutions of partial differential equations in regions with different structures can be obtained by using the residual artificial neural network model and a non-derivative optimisation (NOMAD) algorithm.

Keywords: Residual Deep Neural Networks, DistMesh Triangulation, Mesh Adaptive Direct Search Algorithm, Partial Differential Equations, Optimization

1. GİRİŞ

Diferansiyel denklemler gerçek dünya probleminin matematiksel modellenmesinde etkin bir araç olarak kullanılmaktadır. Diferansiyel denklemlerin analitik çözümlerinin elde edilmesinin mümkün olmadığı durumlarda iterasyon temelli nümerik çözüm yöntemlerine alternatif olarak Lee ve Kang (1990) tarafından yapay sinir ağları kullanılması önerilmiştir.

Bu gelişme ile yapay sinir ağları, adi ve kısmi diferansiyel denklemlerin nümerik çözümlerinde kullanılmaya başlanmıştır. Meade ve Fernandez (1994) lineer adi diferansiyel denklemlerin nümerik çözümü için ileri beslemeli yapay sinir ağı modeli kullanılmıştır.

Dissanayake ve Phan-Thien (1994) kısmi diferansiyel denklemlerin yapay sinir ağları ile çözümü için nümerik bir metot önerilmiştir. Çalışmada lineer ve lineer olmayan iki örneğe yer verilmiştir.

Lagaris vd. (1998) başlangıç ve sınır değer problemlerinin yapay sinir ağları ile çözümü için iki bölümden oluşan bir deneme çözümü kullanılarak uygulanan nümerik bir metot önermişlerdir. Deneme çözümünün ilk bölümü başlangıç ya da sınır değerleri sağlamakta ve parametre içermemektedir, ikinci bölüm ileri beslemeli yapay sınır ağı ile değiştirilen parametreleri/ağırlıkları içermektedir.

Aarts ve Van Der Veer (2001) Evrimsel algoritma kullanan bir yapay sinir ağı geliştirerek Kısmi diferansiyel denklemler için nümerik çözümler elde etmeye çalışmışlardır.

Bu tez beş bölümden oluşmaktadır. Tezin takip eden 2. bölümünde kaynak özetlerine yer verilmiştir. 3. bölümde ise kısmi diferansiyel denklemlerin nümerik çözümlerini elde etmede birlikte kullanılması önerilen yöntemlere değinilmiştir. Bu amaçla 3. Bölümde Rezidü yapay sinir ağı modelleri ve ağın eğitiminde kullanılan optimizasyon yöntemi olan Örgü Uyarlamalı Doğrudan Arama yaklaşımı anlatılmaktadır. Yine aynı bölümde kısmi diferansiyel denklemin tanımlı olduğu bölgenin üçgenleştirilmesinde kullanılan DistMesh algoritmasından bahsedilmiştir. Üçgenleştirme işlemi sonucunda elde edilen düğümler Rezidü ağının denetimsiz öğrenme yaklaşımıyla eğitiminde kullanılmaktadır. 4. Bölümde eliptik bir kısmi diferansiyel denklem olan Poisson denkleminin çakışan veya çakışmayan delikler içeren bir bölge üzerindeki çözümleri tezde önerilen yöntem kullanılarak farklı sınır koşulları altında elde edilmiştir. Deneysel çalışmalardan elde edilen bulgular tezin 5.

Bölümünde tartışılmış ve ilgili sonuçlar verilmiştir. Tezin Ek Bölümü tez süresince gerçekleştirilen Julia implementasyonlarından birine ait örnek kodları içerir.

2. KAYNAK ÖZETLERİ

Derin öğrenme yaklaşımları ile kısmi diferansiyel denklemlerin nümerik çözümleri üzerine birçok çalışma yayınlanmıştır.

Çok boyutlu parabolik kısmi diferansiyel denklemlerin nümerik çözümlerini Derin yapay sinir ağları ile elde edilmiş, önerilen metodun etkinliği ve doğruluğu Tensorflow kullanılarak modellenerek gösterilmiştir (E vd., 2017).

Han vd. (2017) lineer olmayan parabolik kısmi diferansiyel denklem içeren problemlerin nümerik çözümü için derin yapay sinir ağı modeli önermiştir.

100 boyutlu Black-Scholes-Barenblatt denklemi, 100 boyutlu Hamilton Jacobi-Bellman denklemi ve 20 boyutlu Allen-Cahn denkleminin çözümleri için derin yapay sinir ağı modeli önerilmiştir (Raissi, 2018).

Berg ve Nyström (2018) önerdikleri tümleşik derin sinir ağı (unified deep neural network) modeli ile dörtgensel olmayan iki boyutlu bir tanım bölgesi üzerinde durağan konveksiyon denkleminin çözülebileceğini göstermişlerdir.

Akışkanlar, kuantum mekaniği, reaksiyon-yayılma sistemleri, doğrusal olmayan sığ su dalgası yayılması gibi alanlarda çalışılan problemlerin çözümleri için derin yapay sinir ağı modeli kullanılmıştır (Raissi vd., 2019).

Berg ve Nyström (2019) tarafından hava ölçüm istasyonlarından elde edilen büyük veri setlerinin incelenmesiyle ortaya çıkan kısmi diferansiyel denklemler derin yapay sinir ağı modeli ile çözülmeye çalışılmıştır.

Eliptik kısmi diferansiyel denklemlerin nümerik çözümleri için derin yapay sinir ağları ve en küçük kareler yöntemi kullanılan bir model önerilmiştir (Cai vd., 2019).

Bir ve iki boyutlu zamana bağlı kısmi diferansiyel denklemlerin evrişimli LSTM yapay sinir ağı ile çözümleri elde edilmiştir (Asem, 2020).

Zakeri vd. (2020) sonlu farklar metodu ve Leaky Relu aktivasyon fonksiyonu kullanan bir derin yapay sinir ağı ile iki boyutlu ısı yayılma probleminin çözümü elde edilmiştir.

Feynman-Kac formülünü esas alan bir derin öğrenme algoritması, yüksek mertebeden doğrusal Kolmogorov kısmi diferansiyel denklemlerin nümerik çözümü için önerilmiştir (Berner vd., 2020).

Bir ve iki boyutlu stokastik kısmi diferansiyel denklemlerin rezidü derin yapay sinir ağları ve fizik-bilgili (physicsinformed) maliyet fonksiyonu kullanılarak çözümleri elde edilmiştir (Karumuri vd., 2020).

Gör (2020) tarafından birinci ve ikinci mertebeden lineer başlangıç değer problemleri Dirichlet sınır koşullarını içeren ikinci mertebeden lineer ve lineer olmayan diferansiyel denklemler ve birinci mertebeden lineer diferansiyel denklem sistemlerinin nümerik çözümleri ileri beslemeli tek ara katmanlı yapay sinir ağları ile Parçacık Sürü Optimizasyonu, Kütle Çekim Arama Algoritması ve Yapay Arı Kolonisi Algoritması ve Karınca Koloni Optimizasyon Algoritması kullanılarak elde edilmiştir.

Eskiizmirliler, Günel ve Polat (2021) ekonomi alanında kullanılan kısmi türevli bir diferansiyel denklem olan Black–Scholes denklemini ileri besmeli yapay sinir ağı ile çözmeyi önermişlerdir. Önerilerinde türev içermeyen popülasyon tabanlı bir optimizasyon yöntemi olan Parçacık Sürü Optimizasyonu (PSO) yaklaşımını kullanarak oluşturdukları ağ modelini eğitmiş ve elde ettikleri sonucu Gradyan Düşüm (Graident Descent) yöntemi ile eğitilen ağ modelinin ürettiği sonuçlarla karşılaştırmışlardır.

Beck vd. (2021) tarafından Stokastik Diferansiyel denklemlerin ve Kolmogorov kısmi diferansiyel denklemlerin çözümü için derin yapay sinir ağı modeli önerilmiştir.

Li vd. (2020) D3M (A Deep Domain Decomposition Method) adını verdikleri metot ile bölge ayrıştırma yaparak kısmi diferansiyel denklemlerin paralel olarak uygulanan yapay sinir ağları kullanılarak nümerik çözümleri elde etmişlerdir.

Kısmi türevli diferansiyel denklemleri derin öğrenme yaklaşımları ile alt problemlere ayırarak nümerik çözümleri üzerine çalışılmış; geliştirilen model, Hamilton–Jacobi–Bellman (HJB) denklemi, doğrusal olmayan Black–Scholes denklemi, yarıdoğrusal ısı denklemi gibi problemler üzerinde test edilmiştir (Beck vd., 2019).

Eliptik kısmi türevli diferansiyel denklemlerin karesel bölge üzerinde nümerik çözümlerini elde etmek amacıyla bölgenin daha küçük karesel bölgelere ayrılarak hatanın azaltılmaya çalışıldığı ve rezidü derin yapay sinir ağı modelinin kullanıldığı bir metot önerilmiştir (Z. Wang vd., 2021).

Parametrik Difüzyon denkleminin nümerik çözümü için, Geist vd. (2021) tarafından Evrişimli Yapay sinir ağları kullanılan bir yöntem önerilmiştir.

3. MATERYAL VE YÖNTEM

Bu bölümde derin yapay sinir ağları kullanılarak dörtgensel olmayan bölgeler üzerinde kısmi diferansiyel denklemlerin nümerik çözümlerinin elde edilmesinde kullanılan metotlardan bahsedilecektir. Öncelikle çözüm uzayı üzerinde düğüm noktaları elde etmemizi sağlayan DistMesh algoritmasından, daha sonra problemin başlangıç ve sınır koşullarını sağlayan deneme çözümünün nasıl inşa edildiğinden ve son olarak kısmi diferansiyel denklemin nümerik çözümünü elde etmede kullanılan sinir ağı modelinin eğitimi aşamasında oluşan hatayı minimize etmek için kullanılan türev içermeyen optimizasyon yönteminden bahsedilecektir.

3.1. Kısmi Türevli Diferansiyel Denklemlerin Yapay Sinir Ağı Çözümleri

Bu bölümde ikinci mertebeden kısmi türevli diferansiyel denklemlerin tanımına, yapay sinir ağları ile nümerik çözümlerini elde etmek için kullandığımız deneme çözümünün ve maliyet fonksiyonunun nasıl elde edildiğine yer verilecektir.

Tanım 3.1.1. Kısmi diferansiyel denklemler, birden fazla bağımsız değişkenin ve en az bir bağımlı değişkenin bulunduğu fonksiyonun bağımsız değişkenlere göre kısmi türevlerinin de yer aldığı denklemlerdir. x ve y bağımsız değişken olmak üzere u bilinmeyen fonksiyonuna bağlı kısmi türevli diferansiyel denklem (3.1) ile tanımlanmıştır.

$$G\left(x, y; u, u_x, u_y, u_{xx}, u_{yy}, u_{xy}, \dots\right) = f(x, y) \tag{3.1}$$

Verilen (3.1) eşitliğinde $u = u(x, y)$ bağımlı değişkeni için

$$u_x = \frac{\partial u}{\partial x}, u_y = \frac{\partial u}{\partial y}, u_{xx} = \frac{\partial^2 u}{\partial x^2}, u_{yy} = \frac{\partial^2 u}{\partial y^2}, u_{xy} = \frac{\partial^2 u}{\partial x \partial y}$$

değerleri u değişkeninin x ve y bağımsız değişkenlerine göre birinci ve ikinci mertebeden kısmi türevlerini ifade etmektedir.

Tez çalışmasında kullanılan Poisson denklemini ele alalım. İki boyutlu Poisson denklemi (3.2) denklemi ile verilmiştir (Lagaris vd., 1998).

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y) \;\; x \in [0,1] \, , y \in [0,1] \tag{3.2}$$

Denklemin Dirichlet sınır koşulları (3.3) eşitlikleri ile verilmiştir.

$$u(0,y) = f_0(y) \, , u(1,y) = f_1(y) \; u(x,0) = g_0(x) \, , u(x,1) = g_1(x) \tag{3.3}$$

Başlangıç ve sınır koşullarını sağlayan deneme çözümü iki bölümden oluşmaktadır. İlk bölüm yalnızca başlangıç ve sınır değerlerini sağlamakta, ağ parametrelerini ve ağırlıklarını içermemektedir. İkinci bölüm ise başlangıç ve sınır değerlerine etki etmeyen $\mathrm{ResNet}(\mathrm{x},\mathrm{y};\mathbf{W})$ yapay sinir ağı modelinin (x,y) noktasındaki çıktısını içeren bölümdür. Deneme çözümü (3.4) eşitliğinde verilmiştir (Z. Wang vd., 2021).

$$\begin{aligned} \mathrm{u_T}(\mathrm{x},\mathrm{y}) = {} & (1-\mathrm{x})f_0(y) + \mathrm{x}f_1(y) + (1-\mathrm{y})[g_0(x) - \{(1-\mathrm{x})\, g_0(0) + x g_0(1)\}] \\ & + \mathrm{y}[g_1(x) - \{(1-\mathrm{x})g_1(0) + \mathrm{x}g_1(1)\}] \\ & + \mathrm{xy}(1-\mathrm{x})(1-\mathrm{y})\, \mathrm{ResNet}(\mathrm{x},\mathrm{y};\mathbf{W}) \end{aligned} \tag{3.4}$$

Problemde bulunan terimlere göre deneme çözümünün x ve y ye göre kısmi türevleri de kullanılmaktadır. Poisson denklemi için deneme çözümünün x ve y ye göre ikinci mertebeden türevleri hesaplanmıştır.

Tez çalışmasında kullanılan ortalama karesel hata MSE (Mean Squared Error) eşitliğinde verilmiştir.

$$\begin{aligned} & \mathrm{MSE} \\ & = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\{ G\left(x_i, y_j; u_\mathrm{T}(x_i,y_j), \frac{\partial u_\mathrm{T}}{\partial x}(x_i,y_j), \frac{\partial u_\mathrm{T}}{\partial y}(x_i,y_j), \frac{\partial^2 u_\mathrm{T}}{\partial x^2}(x_i,y_j), \frac{\partial^2 u_\mathrm{T}}{\partial y^2}(x_i,y_j), \frac{\partial^2 u_\mathrm{T}}{\partial x \partial y}(x_i,y_j), \dots \right) \right. \\ & \left. - f(x_i,y_j) \right\}^2 \end{aligned} \tag{3.5}$$

(3.5) eşitliğinde $n.m$ çözüm uzayının üçgenselleştirilmesinden elde edilen nokta sayısını, $u(x_i,y_j)$ değeri (x_i,y_j) noktasındaki gerçek çözüm değerini, $\mathrm{u_T}(x_i,y_j)$ ise yapay sinir ağı çözümünün değerini ifade etmektedir. Eğitim kümesinin veya test kümesinin hata miktarı hesaplanırken kümelerde bulunan her nokta için $u(x,y)$ ve $\mathrm{u_T}(\mathrm{x},\mathrm{y})$ hesaplanır, (3.5) eşitliği verilen formül ile hata miktarı elde edilir. Amacımız bu hata miktarını azaltmak yani yapay sinir ağı çözümünün gerçek çözüme yaklaşmasını sağlamaktır.

3.2. DistMesh

Tez çalışmasında çözüm uzayı olarak dörtgensel olmayan kompleks bir bölge seçilmiştir. Bölgenin birbiriyle çakışmayan veya çakışan delikler içerdiği varsayılmıştır. Bu bölge üzerinde ağın eğitilmesi için kullanılacak olan düğüm noktalarının belirlenmesi amacıyla DistMesh algoritması kullanılmıştır.

DistMesh algoritması, Delaunay üçgenleştirme (Delaunay triangulation) algoritmasını esas alan bir örgü (mesh) oluşturma algoritmasıdır. Yöntem, 2 ve 3 boyutlu yüzeylerin belirlenen dağılım ile üçgensel bölgelere ayrılmasını ve üçgenlerin köşe noktaları ile yüzeyler üzerinde düğüm noktaları elde edilmesini sağlamaktadır.

Persson ve Strang tarafından 2004 yılında ilk olarak MATLAB programlama dilinde yazılmış, daha sonra diğer programlama dillerine uyarlanmıştır (Strang, 2004), (Persson, 2005).

Kartezyen koordinat düzlemindeki her nokta kümesi, Delaunay algoritması ile köşeleri bu noktalar olan üçgensel bölgelere ayrılabilir. Topolojik olarak üçgenlerin kenarları noktalar arasındaki bağlantılara ve noktalar da üçgenin köşelerine karşılık gelir.

Her kenar, mevcut uzunluğu l ve genişletilmemiş uzunluğu l_0 değerine bağlı bir "güç uzanımı (force displacement)" adı verilen $f(l, l_0)$ ilişkisine sahiptir.

Sınırlardaki düğüm noktalarında, sınıra dik olan ve noktaların sınır dışına çıkmasına engel olacak kadar büyüklüğe sahip bir tepki kuvveti vardır. Düğüm noktalarının konumu temel bilinmeyendir. Bu düğüm noktalarının konumu, yapıdaki statik kuvvet dengesi çözülerek bulunur. İstenilen kenarlar arasındaki bağıl uzunluğun $h(x, y) = 1$ olacak şekilde kenarlardaki kuvvetin hemen hemen eşit olmasıdır. Böylece iyi şekillendirilmiş bir ağ elde edilebilir.

$$\mathbf{p} = [\, \boldsymbol{x}, \boldsymbol{y} \,] \tag{3.6}$$

Kuvvet dengesinin çözülebilmesi için tüm düğüm noktalarının apsis ve ordinat değerleri **p** matrisinde birleştirilir. **p** matrisi (3.6)'de verilmiştir.

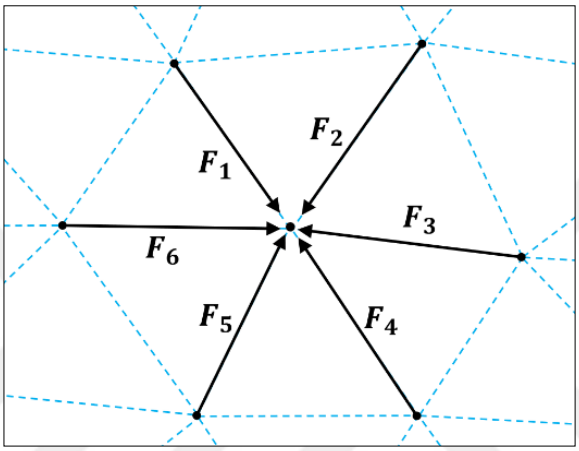
p matrisinin boyutu, N nokta sayısı olmak üzere $N \times 2$'dir ve (3.6) eşitliğinde **x** ve **y** vektörleri sırasıyla verilen N noktanın apsis ve ordinat değerlerini içeren birer sütun vektörüdür.

$$F(\mathbf{p}) = \left[F_{int,x}(\mathbf{p})\ \ F_{int,y}(\mathbf{p})\right] + \left[F_{ext,x}(\mathbf{p})\ \ F_{ext,y}(\mathbf{p})\right] \tag{3.7}$$

Kuvvet vektörü $F(\mathbf{p})$, her düğüm noktası için yatay ve dikey bileşene sahiptir ve (3.7) eşitliği ile tanımlanır.

Burada F_{int} kenarlardaki dahili kuvveti, F_{ext} ise kenarlardaki harici (sınırlardan gelen tepki kuvveti) kuvveti ifade eder.

Bir düğüm noktasına uygulanan harici kuvvetler Şekil 3.1 ile gösterilmiştir (L. Wang ve Persson, 2015).



Şekil 3.1. Bir düğüm noktasına uygulanan harici kuvvetler.

Şekil 3.1'de görüleceği üzere, $F(\mathbf{p})$, düğüm noktalarını birleştiren çizgilerin yani üçgenlerin kenarlarının topolojisine bağlıdır. DistMesh algoritmasında bu yapı Delaunay üçgenleştirme algoritması ile elde edilen düğüm noktaları ile elde edilir. Delaunay üçgenleştirme algoritması, girdi noktalarını köşe kabul eden, birbirleriyle üst üste gelmeyen, kenarlarında başka girdi noktası olmayan üçgenlerden oluşan ve her kenarın en fazla iki üçgenin kenarı olduğu bir yapı oluşturmamızı sağlar. Bu bağlamda üçgenleştirme elde edilen üçgenlerin iç açılarının ölçülerini büyütmeye çalışır. Delaunay tarafından noktaların konumu ve kenarlar değiştirildiğinden $F(\mathbf{p})$ kuvvet vektörü $\mathbf{p}$'ye bağlı sürekli olmayan bir fonksiyondur. $F(\mathbf{p}) = 0$ sistemi $\mathbf{p}$ noktalarının konumlarının dengelenmesi ile çözülür. Tanımlanan bu problem, kuvvet fonksiyonunun süreksiz olması ve sınırlardaki dış tepki kuvveti sebebiyle çözümü zor bir problemdir.

$$\frac{d\mathbf{p}}{dt} = F(\mathbf{p}), \qquad t \geq 0. \tag{3.8}$$

$F(\mathbf{p}) = 0$ sisteminin çözümü için kullanılabilecek basit yaklaşım yapay zamana bağlılık, bazı $\mathbf{p}(0) = \mathbf{p}_0$ noktaları için (3.8) eşitliğinde verilen adi diferansiyel denklem sistemi ile tanımlanır.

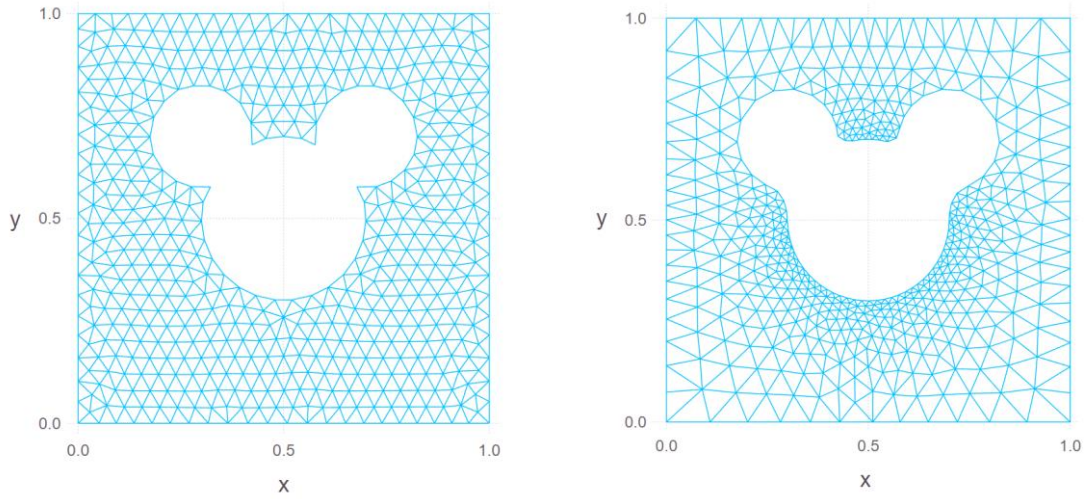
Eğer (3.8) eşitliği ile verilen denklem sisteminin bir çözümü bulunabilirse, bu çözüm $F(\mathbf{p}) = 0$ sistemini sağlar. (3.8) eşitliği ile verilen diferansiyel denklem sistemi İleri Euler metodu ile çözüme yaklaştırılır. Bölünmüş zaman $t_n = n\Delta t$ değerine bağlı yaklaşık çözüm $\mathbf{p}_n \approx \mathbf{p}(t_n)$ ile gösterilecek olursa (3.8) eşitliğinin çözümü (3.9) eşitliği ile elde edilir.

$$\mathbf{p}_{n+1} = \mathbf{p}_n + \Delta t F(\mathbf{p}_n) \tag{3.9}$$

(3.9) eşitliğinde zamana bağlı konum güncelleştirme yapılır. Kısaca kuvvet fonksiyonu değişirken $\mathbf{p}_n$ noktalarının konumu bulunur ve böylece topoloji yani mevcut noktalar kümesinin üçgenleştirilmesi de elde edilir. Düğüm noktalarının konumları güncellenirken noktalar sınırların dışına çıkabilir. Bu durumda sınırlardaki tepki kuvvetiyle karşılaşırlar. Nokta sınıra yakın bir noktaya taşınır. Sınır boyunca hareket edebilirler fakat sınırların dışına çıkamazlar.

Tez çalışmasında çözüm uzayı olarak dörtgensel olmayan, kompleks bir uzay seçilmiştir. Distmesh algoritması, problemin tanımlı olduğu bölgelerin normal ya da normal olmayan dağılım gösteren noktalarla üçgenleştirilmesi için kullanılmıştır. Normal olmayan dağılımda bölge içerisinde tanımlı delik (dairesel bölge) etrafında daha fazla düğüm noktası elde edilmektedir.

Örnek bir tanım bölgesi üzerinde, normal ve normal olmayan dağılım kullanılarak elde edilen düğüm noktaları Şekil 3.2 ile gösterilmiştir.



Şekil 3.2. DistMesh algoritması ile elde edilen düğüm noktaları.

3.3. Rezidü (Artık) Yapay Sinir Ağları

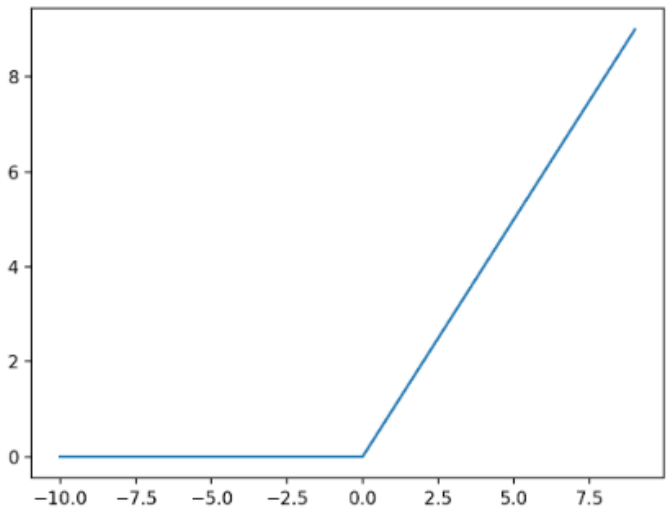
Bu bölümde tez çalışmasında kullanılan yapay sinir ağı modelinden bahsedilecektir. Derin yapay sinir ağları, yapay sinir ağlarının ve teknolojinin gelişmesi ile ortaya çıkan yapay sinir ağlarının bir alt dalıdır. Derin yapay sinir ağları çokça gizli katmana sahip olduğu için eğitilmesi güçtür. Bu nedenle Rezidü Yapay sinir ağları önerilmiştir (He vd., 2016).

Rezidü yapay sinir ağları (ResNet), tam bağlantılı yapay sinir ağlarından farklı olarak katmanları veya bağlantıları atlayarak tekrar bağlantı kurar. ResNet modelinde katmanları atlamanın iki nedeni vardır. Bunlardan ilki katmanlar arasındaki bağlantının değiştirilmesini sağlamayacak kadar küçük türev değerlerinin (Vanishing gradient problem) elde edilmesini engellemek, ikincisi ise ağa çok fazla katman eklenmesi ile ortaya çıkan aşırı eğitim hatasını engellemektir.

Klasik ResNet modelleri genellikle iki ya da üç katman atlanarak doğrusal olmayan ReLU aktivasyon fonksiyonu kullanılarak bağlantı kurar. (3.10) eşitliği ile tanımlanan ReLU (Rectified Linear Unit) fonksiyonu, rampa fonksiyonu olarak da bilinir. Bu fonksiyon modelin daha hazlı ve daha etkili öğrenmesini sağlamaktadır.

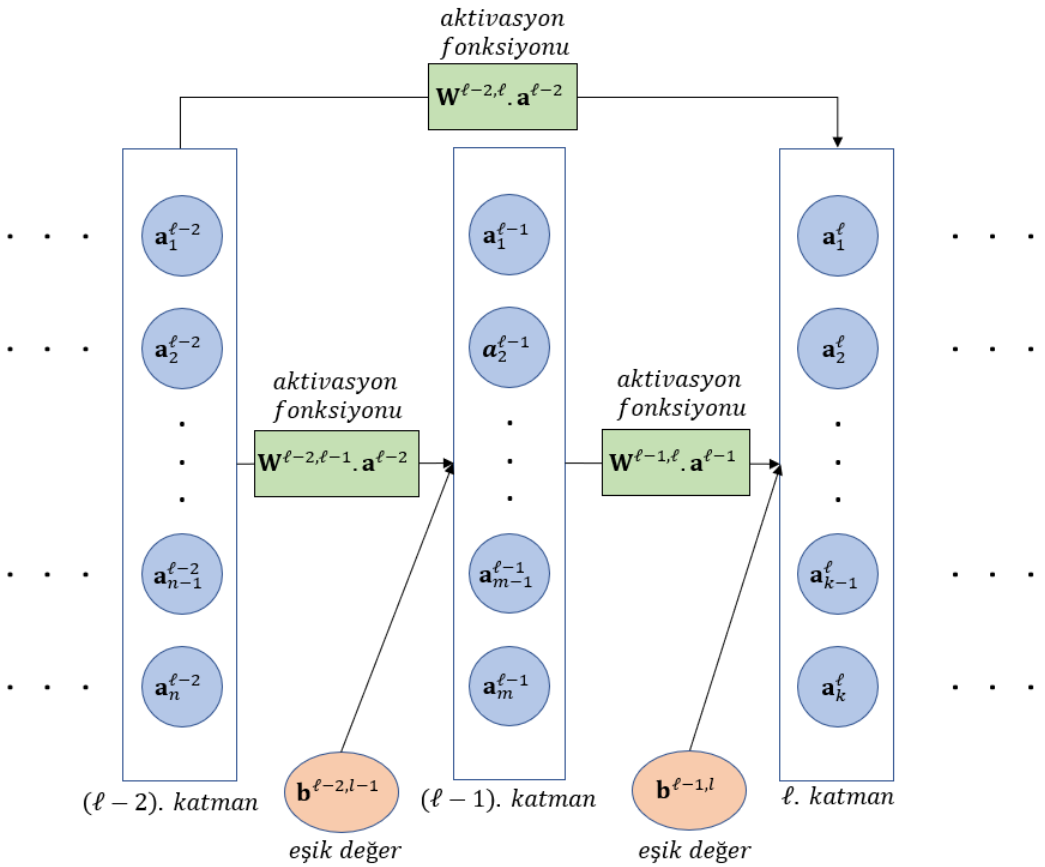
$$f(x) = x^{+} = \max(0, x) \tag{3.10}$$

ReLU aktivasyon fonksiyonunun grafiği Şekil 3.3'te verilmiştir.



Şekil 3.3. ReLU aktivasyon fonksiyonu.

Rezidü yapay sinir ağının yapısı ise Şekil 3.3'te verilmiştir.



Şekil 3.4. Artık (Rezidü) Yapay Sinir Ağının yapısı.

Rezidü ağının çıktıları (3.11) denklemi ile hesaplanır. (3.11) eşitliğinde $\mathbf{W}^{\ell-1,\ell}$, $(\ell-1)$. katman ile ℓ. katman arasındaki ağırlıkların matrisi; $\mathbf{W}^{\ell-2,\ell}$ ise $(\ell-2)$. katman ile ℓ. katman arasındaki ağırlıkların matrisi olmak üzere, ℓ. katmanın çıktıları a^{ℓ} ile tanımlanır.

$$\boldsymbol{a}^{\ell} = g(\boldsymbol{W}^{\ell-1,\ell}\boldsymbol{a}^{\ell} + \boldsymbol{b}^{\ell} + \boldsymbol{W}^{\ell-2,\ell}.\boldsymbol{a}^{\ell-2}) \qquad (3.11)$$

3.4. Örgü Uyarlamalı Doğrudan Arama ile Doğrusal Olmayan Optimizasyon (NOMAD)

Örgü uyarlamalı doğrudan arama algoritması (Mesh Adaptive Direct Search, MADS), doğrusal olmayan optimizasyon problemleri için geliştirilmiş türev içermeyen bir

optimizasyon algoritmasıdır (Audet, Dennis, 2006). Algoritma, mevcut çözüm etrafında arama yaparak daha iyi çözümler elde etmeyi amaçlar.

MADS algoritması, Genelleştirilmiş Desen Arama (GPS) algoritmasının geliştirilmiş halidir.

Tanım 3.4.1. $\mathbf{G} \in \mathbb{R}^{n\times n}$ tersinir matrisi ve $\mathbf{Z} \in \mathbb{Z}^{n\times p}$ matrisi verilsin öyle ki **Z** matrisinin sütunları $\mathbb{R}^n$ için bir pozitif taban ve $\mathbf{D} = \mathbf{GZ}$ olsun. **D** ile belirlenen örgü adım uzunluğu parametresi $\delta^k > 0$ ve merkezdeki mevcut çözüm $\mathbf{x^k}$ olmak üzere M^k örgü kümesi (3.12) eşitliği ile tanımlanır.

$$M^k = \{ \mathbf{x^k} + \delta^k D\mathbf{y} : \mathbf{x^k} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{N}^p, \delta^k > 0 \} \subseteq \mathbb{R}^n \qquad (3.12)$$

(3.10) eşitliğindeki δ^k değişkenine örgü adım uzunluğu parametresi denir.

GPS algoritmasında örgü adım uzunluğu seçim kümesini oluşturularak seçim adımının kontrolü için kullanılır.

$$P^k_{GPS} = \{ \mathbf{x^k} + \delta^k d : d \in \mathbb{D}^k_{GPS} \} \qquad (3.13)$$

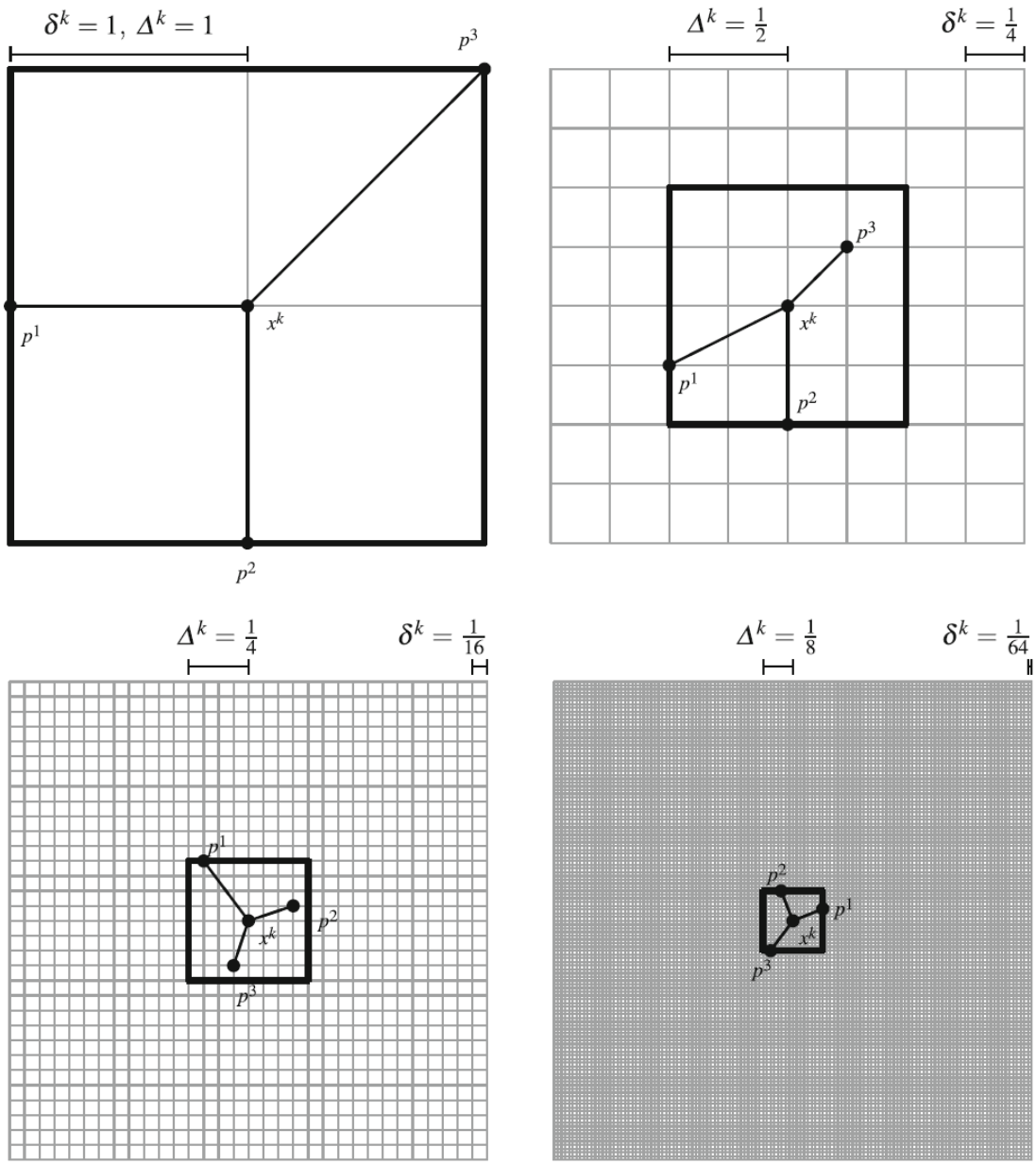
Burada $\mathbb{D}^k_{GPS}$, **D'**nin sütun matrisi $\mathbb{D}$'nin pozitif bir tabanıdır. MADS algoritmasındaki esas fikir, seçim adımında δ^k yerine yeni bir parametre olan çerçeve boyutu parametresi Δ^k oluşturmaktır. Çerçeve boyutu parametresi Δ^k, seçimin yapıldığı çerçeveyi tanımlar. her iterasyonda bu iki parametre $0 < \delta^k \leq \Delta^k$ koşulunu sağlar.

Tanım 3.4.2. $\mathbf{G} \in \mathbb{R}^{n\times n}$ tersinir matrisi ve $\mathbf{Z} \in \mathbb{Z}^{n\times p}$ matrisi verilsin öyle ki **Z** matrisinin sütunları $\mathbb{R}^n$ için bir pozitif taban ve $\mathbf{D} = \mathbf{GZ}$ olsun. Öyle bir $\delta^k > 0$ örgü adım uzunluğu parametresi seçilsin ki belirlenen Δ^k için $\Delta^k \leq \delta^k$ olsun. **D** ile belirlenen çerçeve genişliği Δ^k ve merkezdeki mevcut çözüm $\mathbf{x^k}$ olmak üzere F^k çerçeve kümesi (3.14) eşitliği ile tanımlanır.

$$F^k = \left\{ x \in M^k : |x - \mathbf{x^k}|_\infty \leq b\Delta^k \right\} \qquad (3.14)$$

(3.14) eşitliğinde $b = \max\{|\mathbf{d}'|_\infty : \mathbf{d}' \in \mathbf{D}\}$ ve Δ^k değişkenine çerçeve boyutu denir.

Şekil 3.4'te δ^k ve Δ^k başlangıç değerleri ile algoritma tarafından yenilenen değerler sonucunda yeni elde edilen çerçeve boyutları ve örgü adım uzunlukları gösterilmiştir (Audet, Hare, 2017).



Şekil 3.5. $\mathbb{R}^2$de farklı δ^k ve Δ^k değerleri ile elde edilen örgü örnekleri

Her iterasyonda elde edilen yeni noktalar mevcut noktalarla karşılaştırılıp daha iyi noktalar elde edilmişse o nokta doğrultusunda δ^k ve Δ^k güncellenerek rassal olarak yeni noktalar seçilip algoritma devam ettirilmektedir. δ^k ve Δ^k başlangıçta 1 olarak alınmıştır.

Tanım 3.4.3. $f: \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ amaç fonksiyonu ve kısıtlar kümesi $\Omega \in \mathbb{R}^n$ olmak üzere, $f_\Omega: \mathbb{R}^n \mapsto \mathbb{R} \cup \{\infty\}$ aşırı bariyer fonksiyonu (3.15) ile tanımlanır:

$$f_\Omega(x) := \begin{cases} f(x), & x \in \Omega \\ \infty, & x \notin \Omega \end{cases} \qquad (3.15)$$

Algoritma 1: MADS algoritması

---

f: $\mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$, başlangıç noktası $x^0 \epsilon \Omega$ ve $f_\Omega(x)$ aşırı bariyer fonksiyonu olmak üzere

0. Başlangıç Adımı:

$\Delta^0 \in (0, \infty)$	Δ: Çerçeve Boyutunun Başlangıç Değeri
$D = GZ$	D: Pozitif Taban Matrisi
$\tau \in (0,1), \tau \in Q$	τ: Örgü adım uzunluğu güncelleştirme parametresi
$\epsilon_{stop} \in [0, \infty)$	Durdurma kriteri
$k \leftarrow 0$	İterasyon sayacı

1. Parametre Değişimi Adımı:

Örgü Adım Uzunluğunu $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$ ile ayarla

2. Arama Adımı:

Eğer M^k nın sonlu alt kümesi S^k nın bazı t elemanları için $f_\Omega(t) < f_\Omega(x^k)$ ise $x^k + 1 \leftarrow t$ ve $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$ yap ve 4'e geç aksi halde 3'e geç

Seçim Adımı:

Bir pozitif taban kümesi $\mathbb{D}_\Delta^k$ öyle ki $P^k = \{x^k + \delta^k d : d \in \mathbb{D}_\Delta^k\}$, Δ^k genişliğindeki F^k nın bir alt kümesi olmak üzere, eğer $f(t) < f(x^k)$ olacak şekilde $t \in P^k$ varsa

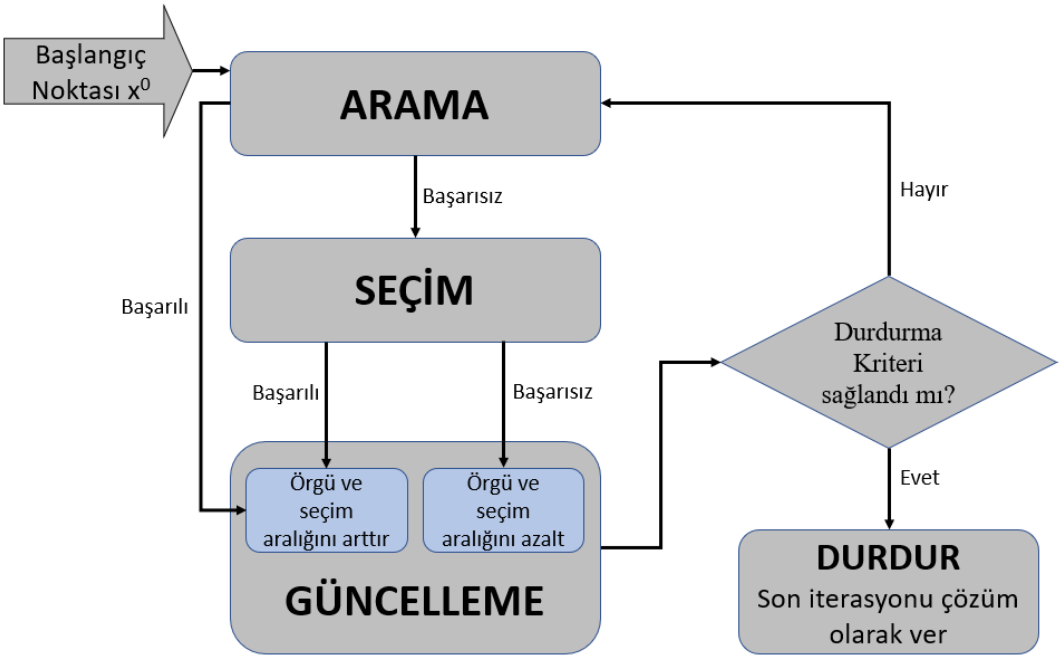
$x^{k+1} \leftarrow t$ ve $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$ yap

aksi halde $x^{k+1} \leftarrow x^k$ ve $\Delta^{k+1} \leftarrow \tau \Delta^k$ yap.

3. Bitirme Adımı:

Eğer $\Delta^{k+1} \geq \epsilon_{stop}$ ise $k \leftarrow k + 1$ yap ve 1'e geç aksi halde durdur

MADS algoritmasına ait akış diyagramı Şekil 3.5’ de verilmiştir (Amaioua vd., 2018).



Şekil 3.6. MADS algoritmasının akış diyagramı.

Tezin takip eden bölümünde üzerinde çakışan veya çakışmayan delikler bulunan bölgeler üzerinde tanımlı farklı sınır koşullarına sahip kısmi diferansiyel denklemlerin nümerik çözümlerinin elde edilmesine yönelik örnekler bulunmaktadır. Tüm örnekler için aynı topolojik yapıya sahip rezidü yapay sinir ağı kullanılmış ve gradyan hesaplama yükünden kurulmak için önerilen sinir ağı modelinin eğitimi NOMAD optimizasyon yöntemiyle gerçekleştirilmiştir.

4. BULGULAR

Bu bölümde tez çalışmasında ele alınan problemlerden ve bu problemlerin nümerik çözüm adımlarından bahsedilecektir. İki farklı eliptik kısmi diferansiyel denklem için iki farklı yaklaşım ile çözüm elde edilmiştir. İlk yaklaşımda sınır değerleri olarak çözüm uzayının dörtgensel olan dış sınırları (OBC) problemin çözümünde kullanılırken diğer yaklaşımda çözüm uzayının iç bölgesinde kalan dörtgensel olmayan sınırlardaki (IBC) çözümler de kullanılmıştır. Çözüm uzayı olarak iki farklı çözüm uzayı seçilmiştir. İlk olarak iç bölgesinde birbiriyle kesişen üç dairesel boşluğa sahip bir bölge (MC), daha sonra da iç bölgesinde kesişmeyen iki dairesel boşluğa sahip bölge (STC) çözüm uzayı olarak seçilmiştir.

Öncelikle Distmesh algoritması kullanılarak çözüm uzayında düğüm noktaları belirlenmektedir. [-100,100] aralığında belirlenen düğüm noktaları daha sonra [0,1] aralığına indirgenmektedir. Yaklaşık 8000 düğüm noktası elde edilmektedir.

Derin yapay sinir ağı olarak Rezidü Yapay Sinir ağı tercih edilmiştir. Yapay sinir ağı modeli 8 nörona sahip iki ara katmandan ve 2 nörona sahip girdi ve çıktı katmanından oluşmaktadır. Aktivasyon fonksiyonu olarak ReLU (Rectified Linear Unit) aktivasyon fonksiyonu kullanılmıştır. Ağın ağırlıkları [-10,10] aralığında değerler almaktadır. Bu katmanlar ile 45 adet ağırlık değeri oluşmuştur. Ağırlık sayısı kadar eşik değer oluşturulmuş ve bu eşik değerlerin başlangıç değeri sıfırdır.

Yapay sinir ağı oluşturulduktan sonra Deneme Çözümü ve Deneme Çözümüne ait kısmi türevler ile Maliyet fonksiyonu tanımlanır. Tez çalışmasında Julia programlama dili kullanıldığından Flux kütüphanesi kullanılarak maliyet fonksiyonu tanımlanmıştır. Optimizasyon algoritması olarak NOMAD kullanılmıştır.

Eğitim aşamasında ilk olarak veri seti 1500 noktalık alt gruplara ayrılarak ağ yığın öğrenme (batch learning) yaklaşımı ile ön eğitimden geçirilmiştir. Yığın öğrenme tamamlandıktan sonra ağ tüm veri seti kullanılarak tekrar eğitilmiştir. Maksimum uygunluk fonksiyonu hesaplama sayısı 20.000 olarak belirlenmiştir.

Her örnek için farklı çözüm uzayları ve farklı yaklaşımlar üzerinde 10 kez test gerçekleştirilmiştir.

Elde edilen ResNet çözümlerinin ve gerçek çözüm grafikleri Makie görselleştirme kütüphanesi kullanılarak oluşturulmuştur.

Tanım 4.1. (Poisson Denklemi) Poisson denklemi, fizikte sıkça kullanılan Laplace denkleminin genelleştirilmiş halidir ve Fransız matematikçi ve fizikçi Siméon Denis Poisson'un adını taşır. Teorik fizikte yaygın olarak kullanılan bir eliptik kısmi diferansiyel denklem olan Poisson denklemi, elektrik yükü veya kütle yoğunluğu dağılımının neden olduğu potansiyel alanı ifade etmek için kullanılır. Potansiyel alanın bilinmesi durumunda, elektrostatik veya yerçekimi (kuvvet) alanı hesaplanabilir.

Genel olarak 3 boyutlu kartezyen koordinat sisteminde (4.14) eşitliğinde verilen formda ifade edilir.

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z)=f(x,y,z) \tag{4.14}$$

Tanım 4.2. Çözüm uzayı olarak seçilen ilk bölge dörtgensel bölge içerisinden üç dairesel bölge çıkarılması ile elde edilmiştir. Çıkarılan bu bölgeler (4.15) ile verilmiştir.

$$\begin{aligned}\mathcal{C}_1&=\{(x,y)|(x-0.5)^2+(y-0.5)^2\leq 0.4^2\}\\ \mathcal{C}_2&=\{(x,y)|(x-0.3)^2+(y-0.7)^2\leq 0.25^2\}\\ \mathcal{C}_3&=\{(x,y)|(x-0.7)^2+(y-0.7)^2\leq 0.25^2\}\end{aligned} \tag{4.15}$$

Bu bölgelerin $[0,1]\times[0,1]\subset\mathbb{R}^2$'den çıkarılması ile elde edilen MC (Mickey Circles) bölgesi (4.16)'de tanımlanmıştır.

$$MC=\{(x,y)|(x,y)\in([0,1]\times[0,1]\subset\mathbb{R}^2-(\mathcal{C}_1\cup\mathcal{C}_2\cup\mathcal{C}_3))\} \tag{4.16}$$

Tanım 4.3. Tez çalışmasında ele alınan problemlerin çözüm uzayı olarak seçilen diğer bölge, dörtgensel bölge içerisinden kesişmeyen iki dairesel bölge çıkarılarak elde edilmiştir. Çıkarılan bölgeler (4.17) ile verilmiştir.

$$\begin{aligned}\mathcal{C}_4&=\{(x,y)|(x-0.3)^2+(y-0.7)^2\leq 0.45^2\}\\ \mathcal{C}_5&=\{(x,y)|(x-0.7)^2+(y-0.3)^2\leq 0.35^2\}\end{aligned} \tag{4.17}$$

Bu bölgelerin $[0,1]\times[0,1]\subset\mathbb{R}^2$'den çıkarılması ile elde edilen STC (Separated Two Circles) bölgesi (4.18)'de tanımlanmıştır.

$$STC=\{(x,y)|(x,y)\in([0,1]\times[0,1]\subset\mathbb{R}^2-(\mathcal{C}_4\cup\mathcal{C}_5))\} \tag{4.18}$$

Örnek 4.1. Tez çalışmasında ele alınan ilk örnek problem (4.19)'de verilmiştir.

$$\Delta u(x) = -\pi \sin(\pi xy)(x^2 + y^2) \tag{4.19}$$

Problemin sınır koşulları (4.20)'de verilmiştir.

$$u(0,y) = 0, u(1,y) = \pi\, sin(\pi y)$$
$$u(x,0) = 0, u(x,1) = \pi\, sin(\pi x) \tag{4.20}$$

Problemin gerçek çözümü ise (4.21)'de verilmiştir.

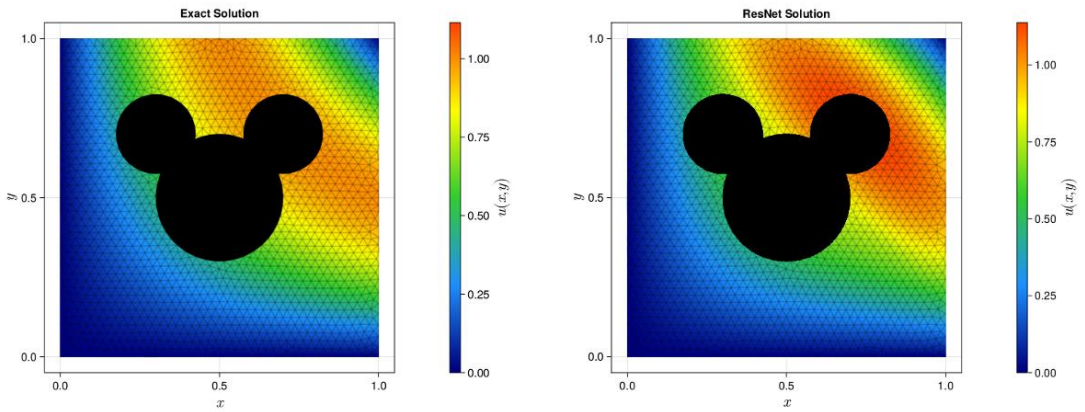
$$u(x,y) = \sin(\pi xy) \tag{4.21}$$

Sınır koşullarını sağlayan deneme çözümü (4.22)'daki gibi belirlenmiştir.

$$u_T(x,y) = x.sin(\pi y) + y\,.sin(\pi x) + x\,.y.(1-x).(1-y).ResNet(x,y;\boldsymbol{W}) \tag{4.22}$$

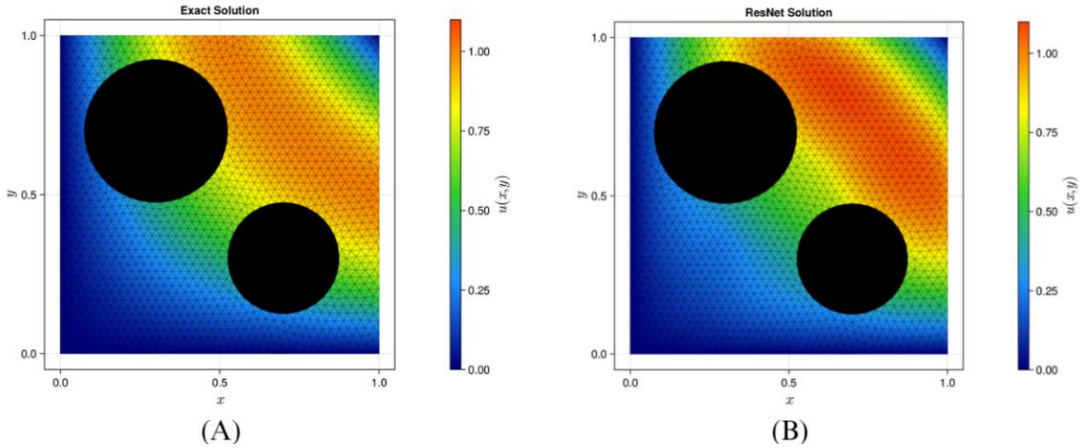
(4.9) eşitliğinde **W** sinir ağı modelinin ağırlık ve eşik değerlerini temsil eden matris olmak üzere $\boldsymbol{p} = (x,y) \in \mathbb{R}^2$ girdisi için $ResNet(x,y;\boldsymbol{W})$ ağın **p** noktasında ürettiği çıktıyı ifade etmektedir.

Örnek 4.1 için yalnızca dörtgensel bölge üzerindeki çözümlerin sınır değerleri olarak kullanıldığı (OBC) yöntemle testler gerçekleştirilmiştir. Bu testler ile elde edilen gerçek çözümlerinin ve ResNet çözümlerinin grafikleri iki farklı çözüm uzayı için Şekil 4.6 ve Şekil 4.7'de verilmiştir.



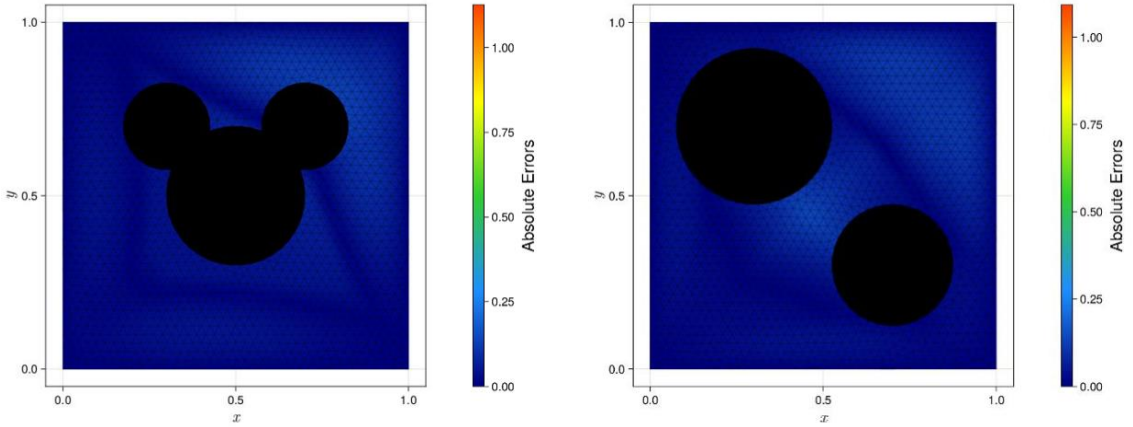
(A) (B)

Şekil 4.1. Örnek 4.1 için ilk çözüm uzayı (MC) üzerinde ve OBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.



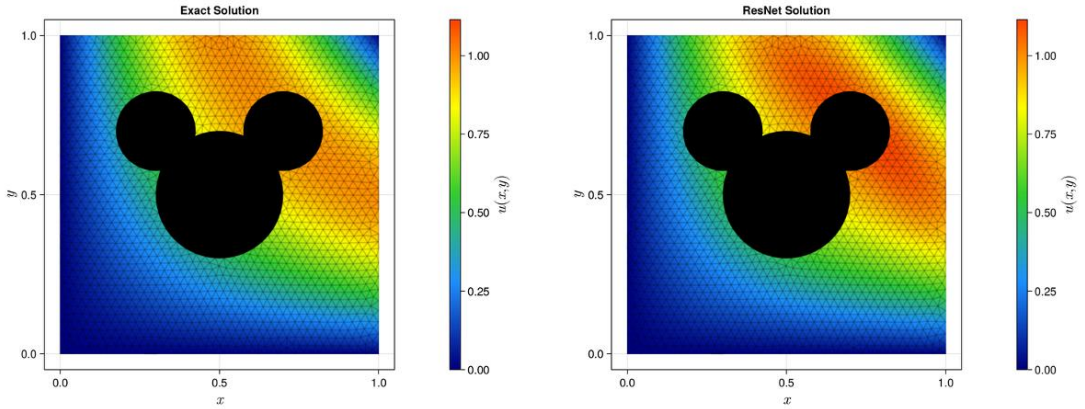
Şekil 4.2. Örnek 4.1 için ikinci çözüm uzayı (STC) üzerinde ve OBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.

Elde edilen mutlak hata grafikleri Şekil 4.3'de verilmiştir.



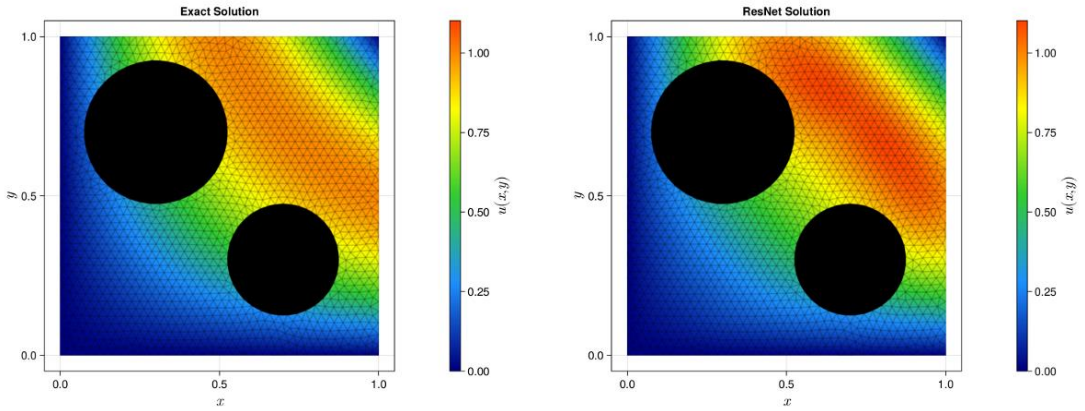
Şekil 4.3. Örnek 4.1 için her iki uzay üzerinde OBC ile elde edilen hata grafikleri.

Örnek 4.1 için dörtgensel dış bölge üzerindeki çözümlerin (OBC) yanı sıra iç bölgedeki dairesel sınırlardaki (IBC) çözümlerin de sınır değerleri olarak kullanıldığı yöntemle testler gerçekleştirilmiştir. Bu testler ile elde edilen gerçek çözümlerinin ve ResNet çözümlerinin grafikleri iki farklı çözüm uzayı için Şekil 4.4 ve Şekil 4.5 ile verilmiştir.



(A) (B)

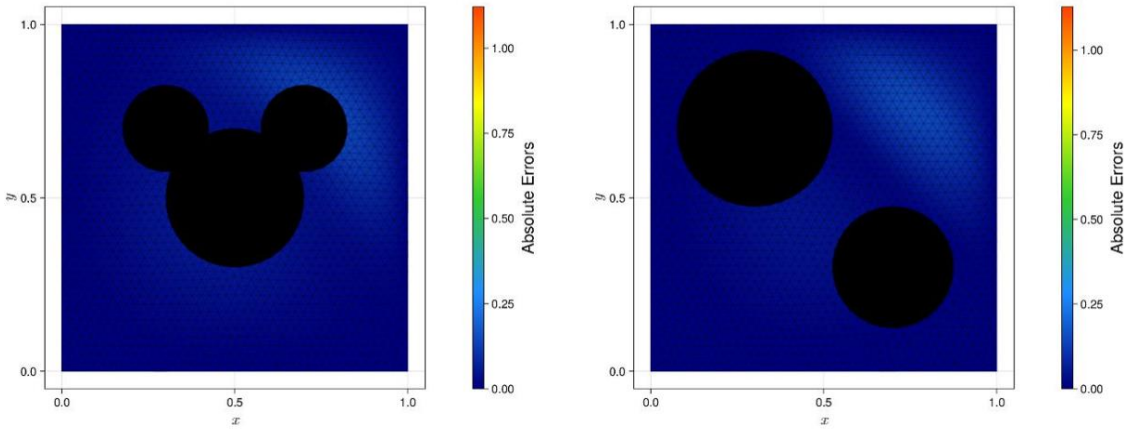
Şekil 4.4. Örnek 4.1 için ilk çözüm uzayı (MC) üzerinde, OBC ve IBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine ait grafikler.



(A) (B)

Şekil 4.5. Örnek 4.1 için ikinci çözüm uzayı (STC) üzerinde, OBC ve IBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.

Elde edilen mutlak hata grafikleri ise Şekil 4.6'da gösterilmektedir.



Şekil 4.6. Örnek 4.1 için her iki uzay üzerinde OBC ve IBC ile elde edilen hata grafikleri.

Farklı sınır koşulları ve çözüm uzayları ile Örnek 4.1 için dört farklı durum oluşmuştur. Bu dört durumun her biri için 10 kez test gerçekleştirilmiştir.

OBC sınır koşulları ve MC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $2{,}814 \times 10^{-3}$, standart sapması $8{,}692 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.1'de ve test kümesi sonuçları Çizelge 4.2'de verilmiştir.

Çizelge 4.1. Örnek 4.1 için OBC sınır koşulları ile MC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_T(x,y)$	$E = \lvert u(x,y) - u_T(x,y) \rvert$
1	0,134	0,049	0,016	0,029	$1{,}356 \times 10^{-2}$
2	0,118	0,024	0,035	0,057	$2{,}204 \times 10^{-2}$
3	0,000	0,100	0,000	0,000	0,000
4	0,016	0,121	0,016	0,029	$1{,}356 \times 10^{-2}$
5	0,000	0,100	0,008	0,017	$8{,}381 \times 10^{-3}$
6	0,000	0,063	0,016	0,029	$1{,}356 \times 10^{-2}$
7	0,082	0,100	0,000	0,000	0,000
8	0,095	0,123	0,008	0,017	$8{,}381 \times 10^{-3}$
⋮	⋮	⋮	⋮	⋮	⋮
8168	1,000	0,970	0,762	0,872	$1{,}096 \times 10^{-1}$
8169	0,987	0,952	0,727	0,834	$1{,}072 \times 10^{-1}$
8170	1,000	0,932	0,682	0,770	$8{,}822 \times 10^{-2}$
8171	0,987	0,952	0,727	0,834	$1{,}072 \times 10^{-1}$
8172	0,971	0,932	0,617	0,699	$8{,}242 \times 10^{-2}$
8173	0,987	0,952	0,727	0,834	$1{,}072 \times 10^{-1}$
8174	0,973	0,976	0,646	0,736	$8{,}980 \times 10^{-2}$

Çizelge 4.2. Örnek 4.1 için OBC sınır koşulları ile MC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,669	0,689	0,992	0,958	$3{,}416\times 10^{-2}$
2	0,802	0,467	0,924	0,954	$3{,}006 \times 10^{-2}$
3	0,954	0,672	0,903	0,970	$6{,}712 \times 10^{-2}$
4	0,437	0,023	0,032	0,048	$1{,}672\times 10^{-2}$
5	0,280	0,012	0,011	0,018	$7{,}599 \times 10^{-3}$
6	0,330	0,317	0,323	0,283	$4{,}055 \times 10^{-2}$
7	0,393	0,503	0,583	0,528	$5{,}406 \times 10^{-2}$
8	0,451	0,903	0,958	1,009	$5{,}128 \times 10^{-2}$
9	0,648	0,187	0,373	0,370	$2{,}769 \times 10^{-3}$
10	0,087	0,990	0,269	0,272	$3{,}180 \times 10^{-3}$
11	0,355	0,074	0,083	0,112	$2{,}873 \times 10^{-2}$
12	0,729	0,215	0,474	0,451	$2{,}285 \times 10^{-2}$
13	0,668	0,976	0,886	0,925	$3{,}874 \times 10^{-2}$
14	0,877	0,473	0,965	1,019	$5{,}479 \times 10^{-2}$
15	0,347	0,743	0,725	0,687	$3{,}794 \times 10^{-2}$
16	0,276	0,956	0,738	0,760	$2{,}197 \times 10^{-2}$
17	0,179	0,560	0,310	0,320	$9{,}911 \times 10^{-3}$
18	0,445	0,316	0,428	0,380	$4{,}788 \times 10^{-2}$
19	0,467	0,060	0,089	0,121	$3{,}166 \times 10^{-2}$
20	0,407	0,332	0,412	0,364	$4{,}745 \times 10^{-2}$
21	0,569	0,671	0,933	0,865	$6{,}785 \times 10^{-2}$

OBC sınır koşulları ve STC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $3{,}074 \times 10^{-3}$, standart sapması $1{,}312 \times 10^{-3}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.3'te ve test kümesi sonuçları Çizelge 4.4'te verilmiştir.

Çizelge 4.3. Örnek 4.1 için OBC sınır koşulları ile STC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,108	0,098	0,033	0,044	$1{,}059 \times 10^{-2}$
2	0,121	0,073	0,028	0,039	$1{,}130 \times 10^{-2}$
3	0,018	0,017	$9{,}949 \times 10^{-4}$	0,001	$5{,}918 \times 10^{-4}$
4	0,043	0,027	0,003	0,005	$2{,}077 \times 10^{-3}$
5	0,082	0,099	0,025	0,034	$8{,}572 \times 10^{-3}$
6	0,108	0,098	0,033	0,044	$1{,}059 \times 10^{-2}$
7	0,082	0,099	0,025	0,034	$8{,}572 \times 10^{-3}$
8	0,067	0,124	0,026	0,036	$1{,}029 \times 10^{-2}$
⋮	⋮	⋮	⋮	⋮	⋮
7664	0,988	0,912	0,302	0,311	$9{,}330 \times 10^{-3}$
7665	1,000	0,932	0,210	0,210	$1{,}110 \times 10^{-11}$
7666	1,000	0,971	0,090	0,090	$1{,}250 \times 10^{-13}$
7667	1,000	0,932	0,210	0,210	$1{,}110 \times 10^{-11}$
7668	0,972	0,932	0,289	0,306	$1{,}768 \times 10^{-2}$
7669	1,000	0,932	0,210	0,210	$1{,}110 \times 10^{-11}$
7670	0,988	0,952	0,181	0,187	$5{,}514 \times 10^{-3}$

Çizelge 4.4. Örnek 4.1 için OBC sınır koşulları ile STC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,062	0,808	0,157	0,173	$1{,}570 \times 10^{-2}$
2	0,310	0,811	0,711	0,666	$4{,}474 \times 10^{-2}$
3	0,749	0,590	0,983	1,020	$3{,}650 \times 10^{-2}$
4	0,905	0,461	0,967	1,002	$3{,}588 \times 10^{-2}$
5	0,771	0,237	0,544	0,533	$1{,}069 \times 10^{-2}$
6	0,851	0,843	0,774	0,872	$9{,}752 \times 10^{-2}$
7	0,495	0,915	0,989	1,063	$7{,}406 \times 10^{-2}$
8	0,797	0,264	0,615	0,618	$2{,}736 \times 10^{-3}$
9	0,117	0,097	0,036	0,047	$1{,}156 \times 10^{-2}$
10	0,246	0,888	0,635	0,621	$1{,}353 \times 10^{-2}$
11	0,318	0,159	0,159	0,185	$2{,}567 \times 10^{-2}$
12	0,429	0,334	0,435	0,365	$6{,}977 \times 10^{-2}$
13	0,480	0,115	0,173	0,210	$3{,}642 \times 10^{-2}$
14	0,946	0,001	0,003	0,003	$1{,}891 \times 10^{-4}$
15	0,329	0,790	0,729	0,671	$5{,}728 \times 10^{-2}$
16	0,507	0,816	0,964	0,992	$2{,}854 \times 10^{-2}$
17	0,519	0,280	0,442	0,410	$3{,}222 \times 10^{-2}$
18	0,276	0,110	0,095	0,123	$2{,}757 \times 10^{-2}$
19	0,895	0,763	0,838	0,929	$9{,}130 \times 10^{-2}$
20	0,173	0,675	0,360	0,384	$2{,}422 \times 10^{-2}$
21	0,924	0,165	0,461	0,469	$8{,}508 \times 10^{-3}$

OBC ve IBC sınır koşulları ile MC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}357 \times 10^{-3}$, standart sapması $1{,}587 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.5'te ve test kümesi sonuçları Çizelge 4.6'da verilmiştir.

Çizelge 4.5. Örnek 4.1 için OBC ve IBC sınır koşulları ile MC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,134	0,049	0,020	0,030	$1{,}020 \times 10^{-2}$
2	0,118	0,024	0,009	0,014	$4{,}953 \times 10^{-3}$
3	0,000	0,100	0,000	0,000	0,000
4	0,0161	0,121	0,006	0,010	$4{,}291 \times 10^{-3}$
5	0,000	0,100	0,000	0,000	0,000
6	0,000	0,063	0,000	0,000	0,000
7	0,082	0,100	0,026	0,041	$1{,}494 \times 10^{-2}$
8	0,095	0,123	0,037	0,056	$1{,}969 \times 10^{-2}$
⋮	⋮	⋮	⋮	⋮	⋮
8168	1,000	0,970	0,091	0,091	$1{,}250 \times 10^{-13}$
8169	0,987	0,952	0,186	0,185	$7{,}070 \times 10^{-4}$
8170	1,000	0,932	0,211	0,211	$1{,}110 \times 10^{-11}$
8171	0,987	0,952	0,186	0,185	$7{,}070 \times 10^{-4}$
8172	0,971	0,932	0,291	0,290	$1{,}408 \times 10^{-3}$
8173	0,987	0,952	0,186	0,185	$7{,}070 \times 10^{-4}$
8174	0,970	0,976	0,153	0,152	$8{,}103 \times 10^{-4}$

Çizelge 4.6. Örnek 4.1 için OBC ve IBC sınır koşulları ile MC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,197	0,221	0,137	0,169	$3{,}220 \times 10^{-2}$
2	0,652	0,662	0,977	1,046	$6{,}835 \times 10^{-2}$
3	0,334	0,791	0,739	0,757	$1{,}799 \times 10^{-2}$
4	0,302	0,455	0,418	0,468	$5{,}001 \times 10^{-2}$
5	0,870	0,317	0,762	0,772	$1{,}019 \times 10^{-2}$
6	0,342	0,642	0,637	0,675	$3{,}783 \times 10^{-2}$
7	0,681	0,951	0,893	0,916	$2{,}241 \times 10^{-2}$
8	0,741	0,611	0,989	1,082	$9{,}251 \times 10^{-2}$
9	0,484	0,321	0,469	0,520	$5{,}086 \times 10^{-2}$
10	0,996	0,329	0,858	0,859	$1{,}304 \times 10^{-3}$
11	0,274	0,564	0,467	0,510	$4{,}269 \times 10^{-2}$
12	0,078	0,934	0,228	0,230	$1{,}831 \times 10^{-3}$
13	0,138	0,133	0,058	0,083	$2{,}567 \times 10^{-2}$
14	0,086	0,758	0,205	0,219	$1{,}372 \times 10^{-2}$
15	0,606	0,698	0,971	1,029	$5{,}801 \times 10^{-2}$
16	0,042	0,225	0,030	0,048	$1{,}815 \times 10^{-2}$
17	0,342	0,194	0,207	0,237	$2{,}927 \times 10^{-2}$
18	0,309	0,181	0,175	0,202	$2{,}720 \times 10^{-2}$
19	0,548	0,202	0,342	0,371	$2{,}965 \times 10^{-2}$
20	0,061	0,287	0,055	0,083	$2{,}810 \times 10^{-2}$
21	0,358	0,941	0,871	0,896	$2{,}485 \times 10^{-2}$

OBC ve IBC sınır koşulları ile STC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}735 \times 10^{-3}$, standart sapması $2{,}73 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.7'de ve test kümesi sonuçları Çizelge 4.8'de verilmiştir.

Çizelge 4.7. Örnek 4.1 için OBC ve IBC sınır koşulları ile STC uzayı üzerinde elde edilen eğitim kümesi sonuçları

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,108	0,984	0,033	0,046	$1{,}339 \times 10^{-2}$
2	0,121	0,073	0,028	0,040	$1{,}202 \times 10^{-2}$
3	0,018	0,017	$9{,}949 \times 10^{-4}$	$1{,}821 \times 10^{-3}$	$8{,}266 \times 10^{-4}$
4	0,043	0,027	0,003	$6{,}391 \times 10^{-3}$	$2{,}671 \times 10^{-3}$
5	0,082	0,099	0,025	0,037	$1{,}215 \times 10^{-2}$
6	0,108	0,098	0,033	0,046	$1{,}339 \times 10^{-2}$
7	0,082	0,099	0,025	0,037	$1{,}215 \times 10^{-2}$
8	0,067	0,124	0,026	0,039	$1{,}287 \times 10^{-2}$
⋮	⋮	⋮	⋮	⋮	⋮
7664	0,988	0,912	0,302	0,302	$8{,}726 \times 10^{-4}$
7665	1,0	0,932	0,210	0,210	$1{,}11 \times 10^{-11}$
7666	1,0	0,971	0,090	0,090	$1{,}25 \times 10^{-13}$
7667	1,0	0,932	0,210	0,210	$1{,}11 \times 10^{-11}$
7668	0,972	0,932	0,289	0,290	$1{,}569 \times 10^{-3}$
7669	1,0	0,932	0,210	0,210	$1{,}11 \times 10^{-11}$
7670	0,988	0,952	0,181	0,182	$2{,}456 \times 10^{-4}$

Çizelge 4.8. Örnek 4.1 için OBC ve IBC sınır koşulları ile STC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,251	0,540	0,414	0,429	$1{,}455 \times 10^{-2}$
2	$5{,}569 \times 10^{-3}$	0,414	$7{,}257 \times 10^{-3}$	0,010	$3{,}251 \times 10^{-3}$
3	0,842	0,852	0,77	0,822	$4{,}913 \times 10^{-2}$
4	0,483	0,288	0,424	0,438	$1{,}366 \times 10^{-2}$
5	0,807	0,148	0,366	0,369	$2{,}599 \times 10^{-3}$
6	0,136	0,034	0,014	0,023	$8{,}545 \times 10^{-3}$
7	0,717	0,139	0,310	0,321	$1{,}159 \times 10^{-2}$
8	0,439	0,153	0,210	0,239	$2{,}862 \times 10^{-2}$
9	0,277	0,171	0,148	0,170	$2{,}192 \times 10^{-2}$
10	0,051	0,390	0,063	0,085	$2{,}213 \times 10^{-2}$
11	0,847	0,239	0,595	0,603	$7{,}554 \times 10^{-3}$
12	0,919	0,296	0,754	0,781	$2{,}620 \times 10^{-2}$
13	0,733	0,546	0,951	0,967	$1{,}588 \times 10^{-2}$
14	0,139	0,719	0,310	0,319	$9{,}127 \times 10^{-3}$
15	0,777	0,733	0,975	1,062	$8{,}613 \times 10^{-2}$
16	0,768	0,037	0,091	0,099	$7{,}938 \times 10^{-3}$
17	0,677	0,029	0,062	0,073	$1{,}014 \times 10^{-2}$
18	0,384	0,719	0,763	0,756	$7{,}255 \times 10^{-3}$
19	0,688	0,431	0,803	0,794	$9{,}594 \times 10^{-3}$
20	0,347	0,276	0,297	0,311	$1{,}421 \times 10^{-2}$
21	0,979	0,135	0,403	0,407	$3{,}763 \times 10^{-2}$

Örnek 4.2. Tez çalışmasında ele alınan ikinci örnek problem (4.23)'da verilmiştir.

$$\Delta u(x) = -2\pi^2 \sin(\pi x) \cdot \cos(\pi y) \tag{4.23}$$

Problemin sınır koşulları (4.24) eşitlikleriyle verilmiştir.

$$u(0,y) = 0, u(1,y) = 0$$
$$u(x,0) = sin(\pi x), u(x,1) = -sin(\pi x) \tag{4.24}$$

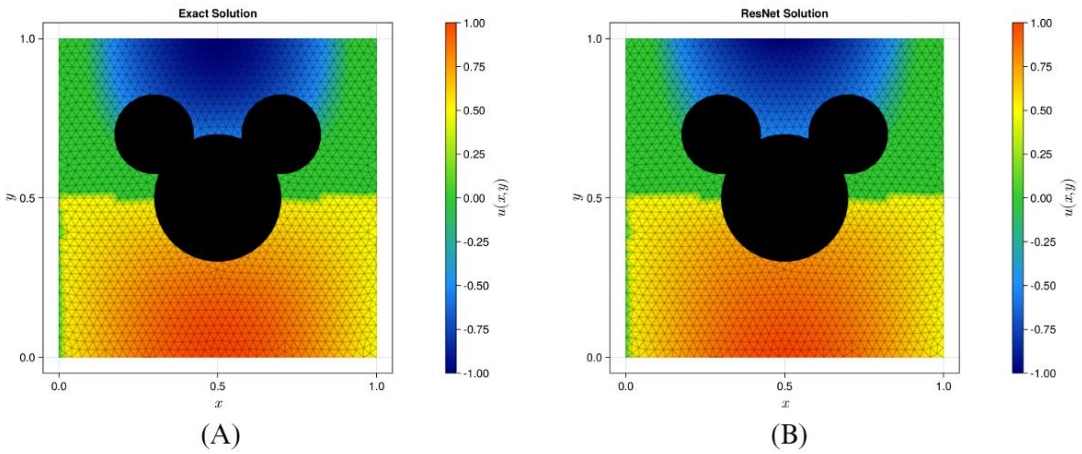
Problemin gerçek çözümü ise (4.25) eşitliği ile verilmiştir.

$$u(x,y) = sin(\pi x) \cdot cos(\pi y) \tag{4.25}$$

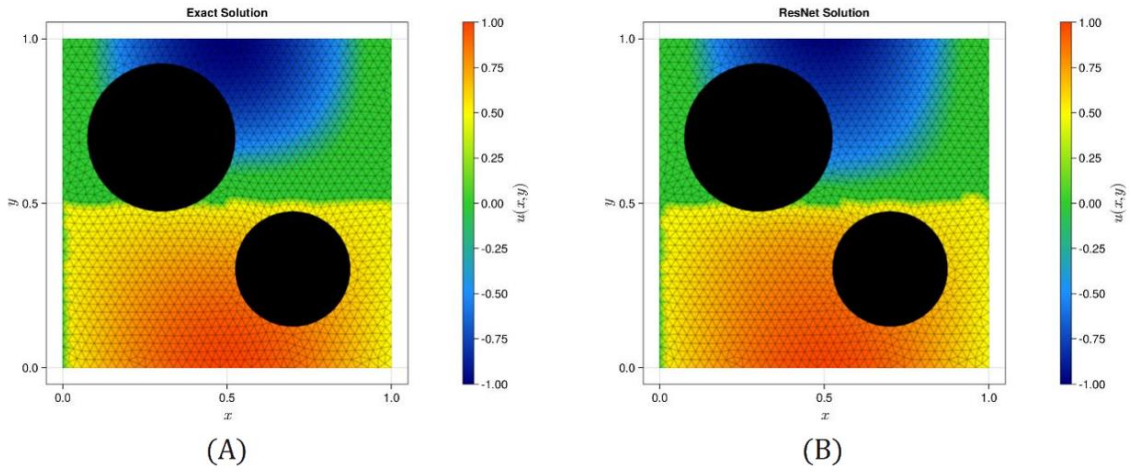
Sınır koşullarını sağlayan deneme çözümü (4.26) eşitliğindeki gibi belirlenmiştir.

$$u_T(x,y) = (1-y).\sin(\pi x) - y.\sin(\pi x) + x.y.(1-x).(1-y).\mathrm{ResNet}(x,y;\mathbf{W}) \tag{4.26}$$

Örnek 4.2 için yalnızca dörtgensel bölge üzerindeki çözümlerin sınır değerleri olarak kullanıldığı (OBC) yöntemle testler gerçekleştirilmiştir. Bu testler ile elde edilen gerçek çözümlerinin ve ResNet çözümlerinin grafikleri iki farklı çözüm uzayı için Şekil 4.7'de ve Şekil 4.8'de verilmiştir.

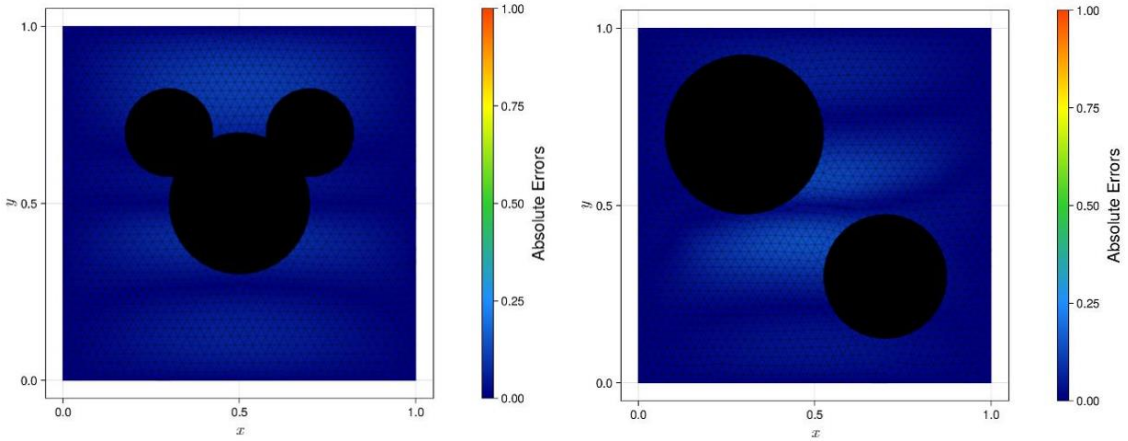


Şekil 4.7. Örnek 4.2 için ilk çözüm uzayı (MC) üzerinde ve OBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.



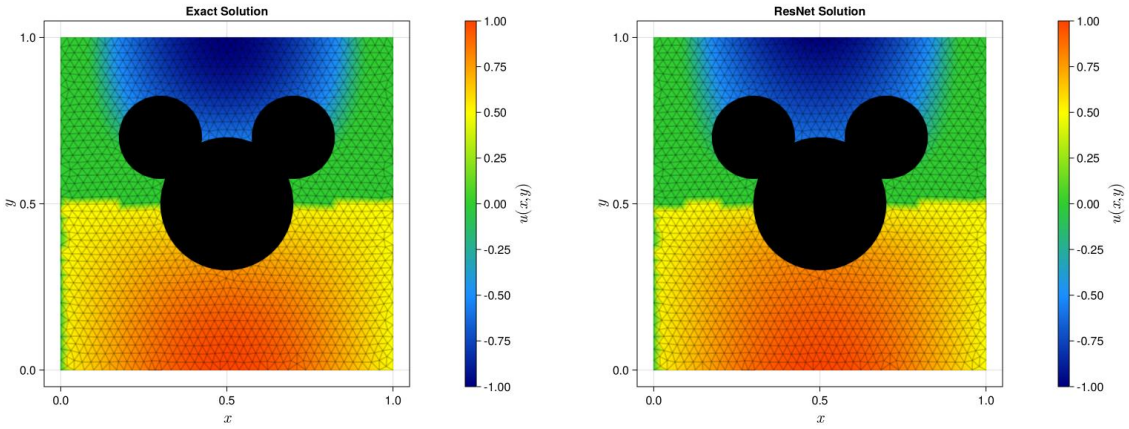
Şekil 4.8. Örnek 4.2 için ikinci çözüm uzayı (STC) üzerinde ve OBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.

Elde edilen mutlak hata grafikleri Şekil 4.9'da verilmiştir.

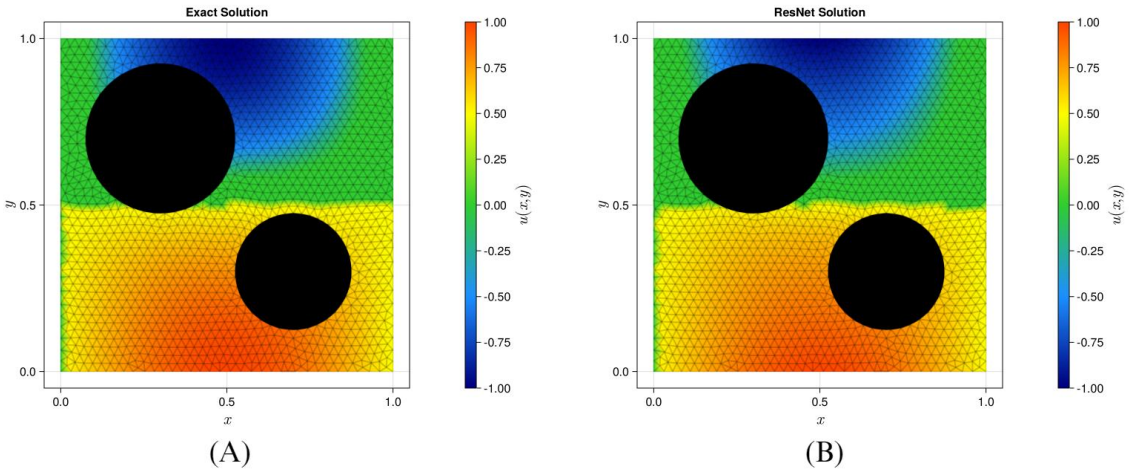


Şekil 4.9. Örnek 4.2 için her iki uzay üzerinde OBC ile elde edilen hata grafikleri.

Örnek 4.2 için dörtgensel dış bölge üzerindeki çözümlerin (OBC) yanı sıra iç bölgedeki dairesel sınırlardaki (IBC) çözümlerin de sınır değerleri olarak kullanıldığı yöntemle testler gerçekleştirilmiştir. Bu testler ile elde edilen gerçek çözümlerinin ve ResNet çözümlerinin grafikleri iki farklı çözüm uzayı için Şekil 4.10'da ve Şekil 4.11'de verilmiştir.

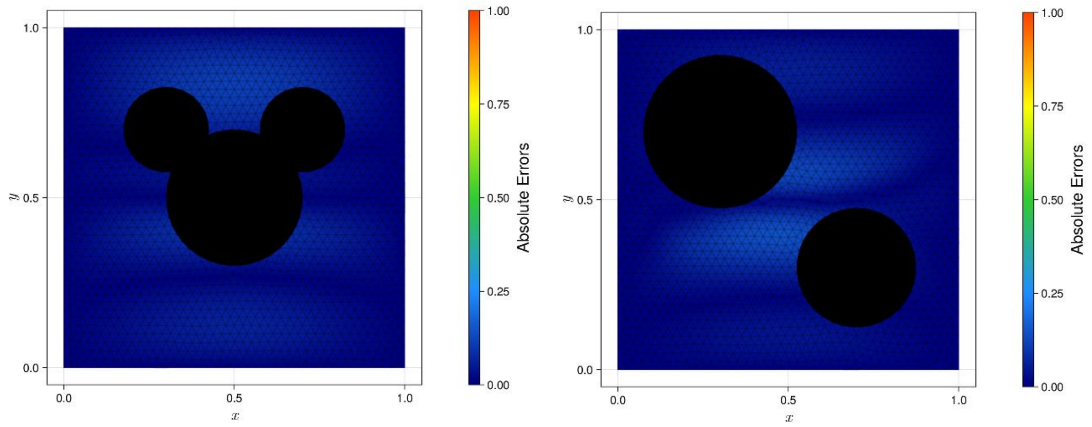


Şekil 4.10. Örnek 4.2 için ilk çözüm uzayı (MC) üzerinde, OBC ve IBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.



Şekil 4.11. Örnek 4.2 için ikinci çözüm uzayı (STC) üzerinde, OBC ve IBC ile elde edilen gerçek çözüm (A) ve ResNet çözümlerine (B) ait grafikler.

Elde edilen mutlak hata grafikleri Şekil 4.12'de verilmiştir.



Şekil 4.12. Örnek 4.2 için her iki uzay üzerinde OBC ve IBC ile elde edilen hata grafikleri.

Örnek 4.2 için de yine hem MC uzayı hem de STC uzayı üzerinde OBC ve IBC sınır koşullarıyla elde edilen 4 farklı durum için 10'ar test gerçekleştirilmiştir.

OBC sınır koşulları ve MC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}457 \times 10^{-3}$, standart sapması $2{,}767 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.9'da ve test kümesi sonuçları Çizelge 4.10'da verilmiştir.

Çizelge 4.9. Örnek 3.2 için OBC sınır koşulları ile MC uzayı üzerinde elde edilen eğitim kümesi sonuçları

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,134	0,049	0,404	0,395	$8{,}156 \times 10^{-3}$
2	0,118	0,024	0,363	0,359	$4{,}279 \times 10^{-3}$
3	0,000	0,100	0,000	0,000	0,000
4	0,0161	0,121	0,047	0,046	$2{,}159 \times 10^{-4}$
5	0,000	0,100	0,000	0,000	0,000
6	0,000	0,063	0,000	0,000	0,000
7	0,082	0,100	0,244	0,239	$4{,}668 \times 10^{-3}$
8	0,095	0,123	0,272	0,268	$4{,}263 \times 10^{-3}$
⋮	⋮	⋮	⋮	⋮	⋮
8168	1,000	0,970	$-1{,}22 \times 10^{-12}$	$-1{,}15 \times 10^{-11}$	$6{,}630 \times 10^{-13}$
8169	0,987	0,952	-0,039	-0,037	$1{,}260 \times 10^{-3}$
8170	1,000	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}380 \times 10^{-12}$
8171	0,987	0,952	-0,039	-0,037	$1{,}260 \times 10^{-3}$
8172	0,971	0,932	-0,086	-0,083	$3{,}658 \times 10^{-3}$
8173	0,987	0,952	-0,039	-0,037	$1{,}261 \times 10^{-3}$
8174	0,970	0,976	-0,082	-0,080	$1{,}495 \times 10^{-3}$

Çizelge 4.10. Örnek 3.2 için OBC sınır koşulları ile MC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,711	0,938	-0,772	-0,733	$3{,}891 \times 10^{-2}$
2	0,78302	0,078	0,611	0,592	$1{,}827 \times 10^{-2}$
3	0,148	0,592	-0,129	-0,153	$2{,}443 \times 10^{-2}$
4	0,391	0,247	0,67069	0,695	$2{,}488 \times 10^{-2}$
5	0,230	0,763	-0,487	-0,465	$2{,}243 \times 10^{-2}$
6	0,114	0,202	0,282	0,289	$7{,}264 \times 10^{-3}$
7	0,297	0,239	0,587	0,608	$2{,}026 \times 10^{-2}$
8	0,529	0,290	0,609	0,669	$5{,}998 \times 10^{-2}$
9	0,922	0,546	-0,034	-0,036	$1{,}150 \times 10^{-3}$
10	0,435	0,483	0,051	0,072	$2{,}079 \times 10^{-2}$
11	0,671	0,913	-0,827	-0,773	$5{,}320 \times 10^{-2}$
12	0,230	0,206	0,527	0,533	$5{,}544 \times 10^{-3}$
13	0,357	0,120	0,837	0,806	$3{,}157 \times 10^{-2}$
14	0,591	0,309	0,541	0,615	$7{,}468 \times 10^{-2}$
15	0,759	0,995	-0,685	-0,682	$3{,}095 \times 10^{-3}$
16	0,299	0,924	-0,786	-0,740	$4{,}575 \times 10^{-2}$
17	0,113	0,036	0,346	0,340	$5{,}475 \times 10^{-3}$
18	$7{,}5617 \times 10^{-3}$	0,204	0,019	0,020	$9{,}950 \times 10^{-4}$
19	0,077	0,679	-0,128	-0,134	$5{,}781 \times 10^{-3}$
20	0,383	0,180	0,788	0,772	$1{,}621 \times 10^{-2}$
21	0,452	0,243	0,714	0,734	$2{,}072 \times 10^{-2}$

OBC sınır koşulları ve STC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}517 \times 10^{-3}$, standart sapması $1{,}777 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.11'de ve test kümesi sonuçları Çizelge 4.12'de verilmiştir.

Çizelge 4.11. Örnek 4.2 için OBC sınır koşulları ile STC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,108	0,984	0,317	0,302	$1{,}475 \times 10^{-2}$
2	0,121	0,073	0,362	0,347	$1{,}514 \times 10^{-2}$
3	0,018	0,017	0,057	0,056	$6{,}526 \times 10^{-4}$
4	0,043	0,027	0,134	0,132	$2{,}401 \times 10^{-3}$
5	0,082	0,099	0,242	0,232	$1{,}075 \times 10^{-2}$
6	0,108	0,098	0,317	0,302	$1{,}475 \times 10^{-2}$
7	0,082	0,099	0,242	0,232	$1{,}075 \times 10^{-2}$
8	0,067	0,124	0,195	0,187	$8{,}487 \times 10^{-3}$
⋮	⋮	⋮	⋮	⋮	⋮
7664	0,988	0,912	-0,034	-0,032	$1{,}926 \times 10^{-3}$
7665	1,000	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7666	1,000	0,971	$-1{,}22 \times 10^{-11}$	$-1{,}15 \times 10^{-11}$	$6{,}57 \times 10^{-13}$
7667	1,000	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7668	0,972	0,932	-0,085	-0,081	$4{,}177 \times 10^{-3}$
7669	1,0	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7670	0,988	0,952	-0,034	-0,033	$1{,}290 \times 10^{-3}$

Çizelge 4.12. Örnek 4.2 için OBC sınır koşulları ile STC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,929	0,451	0,033	0,040	$6{,}721 \times 10^{-3}$
2	0,024	0,369	0,030	0,038	$8{,}488 \times 10^{-3}$
3	0,589	0,357	0,416	0,472	$5{,}608 \times 10^{-2}$
4	0,580	0,781	-0,749	-0,682	$6{,}645 \times 10^{-2}$
5	0,029	0,407	0,026	0,036	$1{,}008 \times 10^{-2}$
6	0,369	0,923	-0,890	-0,830	$5{,}948 \times 10^{-2}$
7	0,352	0,840	-0,785	-0,711	$7{,}412 \times 10^{-2}$
8	0,376	0,976	-0,923	-0,900	$2{,}269 \times 10^{-2}$
9	0,668	0,092	0,827	0,779	$4{,}732 \times 10^{-2}$
10	0,138	0,044	0,416	0,403	$1{,}269 \times 10^{-2}$
11	0,290	0,909	-0,758	-0,703	$5{,}511 \times 10^{-2}$
12	0,320	0,280	0,537	0,547	$9{,}568 \times 10^{-3}$
13	0,114	0,071	0,343	0,329	$1{,}396 \times 10^{-2}$
14	0,760	0,557	-0,123	-0,165	$4{,}234 \times 10^{-2}$
15	0,134	0,169	0,353	0,337	$1{,}526 \times 10^{-2}$
16	0,524	0,851	-0,890	-0,805	$8{,}496 \times 10^{-2}$
17	0,089	0,083	0,267	0,256	$1{,}129 \times 10^{-2}$
18	0,497	0,664	-0,493	-0,500	$7{,}766 \times 10^{-3}$
19	0,112	0,168	0,298	0,286	$1{,}204 \times 10^{-2}$
20	0,341	0,681	-0,474	-0,469	$5{,}427 \times 10^{-3}$
21	0,780	0,475	0,049	0,056	$7{,}284 \times 10^{-3}$

OBC ve IBC sınır koşulları ve MC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}314 \times 10^{-3}$, standart sapması $1{,}927 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.13'te ve test kümesi sonuçları Çizelge 4.14'te verilmiştir.

Çizelge 4.13. Örnek 4.2 için OBC ve IBC sınır koşulları ile MC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,134	0,049	0,404	0,396	$7{,}532 \times 10^{-3}$
2	0,118	0,024	0,363	0,359	$3{,}988 \times 10^{-3}$
3	0,0	0,100	0,0	0,0	0,0
4	0,016	0,121	0,047	0,047	$7{,}19 \times 10^{-4}$
5	0,0	0,100	0,0	0,0	0,0
6	0,0	0,063	0,0	0,0	0,0
7	0,082	0,100	0,244	0,240	$3{,}925 \times 10^{-3}$
8	0,095	0,123	0,272	0,269	$3{,}285 \times 10^{-3}$
⋮	⋮	⋮	⋮	⋮	⋮
8168	1,0	0,970	$-1{,}22 \times 10^{-12}$	$-1{,}15 \times 10^{-11}$	$6{,}63 \times 10^{-13}$
8169	0,987	0,952	-0,039	-0,038	$1{,}187 \times 10^{-3}$
8170	1,0	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
8171	0,987	0,952	-0,039	-0,038	$1{,}187 \times 10^{-3}$
8172	0,971	0,932	-0,086	-0,08	$3{,}437 \times 10^{-3}$
8173	0,987	0,952	-0,039	-0,038	$1{,}187 \times 10^{-3}$
8174	0,97	0,976	-0,082	-0,080	$1{,}419 \times 10^{-3}$

Çizelge 4.14. Örnek 4.2 için OBC ve IBC sınır koşulları ile MC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,125	0,896	-0,363	-0,343	$2{,}065 \times 10^{-2}$
2	0,067	0,761	-0,154	-0,152	$1{,}716 \times 10^{-3}$
3	0,168	0,065	0,493	0,481	$1{,}149 \times 10^{-2}$
4	0,924	0,323	0,123	0,146	$2{,}254 \times 10^{-2}$
5	0,039	0,011	0,125	0,124	$5{,}418 \times 10^{-4}$
6	0,698	0,032	0,807	0,792	$1{,}448 \times 10^{-2}$
7	0,825	$8{,}687 \times 10^{-3}$	0,521947	0,519438	$2{,}508 \times 10^{-3}$
8	0,032	0,792	-0,080	-0,078	$2{,}048 \times 10^{-3}$
9	0,014	0,204	0,037	0,039	$1{,}698 \times 10^{-3}$
10	0,893	0,340	0,157	0,188	$3{,}128 \times 10^{-2}$
11	0,131	0,826	-0,342	-0,323	$1{,}905 \times 10^{-2}$
12	0,913	0,489	$8{,}594 \times 10^{-3}$	$5{,}884 \times 10^{-3}$	$2{,}709 \times 10^{-3}$
13	$1{,}934 \times 10^{-3}$	0,644132	$-2{,}65868 \times 10^{-3}$	$-2{,}743 \times 10^{-3}$	$8{,}48 \times 10^{-5}$
14	0,646	0,732	-0,597	-0,585	$1{,}22 \times 10^{-2}$
15	0,351	0,771	-0,673	-0,637	$3{,}550 \times 10^{-2}$
16	0,834	0,612	-0,171	-0,191	$2{,}008 \times 10^{-2}$
17	0,969	0,888	-0,090	-0,085	$4{,}445 \times 10^{-3}$
18	0,929	0,975	-0,217	-0,213	$4{,}140 \times 10^{-3}$
19	0,672	0,752	-0,610	-0,587	$2{,}282 \times 10^{-2}$
20	0,691	0,757	-0,595	-0,572	$2{,}370 \times 10^{-2}$
21	0,042	0,759	-0,097	-0,097	$4{,}795 \times 10^{-4}$

OBC ve IBC sınır koşulları ve STC çözüm uzayı üzerinde gerçekleştirilen testlerde elde edilen karesel hataların ortalaması $1{,}618 \times 10^{-3}$, standart sapması $3{,}469 \times 10^{-4}$ olmuştur. Bu testlere ait eğitim kümesi sonuçları Çizelge 4.15'te ve test kümesi sonuçları Çizelge 4.16'da verilmiştir.

Çizelge 4.15. Örnek 4.2 için OBC ve IBC sınır koşulları ile STC uzayı üzerinde elde edilen eğitim kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,108	0,984	0,317	0,301	$1{,}546 \times 10^{-2}$
2	0,121	0,073	0,362	0,347	$1{,}574 \times 10^{-2}$
3	0,018	0,017	0,057	0,056	$6{,}776 \times 10^{-4}$
4	0,043	0,027	0,134	0,132	$2{,}491 \times 10^{-3}$
5	0,082	0,099	0,242	0,231	$1{,}132 \times 10^{-2}$
6	0,108	0,098	0,317	0,301	$1{,}546 \times 10^{-2}$
7	0,082	0,099	0,242	0,231	$1{,}132 \times 10^{-2}$
8	0,067	0,124	0,195	0,186	$9{,}084 \times 10^{-2}$
⋮	⋮	⋮	⋮	⋮	⋮
7664	0,988	0,912	-0,034	-0,032	$2{,}006 \times 10^{-2}$
7665	1,0	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7666	1,0	0,971	$-1{,}22 \times 10^{-11}$	$-1{,}15 \times 10^{-11}$	$6{,}57 \times 10^{-13}$
7667	1,0	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7668	0,972	0,932	$-8{,}554 \times 10^{-2}$	$-8{,}121 \times 10^{-2}$	$4{,}331 \times 10^{-3}$
7669	1,0	0,932	$-1{,}20 \times 10^{-11}$	$-1{,}06 \times 10^{-11}$	$1{,}38 \times 10^{-12}$
7670	0,988	0,952	-0,034	-0,033	$1{,}335 \times 10^{-3}$

Çizelge 4.16. Örnek 4.2 için OBC ve IBC sınır koşulları ile STC uzayı üzerinde elde edilen test kümesi sonuçları.

No	x	y	$u(x,y)$	$u_{\mathrm{T}}(x,y)$	$E = \lvert u(x,y) - u_{\mathrm{T}}(x,y) \rvert$
1	0,519	0,019	0,996	0,978	$0{,}01{,}74865 \times 10^{-2}$
2	0,406	0,694	-0,548	-0,532	$1{,}595 \times 10^{-2}$
3	0,708	0,930	-0,775	-0,726	$4{,}834 \times 10^{-2}$
4	0,635	0,671	-0,468	-0,471	$3{,}262 \times 10^{-3}$
5	0,968	0,451	0,015	0,019	$4{,}679 \times 10^{-3}$
6	0,858	0,821	-0,363	-0,333	$3{,}013 \times 10^{-2}$
7	0,979	0,337	0,032	0,038	$6{,}018 \times 10^{-3}$
8	0,236	0,828	-0,581	-0,527	$5{,}389 \times 10^{-2}$
9	$3{,}8990 \times 10^{-3}$	0,796	$-9{,}832 \times 10^{-3}$	$-9{,}223 \times 10^{-3}$	$6{,}091 \times 10^{-4}$
10	0,710	0,672	-0,408	-0,413	$5{,}793 \times 10^{-3}$
11	0,249	0,809	-0,584	-0,530	$5{,}390 \times 10^{-2}$
12	0,046	0,585	-0,038	-0,041	$2{,}884 \times 10^{-3}$
13	0,527	0,294	0,599	0,611	$1{,}259 \times 10^{-2}$
14	0,187	0,960	-0,551	-0,530	$2{,}099 \times 10^{-2}$
15	0,289	0,872	-0,083	-0,077	$5{,}987 \times 10^{-3}$
16	0,020	0,949	-0,064	-0,061	$2{,}634 \times 10^{-3}$
17	0,028	0,397	0,028	0,036	$8{,}029 \times 10^{-3}$
18	0,212	0,831	-0,534	-0,485	$4{,}859 \times 10^{-2}$
19	0,251	0,818	-0,597	-0,542	$5{,}573 \times 10^{-2}$
20	$1{,}948 \times 10^{-3}$	0,033	$6{,}0897 \times 10^{-3}$	$5{,}96722 \times 10^{-3}$	$1{,}224 \times 10^{-4}$
21	0,404	0,954	-0,945	-0,902	$4{,}326 \times 10^{-2}$

Bu bölümde verilen örnek problemlerin Bölüm 2'de önerilen yöntemle çözülebilmesi için Julia programlama dilinde kodlamalar gerçekleştirilmiştir. Gerçekleştirilen Julia implementasyonları Aydın Adnan Menderes Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü (ADU-BAP) tarafından desteklenen FEF-22026 numaralı güdümlü proje kapsamında kurulan Fizik-Matematik Yüksek Başarımlı Hesaplama Sisteminin (High Processing Computing System, HPC) sunucuları üzerinde basit paralel programlama mantığıyla test edilmiştir. Örnek 3.1 için yazılan örnek Julia kodu Ek 1'de sunulmuştur. Yazılan kodlar 64 bit Rocky Linux 8.7 işletim sisteminde, Julia 1.8.5 sürümü kullanılarak test edilmiştir. Bu kodlar, Slurm iş yükü yöneticisi uygulaması üzerinden Ek 2'de örneği sunulan toplu komut betiği aracılığıyla sunucuya iş paketi olarak gönderilmiştir. Böylelikle aynı kod eş zamanlı olarak farklı parametre değerleri ile 10 kez çalıştırılmıştır. Kodlarda makine öğrenmesi yaklaşımları için Julia'da yazılmış açık kaynaklı bir makine öğrenimi yazılım kütüphanesi ve ekosistemi olan Flux'dan faydalanılmıştır (Innes, 2018; Innes, Saba, Fischer vd., 2018).

Tezin takip eden bölümünde elde edilen bulgulardan çıkarsaması yapılan sonuçlara ve tez çalışması süresinde karşılaşılan zorluklara değinilmiştir. Bu bağlamda bu tezin devamı niteliğinde olacak şekilde ileride yapılabilecek veya yapılması planlanan çalışma önerilerine de yine bir sonraki bölümde değinilmiştir.

5. TARTIŞMA

Bu tezde birçok gerçek dünya probleminin matematiksel modellenmesinde etkin bir araç olarak kullanılan kısmi diferansiyel denklemlerin, farklı bölgelerde derin yapay sinir ağı kullanılarak nümerik çözümleri elde edilmiştir. Ayrıca bu çalışmada türev tabanlı olmayan Nomad optimizasyon algoritması kullanılmıştır. Böylelikle sinir ağı modelinin bilinmeyen ağırlık ve eşik değerlerinin optimal değerlerinin belirlenmesi için iş yükü maliyetini fazlaca arttıran gradyan hesaplamalarından kaçınılmış olur.

Genel olarak bakıldığında tezde, iki farklı eliptik kısmi diferansiyel denklem için farklı yaklaşımlarla çözümler elde edildi. İlk yaklaşımda, çözüm uzayının dörtgensel olan dış sınırları (OBC) kullanılarak problemin çözümü gerçekleştirildi. Diğer yaklaşımda ise çözüm uzayının iç bölgesinde kalan dörtgensel olmayan sınırlar (IBC) kullanıldı. İki farklı çözüm uzayı seçildi: birincisinde, iç bölgesinde birbiriyle kesişen üç dairesel boşluğa sahip bir bölge (MC), ikincisinde ise iç bölgesinde kesişmeyen iki dairesel boşluğa sahip bir bölge (STC) belirlendi.

Çözüm uzayı belirlendikten sonra Distmesh algoritması kullanılarak düğüm noktaları belirlendi. Daha sonra, bu düğüm noktaları [0,1] aralığına indirgendi ve yaklaşık 8000 düğüm noktası elde edildi. Rezidü Yapay Sinir Ağı, derin yapay sinir ağı olarak kullanıldı. Yapay sinir ağı oluşturulduktan sonra maliyet fonksiyonu tanımlandı ve Flux kütüphanesi kullanılarak Julia programlama diliyle uygulandı. Optimizasyon algoritması olarak NOMAD kullanıldı.

Eğitim aşamasında, veri seti 1500 noktalık alt gruplara ayrılarak ağ yığın öğrenme yaklaşımıyla ön eğitime tabi tutuldu. Yığın öğrenme tamamlandıktan sonra ağ, tüm veri seti kullanılarak tekrar eğitildi. Her bir örneğin farklı çözüm uzayları ve yaklaşımlar üzerinde 10 kez test edilmesi sağlandı.

Sonuçlar, her bir durum için elde edilen gerçek çözümler ve ResNet çözümlerinin grafikleriyle sunuldu. Elde edilen mutlak hata grafikleri incelendiğinde, farklı sınır koşulları ve çözüm uzayları için farklı hata değerleri elde edildi. Örnek 3.1 için OBC sınır koşulları ve MC çözüm uzayı üzerinde yapılan testlerde elde edilen karesel hataların ortalaması $2{,}814\times10^{-3}$ ve standart sapması $8{,}692\times10^{-4}$ olarak bulundu. Aynı örnek için OBC sınır koşulları ve STC

çözüm uzayı üzerinde yapılan testlerde ise kabul edilebilir ortalama karesel hata değerleri $3{,}074\times10^{-3}$ ortalama ve $1{,}312\times10^{-3}$ standart sapma değeriyle bulundu.

Örnek 3.2 için ise OBC sınır koşulları ve MC çözüm uzayı üzerinde yapılan testlerde ortalama karesel hataların ortalaması $1{,}488\times10^{-3}$ ve standart sapması $3{,}767\times10^{-4}$ olarak hesaplandı. Aynı örnek için OBC sınır koşulları ve STC çözüm uzayı üzerinde yapılan testlerde ise ortalama karesel hata değerleri $1{,}692\times10^{-3}$ ortalama ve $7{,}105\times10^{-4}$ standart sapma değerleriyle elde edildi.

Bu sonuçlar, farklı sınır koşulları ve çözüm uzayları için yapılan testlerde farklı hata değerleri olduğunu göstermektedir. MC çözüm uzayı, STC çözüm uzayına göre daha yüksek hata değerlerine sahip olmuştur. Bu durum, çözüm uzayının şeklinin çözüm kalitesini etkilediğini göstermektedir. Ayrıca, elde edilen sonuçlar ResNet çözümlerinin gerçek çözümlere oldukça yakın olduğunu göstermektedir. Bu da Rezidü Yapay Sinir Ağı'nın nümerik çözümleri üretme yeteneğini göstermektedir.

6. SONUÇ VE ÖNERİLER

Kısmi türevli diferansiyel denklemlerin, klasik iteratif yöntemlerle nümerik çözümleri elde edilirken istenilen hata payıyla çözüme yakınsama sağlanması için genellikle problemin tanım bölgesinin ayrıştırılmasıyla elde edilen çok sayıda düğüm noktasına ihtiyaç duyulur. Bu ayrıklaştırma bölge üzerinde bir ızgara oluşturularak veya üçgenselleştirme yaklaşımlarıyla gerçekleştirilir. Ancak bu yollarla düğüm noktaların elde edilmesi genellikle oldukça maliyetlidir. Ayrıca, bazı durumlarda gerçek çözümün kendisi bile sınırlı sayıda noktada mevcut olabilir. Yapay sinir ağı modelleri ile nümerik çözüm elde etmek içinde aynı durum söz konusudur. Bununla birlikte klasik yöntemler sadece düğüm noktalarında yaklaşık çözüm sunarken sinir ağı modelleri tüm tanım bölgesi üzerinde bir çözüm sunabilirler.

Kısmi türevli diferansiyel denklemlerin yapay sinir ağı çözümleri için önerilen modelin başlangıç ve sınır koşullarını sağlaması gerekmektedir. Sinir ağı modeli bu koşulları öğrenmede zorluk yaşayabilir. Karşılaşılan bu problemin üstesinden gelmek için tezde deneme fonksiyonları kullanılması önerilmektedir. Deneme fonksiyonları başlangıç ve sınır koşullarını diferansiyel denklemle birlikte sağlayan ve sinir ağı modelinin çıktısına bağlı fonksiyonlardır.

Tezde, önerilen sinir ağı modelinin DistMesh üçgenselleştime yaklaşımıyla elde edilen düğüm noktalarından oluşan eğitim verisi kullanılarak kısmi diferansiyel denklemlerin çözümünü tahmin edebildiği gösterilmiştir. Ancak, önerilen modelin genelleştirme yeteneği sınırlı olabilir. Yani, her ne kadar tezde sunulan örnekler için, test aşamasında tanım bölgesi üzerinde rassal olarak üretilen noktalarda önerilen model kabul edilebilir çözümler üretse de farklı örnekler için eğitim verileri dışındaki noktalarda önerilen modelin yaklaşık çözümü tahmin etme yeteneği zayıf olabilir. Böyle bir sorunla karşılaşıldığında sinir ağı modelinin karmaşıklığı arttırılabilir veya önerilen optimizasyon yaklaşımı yerine alternatif teknikler denenebilir.

Dikkat edilmesi gereken bir diğer husus, yapay sinir ağlarının eğitim aşamasında düğüm noktalarında çözüme aşırı uyumluluk sağlayabilmeleridir. Ağlar eğitim verisine aşırı uyum sağlayabilir ve genelleme yeteneklerini kaybedebilir, bunun sonucunda sinir ağı modelinin ürettiği nümerik çözümler gerçek çözümlerden uzaklaşır. Tezde bu sorunun üstesinden gelebilmek için yığın öğrenme (batch learning) yaklaşımı kullanılmıştır. Bölüm 3'te verilen

örneklerde problemin tanım bölgesi üzerinde DistMesh üçgenleştirme yaklaşımı kullanılarak yaklaşık 8000 düğüm noktası elde edilmiştir. Bu nokta kümesinden rassal olarak seçilen 1500 noktalı alt eğitim kümeleri oluşturulmuş ve önerilen model bu kümelerdeki düğüm noktaları ile art arda eğitilmiştir. Böylece ağın eğitim kümesindeki noktalara aşırı uyum sağlaması önlenmiştir. Eğitim aşaması tamamlandıktan sonra bölge üzerinde rastgele üretilen noktalarda sinir ağı modelinin çözümleri elde edilmiştir. Test işlemi farklı ağ parametreleri ile 10 kez tekrarlanmıştır ve Bölüm 4'te görülebileceği üzere performans analizi ortalama karesel hataların ortalaması ve standart sapması hesaplanarak gerçekleştirilmiştir. Elde edilen bulgular göstermektedir ki her test işleminde kabul edilebilir hata oranıyla çözüme yakınsanmaktadır. Tüm örneklerde elde edilen standart sapma değerlerinin düşük olması da eğitim kümesindeki düğüm noktalarında aşırı uyum sorunundan kaçınıldığını ispatlar niteliktedir.

Diğer yandan kısmi türevli diferansiyel denklemleri çözmek için kullanılan sinir ağı modelleri, kompleks bölgeler üzerinde karmaşık denklemlerin davranışlarını tanımlamada yeterli model karmaşıklığına sahip olmayabilir. Modelin topolojisi çok yalın ise ağ istenilen çözümleri üretemeyebilir. Tersine model oldukça bir karmaşık topolojiye sahipse bu kez de ağın eğitimi çok maliyetli olur. Bu tez çalışması süresince de görüldüğü üzere kısmi türevli diferansiyel denklemi çözebilecek bir optimal sinir ağı topolojisini belirlemek oldukça güçtür. Bu nedenle bu tezde de olduğu gibi ağ modeli çoğu zaman sezgisel olarak belirlenir. Haliyle tezde elde edilen nümerik sonuçların optimal sonuçlar olduğu söylenemez ve önerilen modelden daha iyi çözüm üreten daha farklı topolojiye sahip bir sinir ağı modeli bulunması daima olasıdır.

Ağ topolojisini belirlerken ara katman sayılarının, her ara katmandaki proses elemanlarının (nöronların) sayısının ve ara katmanlar arasında geçişi sağlayan aktivasyon fonksiyonlarının seçimi de üretilen nümerik sonuçlar üzerinde etkiye sahiptir. Özellikle aktivasyon fonksiyonlarının doğru bir şekilde ayarlanması, ağın daha iyi performans göstermesine yardımcı olabilir. Tezde deneme-yanılma yoluyla aktivasyon fonksiyonu olarak ReLU (Rectified Linear Unit) fonksiyonu seçilmiştir.

Sonuç olarak, bu tezde farklı sınır koşulları ve çözüm uzayları için Rezidü Yapay Sinir Ağı kullanarak eliptik kısmi diferansiyel denklemlerin nümerik çözümü gerçekleştirmiştir ve bu çalışma, benzer problemlerin çözümünde yapay sinir ağlarının kullanılabilirliğini ve etkinliğini vurgulamaktadır.

KAYNAKLAR

Aarts, L. P., van der Veer, P. (2001). Neural Network Method for Solving Partial Differential Equations. Neural Processing Letters, 14(3), 261–271. doi: 10.1023/a:1012784129883

Amaioua, N., Audet, C., Conn, A. R., Le Digabel, S. (2018). Efficient solution of quadratically constrained quadratic subproblems within the mesh adaptive direct search algorithm. *European Journal of Operational Research*, *268*(1), 13–24. doi: 10.1016/j.ejor.2017.10.058

Asem, M. (2020). DiffusionNet: Accelerating the solution of Time-Dependent partial differential equations using deep learning. *Arxiv.org*. doi: 10.48550/arXiv.2011.10015

Audet, C., Dennis, J. E. (2006). Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, *17*(1), 188–217. doi: 10.1137/040603371

Audet, C., Hare, W. (2017). Mesh adaptive direct search. In: Derivative-free and blackbox optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham. Pp. 135–156. doi: 10.1007/978-3-319-68913-5_8

Beck, C., Becker, S., Cheridito, P., Jentzen, A., Neufeld, A. (2020). Deep learning based numerical approximation algorithms for stochastic partial differential equations and high-dimensional nonlinear filtering problems. ArXiv:2012.01194 [Cs, Math, Stat]. doi: 10.48550/arXiv.2012.01194

Beck, C., Becker, S., Cheridito, P., Jentzen, A., Neufeld, A. (2021). Deep Splitting Method for Parabolic PDEs. *SIAM Journal on Scientific Computing*, *43*(5), A3135–A3154. doi: 10.1137/19m1297919

Beck, C., Becker, S., Grohs, P., Jaafari, N., Jentzen, A. (2021). Solving the Kolmogorov PDE by Means of Deep Learning. *Journal of Scientific Computing*, *88*(3). doi : 10.1007/s10915-021-01590-0

Berg, J., Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, *317*, 28–41. doi: 10.1016/j.neucom.2018.06.056

Berg, J., Nyström, K. (2019). Data-driven discovery of PDEs in complex datasets. *Journal of Computational Physics*, *384*, 239–252. doi: 10.1016/j.jcp.2019.01.036

Berner, J., Dablander, M., Grohs, P. (2020). Numerically Solving Parametric Families of High-Dimensional Kolmogorov Partial Differential Equations via Deep Learning. doi: 10.48550/arXiv.2011.04602

Cai, Z., Chen, J., Liu, M., Liu, X. (2020). Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *Journal of Computational Physics*, *420*, 109707. doi: 10.1016/j.jcp.2020.109707

E, W., Han, J., Jentzen, A. (2017). Deep Learning-Based Numerical Methods for High-Dimensional Parabolic Partial Differential Equations and Backward Stochastic Differential Equations. *Communications in Mathematics and Statistics*, *5*(4), 349–380. doi: 10.1007/s40304-017-0117-6

E, W., Yu, B. (2018). The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems. *Communications in Mathematics and Statistics*, *6*(1), 1–12. doi: 10.1007/s40304-018-0127-z

Eskiizmirliler, S., Günel, K., Polat, R. (2021). On the Solution of the Black–Scholes Equation Using Feed-Forward Neural Networks. *Comput Econ*, *58*, 915–941 doi: 10.1007/s10614-020-10070-w

Geist, M., Petersen, P., Raslan, M., Schneider, R., Kutyniok, G. (2021). Numerical Solution of the Parametric Diffusion Equation by Deep Neural Networks. *Journal of Scientific Computing*, *88*(1). doi: 10.1007/s10915-021-01532-w

Gör, İ. (2020). Diferansiyel Denklemlerin Yapay Sinir Ağları ile Nümerik Çözümleri, Doktora Tezi, Aydın Adnan Menderes Üniversitesi, Fen Bilimleri Enstitüsü, Matematik Anabilim Dalı, Aydın.

Han, J., Jentzen, A., E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, *115*(34), 8505–8510. doi: 10.1073/pnas.1718942115

He, K., Zhang, X., Ren, S., Sun, J., (2015). Deep residual learning for image recognition. Technical Report. Microsoft Research. doi: 10.48550/arXiv.1512.03385

Innes, M. (2018). Flux: Elegant machine learning with Julia. *Journal of Open Source Software*, *3*(25), 602, doi: 10.21105/joss.00602

Innes, M., Saba, E., Fischer, K., Gandhi, D., vd. (2018). Fashionable modelling with flux. arXiv preprint doi: 10.48550/arXiv.1811.01457

Karumuri, S., Tripathy, R., Bilionis, I., Panchal, J. (2020). Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks. *Journal of Computational Physics*, *404*, 109120. doi: 10.1016/j.jcp.2019.109120

Lagaris, I. E., Likas, A., Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, *9*(5), 987–1000. doi: 10.1109/72.712178

Lee, H., Kang, I. S. (1990). Neural algorithm for solving differential equations. *Journal of Computational Physics*, *91*(1), 110–131. doi: 10.1016/0021-9991(90)90007-n

Li, K., Tang, K., Wu, T., Liao, Q. (2019). D3M: A deep domain decomposition method for partial differential equations. *IEEE Access*, *8*, 5283–5294. doi: 10.1109/access.2019.2957200

Malek, A., Shekari Beidokhti, R. (2006). Numerical solution for high order differential equations using a hybrid neural network—Optimization method. *Applied Mathematics and Computation*, *183*(1), 260–271. doi: 10.1016/j.amc.2006.05.068

Meade, A. J., Fernandez, A. A. (1994). The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, *19*(12), 1–25. doi: 10.1016/0895-7177(94)90095-7

Nabian, M. A., Meidani, H. (2019). A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Engineering Mechanics*, *57*, 14–25. doi: 10.1016/j.probengmech.2019.05.001

Persson, P.-O., Strang, G. (2005). A Simple Mesh Generator in MATLAB. *SIAM Review*, *46*(2), 329–345. doi: 10.1137/s0036144503429121

Raissi, M. (2018). Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations. *ArXiv:1804.07010 [Cs, Math, Stat]*. doi: 10.48550/arXiv.1804.07010

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. doi: 10.1016/j.neunet.2014.09.003

Sheng, H., Yang, C. (2021). PFNN: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries. *Journal of Computational Physics*, *428*, 110085. doi: 10.1016/j.jcp.2020.110085

Wang, L., Persson., P.O. (2015). A high-order discontinuous galerkin method with unstructured space–time meshes for two-dimensional compressible flows on domains with large deformations. *Computers & Fluids*, *118*, 53–68, doi: 10.1016/j.compfluid.2015.05.026.

Wang, Z., Weng, F., Liu, J., Cao, K., Hou, M., Wang, J. (2021). Numerical solution for high-dimensional partial differential equations based on deep learning with residual learning and data-driven learning. *International Journal of Machine Learning and Cybernetics*, *12*. doi: 10.1007/s13042-021-01277-w

Xu, H., Zhang, D., Zeng, J. (2021). Deep-learning of parametric partial differential equations from sparse and noisy data. *Physics of Fluids*, *33*(3), 037132. doi: 10.1063/5.0042868

Zakeri, B., Monsefi, A. K., Darafarin, B. (2020). Deep Learning Prediction of Heat Propagation on 2-D Domain via Numerical Solution. *Data Science: From Research to Application*, *45*, 161–174. doi: 10.1007/978-3-030-37309-2_13

EKLER

Ek 1. OBC sınır koşulları ile verilen Örnek 3.1'in MC bölgesi üzerinde ResNet çözümleri elde etmede kullanılan Julia kodu

```
# Problem: u_xx + u_yy= -pi^2*sin(pi*x*y)*(x^2+y^2)
# x0 = 0, y0 = 0, x1 = sin(pi*y), y1 = sin(pi*x)
# Analytic Solution : u = sin(pi*x*y)
using ModelingToolkit, Flux, Statistics, NOMAD, Makie, Gadfly, DistMesh2D
using CairoMakie, ImageCore, GeometryBasics, Triangulate, Colors, ColorSchemes
using DataFrames, CSV, Printf, Random

##DistMesh
function drectangle(
p,
x1::T,
x2::T,
y1::T,
y2::T,
)::T where {T<:Float64}
return -min(min(min(-y1 + p[2], y2 - p[2], -x1 + p[1], x2 - p[1])))
end

function dcircle(
p,
xc::T,
yc::T,
r::T
)::T where {T<:Float64}
return sqrt((p[1] - xc) .^ 2 + (p[2] - yc) .^ 2) - r
end

function dunion(
p,
dfs::Function...,
)::Float64
return min([fun(p) for fun in dfs]...)
end

function ddiff(
p,
d1::Function,
dfs::Function...
)::Float64
return max(d1(p), [-fun(p) for fun in dfs]...)
end

function dintersect(
p,
dfs::Function...
)::Float64
return max([fun(p) for fun in dfs]...)
```

```
    end

    function huniform(p)::Float64
    return ones(size(p, 1), 1)[1]
    end

    function protate(p,phi)
    return p*[cos(phi) -sin(phi); sin(phi) cos(phi)]
    end

    # define the region
    M1_x, M1_y, r1 = 0.5, 0.5, 0.4
    M2_x, M2_y, r2 = 0.3, 0.7, 0.25
    M3_x, M3_y, r3 = 0.7, 0.7, 0.25
    low, up = -100.0, 100.0
    scale_factor = 100
    outrect(p) = drectangle(p, low, up, low, up)
    incircle1(p) = dcircle(p, low + (up -low)*M1_x, low + (up -low)*M1_y, scale_factor*r1)
    incircle2(p) = dcircle(p, low + (up -low)*M2_x, low + (up -low)*M2_y, scale_factor*r2)
    incircle3(p) = dcircle(p, low + (up -low)*M3_x, low + (up -low)*M3_y, scale_factor*r3)
    fd(p) = ddiff(p, outrect, incircle1,incircle2,incircle3)
    fh(p) = min(4 * sqrt(sum(p.^2)) - 100)
    bbox = [low low; up up]
    h0 = 5.0
    pfix = [low low; low up; up low; up up]
    dptol = 0.05
    # Discretize the region
    x, y = distmesh2d(fd, fh, bbox, h0, pfix=pfix, dptol=dptol)

    # scale the region in [0.1]
    x = ((x ./ 100) .+1) ./2
    y = ((y ./ 100) .+1) ./2
    original_x = x
    original_y = y
    gdPlt = Gadfly.plot(x = x, y = y, Geom.path, Coord.cartesian(fixed = true))
    data = [x y]
    originalData = data

    # Store the indices of NaN values
    indices = findall(isnan,data[:,1])
    newnames = vec(string.(["x","y"]))
    # Create a dataframe
    df = DataFrame(data, newnames)
    rename!(df, newnames)
    # Clear NaN Values
    df = filter(:x => x -> !any(f -> f(x), (ismissing, isnothing, isnan)), df)
    data = Array(df)
    data = Array(data')
    x = data[1,:]
    y = data[2,:]

    @parameters x, y
    @variables u(..)

    ## Neural network
    struct IdentitySkip
    inner
    end
```

```
(m::IdentitySkip)(x) = m.inner(x) .+ x
noInputs = 2 # number of dimensions
global net = Chain(Dense(noInputs,8,relu), IdentitySkip(Dense(8,8)), Dense(8,2,relu), IdentitySkip(Dense(2, 2)), Dense(2,1))
weights, reconstruct = Flux.destructure(net)
dim = length(weights)
weights = -10 .+ 20*rand(Float64, dim)
net = reconstruct(weights)

## Construct Trial Solution
function trialSolution(data, net)
x = data[1,:]
y = data[2,:]
x = x'
y = y'

data_x_PlusEps = data
data_x_MinusEps = data
data_x_PlusEps[1,:] .+= sqrt(eps())
data_x_MinusEps[1,:] .-= sqrt(eps())

data_y_PlusEps = data
data_y_MinusEps = data
data_y_PlusEps[2,:] .+= sqrt(eps())
data_y_MinusEps[2,:] .-= sqrt(eps())

dxNet = (net(data_x_PlusEps) .- net(data)) ./ sqrt(eps())
d2xNet = (net(data_x_PlusEps) .- 2* net(data) .+ net(data_x_MinusEps)) ./ eps()
dyNet = (net(data_y_PlusEps) .- net(data)) ./ sqrt(eps())
d2yNet = (net(data_y_PlusEps) .- 2*net(data) .+ net(data_y_MinusEps)) ./ eps()

u = x .* sin.(π*y) .+ y .* sin.(π*x) .+ x .* y .* (1 .- x) .* (1 .- y) .* net(data)
du_dx = sin.(π*y) .+ π*y .* cos.(π*x) .+ y.*(1 .- y) .* (1 .- 2*x) .* dxNet
d2u_dx2 = -π^2*y .* sin.(π*x) .+ y .* (1 .- y) .* ((1 .- 2*x) .* dxNet .- 2*net(data) .+ (x .- x .^ 2) .* d2xNet .+ (1 .- 2*x) .* dxNet)
du_dy = sin.(π*x) .+ π*x .* cos.(π*y) .+ x .* (1 .- x) .* (1 .- 2*y) .* dyNet
d2u_dy2 = -π^2*x .* sin.(π*y) .+ x .* (1 .- x) .* ((1 .- 2*y) .* dyNet .- 2*net(data) .+ (y .- y .^ 2) .* d2yNet .+ (1 .- 2*y) .* dyNet)

u, du_dx, du_dy, d2u_dx2, d2u_dy2
end

function loss(data,net)
x = data[1,:]
y = data[2,:]
u, du_dx, du_dy, d2u_dx2, d2u_dy2 = trialSolution(data, net)
lhs = d2u_dx2 .+ d2u_dy2
rhs = F(x, y)'
Flux.Losses.mse(lhs,rhs)
end

## Define Problem

function F(x,y)
-π^2*sin.(π*x .* y) .* (x .^ 2 .+ y .^ 2)
end

analyticSol(x,y) = sin.(π.*x.*y)
```

```
161
162 #Boundary Conditions
163 #bcs = [u(0,y) ~ 0.,
164 #  u(1,y) ~ sin(π*y),
165 #  u(x,0) ~ 0.,
166 #  u(x,1) ~ sin(π*x)]
167
168 # constraints
169 function c(x)
170 return 0
171 end
172
173 function eval_fct(x)
174 net = reconstruct(x)
175 bb_outputs = [loss(data,net), c(x)]
176 success = true
177 count_eval = true
178 return (success, count_eval, bb_outputs)
179 end
180
181 VarMin = -10 .* ones(Float64, dim)
182 VarMax = 10 .* ones(Float64, dim)
183 pb = NomadProblem(dim, # number of inputs of the blackbox
    1, # number of outputs of the blackbox
    ["OBJ"], # type of outputs of the blackbox
    eval_fct;
    lower_bound=VarMin,
    upper_bound=VarMax)
184
185 # Fix some options
186 pb.options.max_bb_eval = 2*10^4
187 pb.options.display_all_eval = true
188 #pb.options.display_stats = ["BBE", "EVAL", "SOL", "OBJ", "CONS_H"]
189 #some display options
190 pb.options.display_stats = ["BBE", "EVAL", "OBJ"] # some display options
191
192 ## Train Network using Batch Learning via NOMAD
193 data2 = data;
194 sample_size = size(data2,2)
195 data = data[:,randperm(sample_size)]
196 batch_size = 1500
197 no_batch = Int32(floor(sample_size/batch_size))
198 for i in 1:no_batch
199 global weights, reconstruct = Flux.destructure(net)
200 indices = (i-1)*batch_size+1:i*batch_size
201 global data = data2[:,indices]
202 @printf("Batch No: %d\n",i)
203 W = Vector{Float64}(weights)
204 @elapsed result = NOMAD.solve(pb,W)
205 println("Solution: ", result.x_best_feas)
206 W = Vector{Float64}(result.x_best_feas)
207 global net = reconstruct(W)
208 println("------------------------------------------------------------")
209 end
210 println("------------------------------------------------------------")
211 data = data2
212 # Train network with all data
213 W = Vector{Float64}(weights)
```

```
@elapsed result = NOMAD.solve(pb,W)
println("Solution: ", result.x_best_feas)
weights = Vector{Float32}(result.x_best_feas)
net = reconstruct(weights)

## Get solutions
u_real = analyticSol(data[1,:], data[2,:])
u_predict, du_dx, du_dy, d2u_dx2, d2u_dy2 = trialSolution(data, net)
Err = abs.(reshape(u_real,1,length(u_real)) .- u_predict)

## Construct mesh
data2 = data
u_real2 = analyticSol(data2[1,:], data2[2,:])
u_predict2, du_dx, du_dy, d2u_dx2, d2u_dy2 = trialSolution(data2, net)
innernodes = Point[]
k=1
while k ≤ size(data2,2)
p = Point(data2[:,k]...)
push!(innernodes, p)
global k = k + 1
end
innernodes = [getindex.(innernodes, 1) getindex.(innernodes, 2)]

# Triangulate the points.
triin = TriangulateIO()
triin.pointlist = innernodes'
msh, _ = triangulate("vQ", triin)

# Convert TriangleMesh.TriMesh to GeometryBasics.Mesh
pts = [Point(val[1], val[2]) for val in eachcol(msh.pointlist)]
fcs = [TriangleFace(val[1], val[2], val[3]) for val in eachcol(msh.trianglelist)]

## Plots
CairoMakie.activate!()
low = colorant"navyblue";
medium = colorant"dodgerblue";
high = colorant"limegreen";
min_u = minimum([minimum(u_real) minimum(u_predict)])
max_u = maximum([maximum(u_real) maximum(u_predict)])
Ucolormap = colorsigned(low,medium,high) ∘ scalesigned(min_u,(3*min_u+max_u)/4,(3*min_u+2*max_u)/4)
Uimage = Ucolormap.(u_predict)

low2 = high;
medium2 = colorant"yellow";
high2 = colorant"orangered";
Ucolormap2 = colorsigned(low2,medium2,high2) ∘ scalesigned((3*min_u+2*max_u)/4,(3*min_u+3*max_u)/4,max_u)

# Plot ResNet Solution
for i in 1:length(u_predict)
if u_predict[i] >(min_u+max_u)/2
Uimage[i] = Ucolormap2(u_predict[i]);
end
end

cm=("navyblue","dodgerblue","limegreen","yellow","orangered")
cm = ColorScheme([range(colorant"navyblue", colorant"dodgerblue", length=100);
```

```
range(colorant"dodgerblue", colorant"limegreen", length=100);
range(colorant"limegreen", colorant"yellow", length=100);
range(colorant"yellow", colorant"orangered", length=100)])

msh = GeometryBasics.Mesh(pts, fcs)
fig,ax,figMesh = Makie.mesh(msh, colormap = cm, color = Uimage[1 : length(pts)],
axis = (aspect = 1, xlabel = L"x", ylabel = L"y",
title = "ResNet Solution",ylabelsize = 22,xlabelsize= 22))
cb = Colorbar(fig[1, 2],colormap=cm, limits = (min_u, max_u), ticks = min_u:0.25:max_u,label = L"u(x,y)",
height = Relative(1), width = 15, labelsize = 22, halign = :right,
valign = :bottom,highclip = :orangered, lowclip = :navyblue)
Makie.wireframe!(msh,overdraw = true, transparency = true, color = (:black, 0.1), linewidth = 2)
#fig=current_figure()
poly!(Circle(Point(M1_x, M1_y), r1/2), color = :black)
poly!(Circle(Point(M2_x, M2_y), r2/2), color = :black)
poly!(Circle(Point(M3_x, M3_y), r3/2), color = :black)
fig=current_figure()
save("../Julia_Projects/outs/ex1_v1/plot_ResNet_Solution_" * ARGS[1] * ".png", fig, px_per_unit = 2) # double the resolution of the resulting PNG

# Plot exact solution
Uimage = Ucolormap.(u_real)
for i in 1:length(u_real)
if u_real[i] > (min_u+max_u)/2
Uimage[i] = Ucolormap2(u_real[i]);
end
end

fig2,ax2,figMesh2 = Makie.mesh(msh, colormap = cm, color = Uimage[1 : length(pts)],
axis = (aspect = 1, xlabel = L"x", ylabel = L"y",
title = "Exact Solution",ylabelsize = 22, xlabelsize= 22))
cb = Colorbar(fig2[1, 2],colormap=cm, limits = (min_u, max_u), ticks = min_u:0.25:max_u,label = L"u(x,y)",
height = Relative(1), width = 15, labelsize = 22, halign = :right,
valign = :bottom,highclip = :orangered, lowclip = :navyblue)
Makie.wireframe!(msh,overdraw = true, transparency = true, color = (:black, 0.1), linewidth = 2)
poly!(Circle(Point(M1_x, M1_y), r1/2), color = :black)
poly!(Circle(Point(M2_x, M2_y), r2/2), color = :black)
poly!(Circle(Point(M3_x, M3_y), r3/2), color = :black)
fig=current_figure()
save("../Julia_Projects/outs/ex1_v1/plot_Exact_Solution_" * ARGS[1] * ".png", fig, px_per_unit = 2)
# double the resolution of the resulting PNG

# Plot Absolute Errors
Uimage = Ucolormap.(Err)
for i in 1:length(Err)
if Err[i] > (min_u+max_u)/2
Uimage[i] = Ucolormap2(Err[i]);
end
end

fig3,ax3,figMesh3 = Makie.mesh(msh,
colormap = cm, color = Uimage[1 : length(pts)],
axis = (aspect = 1, xlabel = L"x", ylabel = L"y",ylabelsize = 22,
xlabelsize= 22))
cb = Colorbar(fig3[1, 2],colormap=cm, limits = (min_u, max_u), ticks = min_u:0.25:max_u,label = "Absolute Errors",
```

```
    height = Relative(1), width = 15, labelsize = 22, halign = :right,
    valign = :bottom,highclip = :orangered, lowclip = :navyblue)
309 Makie.wireframe!(msh,overdraw = true, transparency = true, color = (:black, 0.1), linewidth = 2)
310 poly!(Circle(Point(M1_x, M1_y), r1/2), color = :black)
311 poly!(Circle(Point(M2_x, M2_y), r2/2), color = :black)
312 poly!(Circle(Point(M3_x, M3_y), r3/2), color = :black)
313 fig=current_figure()
314 save("../Julia_Projects/outs/ex1_v1/plot_Absolute_Errors_" * ARGS[1] * ".png", fig, px_per_unit = 2)
    # double the resolution of the resulting PNG
315
316 ## Display results
317 # MSE
318 x = data[1,:]
319 y = data[2,:]
320 errs = abs.(u_real .- vec(u_predict))
321 df = DataFrame(x=x, y = y, Exact_Solution=vec(u_real),ResNet_Solution=vec(u_predict),
322 Error=vec(errs))
323 # Show all rows of DataFrame
324 show(df, allrows=false)
325
326 println("")
327 println("MSE for training set : ",Flux.Losses.mse(vec(u_real),vec(u_predict)))
328 CSV.write("../Julia_Projects/outs/ex1_v1/results_training_set_" * ARGS[1] * ".csv", df)
329
330 x_test = vec(rand(21,1))
331 y_test = vec(rand(21,1))
332 u_real = analyticSol(x_test, y_test)
333 data = Array([x_test y_test])
334 data = data'
335 u_predict, du_dx, du_dy, d2u_dx2, d2u_dy2 = trialSolution(data, net)
336 errs = abs.(vec(u_real) .- vec(u_predict))
337 df_test = DataFrame(x=vec(x_test), y=vec(y_test),
    Exact_Solution=vec(u_real),ResNet_Solution=vec(u_predict),Error=vec(errs))
338 # Show all rows of DataFrame
339 show(df_test, allrows=true)
340
341 println("")
342 println("MSE for test set : ",Flux.Losses.mse(vec(u_real),vec(u_predict)))
343 CSV.write("../Julia_Projects/outs/ex1_v1/results_test_set_" * ARGS[1] * ".csv", df_test)
```

Ek 2. Julia Implementasyonunun ADU-BAP Fizik/Matematik Yüksek Başarımlı Hesaplama Sistemi Sunucuları Üzerinde Çalışabilmesi için Toplu Betik Dosyası Örneği

```
1   #!/bin/bash
2
3   ## Resource Request
4   #SBATCH --job-name=Julia_ResNet_Job
5   #SBATCH --output=./outs/ex1_v1/Julia_ResNet_Job-%A_%a.out
    # %A in the #SBATCH line becomes the job ID, %a in the #SBATCH line becomes the array
    index
6   #SBATCH --time=1-12:00:00
7   #SBATCH --ntasks=10
8   #SBATCH --cpus-per-task=1
9   #SBATCH -A [account / project name]
10  #SBATCH -p [partition (queue) name]
11  #SBATCH --mem-per-cpu=512M
12  #SBATCH --mail-user=user@adu.edu.tr
13  #SBATCH --mail-type=ALL
14  #SBATCH --array=0-9    # job array index
15  #SBATCH --get-user-env
16
17  # Load software (Julia default is 1.8.5 version)
18  module load julia
19
20  ## Job Steps
21  srun julia ResNet-nomad_2023.04.17_ex1_v1.jl $SLURM_ARRAY_TASK_ID
```

T.C. T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLİMSEL ETİK BEYANI

"Kompleks Geometrilerde Kısmi Türevli Diferansiyel Denklemlerin Derin Öğrenme Yaklaşımları ile Nümerik Çözümleri" başlıklı Doktora tezimdeki bütün bilgileri etik davranış ve akademik kurallar çerçevesinde elde ettiğimi, tez yazım kurallarına uygun olarak hazırlanan bu çalışmada, bana ait olmayan her türlü ifade ve bilginin kaynağına eksiz atıf yaptığımı bildiririm. İfade ettiklerimin aksi ortaya çıktığında ise her türlü yasal sonucu kabul ettiğimi beyan ederim.

04/08/2023

Özcan KOLYİĞİT

ÖZ GEÇMİŞ

Soyadı, Adı : Kolyiğit, Özcan

Yabancı Dil : İngilizce

EĞİTİM

Derece	Yer, Kurum	Mezuniyet
Doktora	Aydın, Aydın Adnan Menderes Üniversitesi Matematik A.B.D.	2023
Y. Lisans	Aydın, Aydın Adnan Menderes Üniversitesi Matematik A.B.D.	2013
Lisans	Aydın, Aydın Adnan Menderes Üniversitesi Fen Edebiyat Fakültesi Matematik Bölümü	2010

İŞ DENEYİMİ

Yıl	Yer, Kurum	Unvan
2015 -	MEB	Öğretmen
2010 - 2014	Aydın Uğur Dershanesi	Öğretmen

AKADEMİK YAYINLAR

1. BİLDİRİLER

A) Uluslararası Kongrelerde Yapılan Bildiriler

Kolyiğit, Ö., Günel, K. (2022). Numerical Solutions of Partial Differential Equations on Non-Rectangular Geometries with Deep Learning Approaches, International Conference on Artificial Intelligence and Applied Mathematics (ICAIAME 2022), 20-22 May 2022, Baku, Azerbaijan.

Ahmad, M.J., Kolyiğit, Ö. ve Günel, K. (2024). A Piece-wise Trial Solution for PDEs Defined with Boundary Conditions on the Inner Voids, The SIAM Conference on Parallel Processing for Scientific Computing (PP24), March 5 - 8, 2024, Baltimore, Maryland, U.S. (submitted)

B) Ulusal Kongrelerde Yapılan Bildiriler

Kolyiğit, Ö., Aşlıyan, R., Günel, K. (2012). Türkçe Dokümanlar İçin Yazar Tanıma. Akademik Bilişim'12 - XIV. Akademik Bilişim Konferansı Bildirileri, 1 - 3 Şubat 2012, Uşak Üniversitesi