T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MESAFEYE BAĞLI TERS AĞIRLIK YÖNTEMİ İLE GRİDLEME VE KONTUR ÇİZİMİ

İRFAN DUMAN

YÜKSEK LİSANS TEZİ
ELEKTRİK-ELEKTRONİK VE BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

DANIŞMAN
PROF. DR. RESUL KARA

DÜZCE, 2018

T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MESAFEYE BAĞLI TERS AĞIRLIK YÖNTEMİ İLE GRİDLEME VE KONTUR ÇİZİMİ

İrfan DUMAN tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Resul KARA

Düzce Üniversitesi

Eş Danışman

Dr. Öğr. Üyesi Erkan ÇETİNER

Zonguldak Bülent Ecevit Üniversitesi

Jüri Üyeleri

Prof. Dr. Resul KARA
Düzce Üniversitesi _____

Dr. Öğr. Üyesi Şeyhmus AYDOĞDU
Nevşehir Hacı Bektaş Veli Üniversitesi _____

Dr. Öğr. Üyesi Fatih KAYAALP
Düzce Üniversitesi _____

Tez Savunma Tarihi: 11/10/2018

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

11 Ekim 2018

(İmza)

İrfan DUMAN

TEŞEKKÜR

Yüksek lisans öğrenimimde ve bu tezin hazırlanmasında gösterdiği her türlü destek ve yardımdan dolayı çok değerli hocam Prof. Dr. Resul KARA'ya en içten dileklerimle teşekkür ederim.

Tez çalışmam boyunca değerli katkılarını esirgemeyen eş danışmanım Dr. Öğr. Üyesi Erkan ÇETİNER'e de şükranlarımı sunarım.

Bu çalışma boyunca yardımlarını ve desteklerini esirgemeyen sevgili aileme ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

Bu tez çalışması, Düzce Üniversitesi BAP-2016.06.01.446 numaralı Bilimsel Araştırma Projesiyle desteklenmiştir.

11 Ekim 2018 **İrfan DUMAN**

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

BT	Bilgisayarlı tomografi
CPU	Central processing unit – Merkezi işlemci birimi
C#	C Sharp
ERT	Eksternal radyoterapi
GDI	Graphics device interface – Grafik aygıt arabirimi
GUI	Graphical user interface – Grafiksel kullanıcı arayüzü
GPU	Graphics processing unit – Grafik işleme birimi
GPS	Global positioning system – Küresel yer belirleme sistemi
MR	Manyetik rezonans
MRG	Manyetik rezonans görüntüleme
SPECT	Tek foton emisyonlu bilgisayarlı tomografi
SQL	Structured query language – Yapılandırılmış sorgu dili
WPF	Windows presentation foundation – Windows gösterim altyapısı

ÖZET

MESAFEYE BAĞLI TERS AĞIRLIK YÖNTEMİ İLE GRİDLEME VE KONTUR ÇİZİMİ

İrfan DUMAN
Düzce Üniversitesi
Fen Bilimleri Enstitüsü, Elektirik-Elektronik ve Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Prof. Dr. Resul KARA
Ekim 2018, 49 sayfa

Birçok alanda kullanılan konturlama metodunun genel amacı bilimsel çalışmalarda elde edilen verinin hangi değerler etrafında yoğunlaştığını göstererek veriyi anlaşılabilir hale getirmektir. Bu çalışmada, herhangi bir alanda yapılacak çalışma için toplanan ham verileri ön işlemlerden geçirip konturlama metodu ile gösterebilmek için bir yazılım geliştirilmiştir. XYZ ham verileri ön işlemden geçirilirken, öncelikle bir koordinat ağı ya da veri düzlem ağı denilen grid oluşturulmuştur. Gridleme işleminde mesafenin tersi yöntemi kullanılmıştır. Grid alanının her bir düğüm noktası için bir XYZ verisi atanarak gridleme işlemi tamamlanmıştır. Veriler ön işlemden geçirilip gridleme tamamlandıktan sonra ise oluşturulan grid üzerine konturlama yapılarak toplanan verilerin kontur haritaları çıkartılmıştır. Önerilen yöntemle elde edilen sonuçlar Surfer yazılımından elde edilen sonuçlarla karşılaştırılmıştır. Karşılaştırma sonucunda aynı değerler ve kontur grafikleri elde edilmiştir. Karşılaştırma sonuçları göz önüne alındığında ticari ürünlere alternatif olarak kullanılabileceği düşünülmektedir.

Anahtar sözcükler: Gridleme, Konturlama, Mesafenin tersi yöntemi.

ABSTRACT

GRIDDING AND CONTOURING WITH INVERSE DISTANCE WEIGHT METHOD

İrfan DUMAN
Düzce University
Graduate School of Natural and Applied Sciences, Department of Elektrical-Electronics and Computer Engineering
Master's Thesis
Supervisor: Prof. Dr. Resul KARA
October 2018, 49 pages

The general purpose of the contouring method that used in many areas is to make the data understandable by showing the data concentration from obtained data in scientific studies. In this study, a software has been developed to show raw data gathered from any field of studied area with contouring method after pre-processed. During the pre-processing of XYZ raw data, at first, a grid called a coordinate network or a data plane network has been created. The inverse distance weight method was used in the gridding process. Gridding is completed by assigning an XYZ data for each node of the grid area. After the pre-processing the data and the gridding, contour maps of the collected data were obtained by contouring of the generated grid. The results obtained with the proposed method were compared with the results obtained from Surfer software. At the end of the comparison, the same values and contour graphs were obtained. Considering the comparison results, it is thought that it can be used as an alternative to commercial products.

Keywords: Contouring, Gridding, Inverse distance weight method.

1. GİRİŞ

İnsanlar, cevabını ya da nedenini merak ettiği şeyler ile karşılaşır ve bunun cevabını çevrelerinden veri toplayarak bulurlar. Bazen soruların cevabı tek bir veri ile cevaplanabilirken, bazen tek bir veri bu sorunun cevabını karşılayamamakta ve birden fazla veriyle istenilen cevaba ulaşıla bilinmektedir. Araştırmalar sonucunda elde edilen büyük miktardaki verileri daha anlaşılır ve kolay yorumlanabilir hale getirmek için grafikler kullanılmaktadır.

Kontur grafikleri ya da kontur çizgileri de var olan veya ölçüm yaparak elde edilen veri setlerindeki verilerin hangi değerler etrafında yoğunlaştığını göstermek amacıyla haritalarda sıklıkla kullanılmaktadır. Bu haritalardan biri de topografya haritalarıdır. Topografya haritalarında kontur çizgileri eşyükselti eğrileri olarak yorumlanmaktadır. Eşyükselti eğrileri, topografya haritalarında kullanılan en yaygın ve en gelişmiş yöntemdir. Eşyükselti eğrileri, adından da anlaşıldığı gibi, belli bir düzeyden (deniz düzeyi) başlayarak, topografyanın eşyükselti noktalarını birleştiren eğrilerdir. Bu eşyükselti eğrileri, yükselti dağılım ve değişimini göstermektedir [1], [2]. Yeraltı haritaları oluşturulurken yapılan sismik, manyometrik, gravimetrik vb. ölçümlerle elde edilen verilerin analizi sonrasında sahadaki veri değişimleri kontur çizgileri yardımıyla gösterilerek jeofizik haritaları hazırlanmaktadır [3].

Kontur çizgileri haritalar gibi birçok alanda verilerin değişimi ve yoğunlaştığı alanları daha kolay görebilmek ve yorumlayabilmek için kullanılmaktadır. Sayısallaştırılmış haritalar için veriler; saha araştırmalarından, kontur hatlarından, fotogrametri tekniklerinden, lazer altimetreler ve diğer toplanan düzensiz aralıklı üç boyutlu noktalardan elde edilmektedir [4]. Meteorolojik bilgi sistemlerinde, istasyonlardan toplanan sıcaklık ve basınç gibi veriler analiz edilerek verilerinin bölgelere göre değişimleri yine konturlama metotlarıyla gösterilmektedir. Tıp alanında manyetik rezonans görüntüleme (MRG) verilerinin daha anlaşılabilir olması içinde yine konturlama metotlarından faydalanılmaktadır.

Birçok alanda kullanılan konturlama metodunun genel amacı ise çok sayıda toplanan veriyi değerlendirirken daha kolay ve anlaşılabilir hale getirmektir. Bu toplanan verilerin

ortak noktası ise X, Y konumlarına ya da bir koordinat ağındaki X,Y koordinatlarına bağlı spesifik Z değerleridir. Bu Z değerleri çalışma alanına göre o konumdaki sıcaklık, basınç, yükseklik ya da bir sondaj alanındaki madenin o konumdaki yoğunluğu olabilir. Toplanan çok sayıdaki veri setlerini ön işlemden geçirip gösterimini yapabilmek için de bilgisayar yazılımlarına ihtiyaç duyulmaktadır. Her bir çalışma alanı için kullanılan yazılımlar, veri setlerinin ve gösterimlerinin aynı olmasına rağmen yazılımlardan alınan çıktılar farklılık göstermektedir. Bunun yanı sıra var olan yazılımlar konturlama için farklı amaçlarla kullanılabilmektedir. Örneğin, Surface isimli konturlama yazılımı sadece haritalama amacıyla tercih edilmektedir.

Bu çalışmada, herhangi bir alanda yapılacak çalışma için toplanan ham verileri ön işlemlerden geçirip konturlama metodu ile gösterilebilmek için bir yazılım geliştirilmiştir. Toplanan XYZ verileri ön işlemden geçirilirken, öncelikle bir koordinat ağı ya da veri düzlem ağı denilen grid oluşturulmuştur. Grid noktasının her bir düğüm noktası için bir XYZ verisi atanmış ve gridleme işlemi tamamlanmıştır. Veriler ön işlemden geçirilip gridleme tamamlandıktan sonra oluşturulan grid üzerine konturlama yapılarak toplanan verilerin kontur haritaları çıkartılmıştır.

Liang ve arkadaşları [5] çalışmalarında ters mesafe ağırlık enterpolasyonunu kullanarak ölçeklenebilir veri akış moturu yaklaşımında bulunmuşlardır. Çeşitli sensörlerden elde edilen bilgileri veri akış yöntemini kullanarak çevresel tehlike algılama, deprem, orman yangını ve radyasyon izleme gibi gerçek zamanlı uygulamalarda kullanılacak akış sorgusu yapısı tasarlamışlardır. Çalışmalarına veri akışları için Kriging hesaplamalarını uyarlama çabaları sarf edilmiş fakat hesaplama karmaşıklığı ve verim düşüklüğü nedeniyle mesafenin tersi yöntemini geliştirerek çalışmalarında kullanmışlardır. Mesafenin tersi yöntemi bu çalışmada da görüldüğü üzere büyük miktarlardaki veri setleri üzerinde hızlı sonuçlar elde edilmek için tercih edilmektedir.

Çalışmalarda farklı enterpolasyon yöntemleri kullanılarak gridleme işlemleri yapılsa da bu grid verilerinin yoğunlaştığı noktaları anlayabilmek ve verileri anlamlandırabilmek için grafiklere ihtiyaç duyulur. Kontur grafikleri de birçok alanda verileri daha kolay anlamlandırabilmemizi sağlarlar.

Menteşe [6] çalışmasında, karaciğer nakli ameliyatlarından önce damar bağlantılarının elde edilmesi için MRG, BT ve ultrason görüntülerini kullanmıştır. Bu görüntüler üzerinde karaciğer damar segmantasyonu işlemleri gerçekleştirmiştir. Bu işlemler

sonrasında elde ettikleri verilerle görüntüler üzerinde bölge tespiti, damar yapısını belirleme, otomatik sınıflandırma, lob hacimlerini belirleme, lob ayrıştırma işlemlerinde konturlama metodunu kullanmıştır.

Biltekin [7] çalışmasında, lokal ileri evre serviks kanserlerinin radyoterapi ile tedavi öncesinde planlama basamaklarında konturlamayı kullanmıştır. Hedef ve kritik organlara yönelik hasta üzerinde yüksek riskli ve orta riskli olmak üzere iki farklı hedef hacim tanımlaması yapılmıştır. Yüksek riskli hedef hacim tanımlaması için yapılan konturlamada eksternal radyoterapi (ERT) sonrası yanıt değerlendirme amacı ile çekilen manyetik rezonans (MR) görüntüleri ve yapılan ayrıntılı muayene bulguları veri olarak kullanılmıştır. Bu konturlama aracılığı ile hedef ve kritik organlar birbirinden ayırt edilerek radyoterapi ile tedavi sırasında belirsizliklerin mümkün olduğunca ortadan kaldırılması veya azaltılmasında büyük önem taşımaktadır.

Baykara ve Alemdaroğlu [8] çalışmalarında, yüksek hızlı İHA tasarımını ve en iyi aerodinamik performansı veren yüksek hızlı İHA'nın seçilebilmesi için çeşitli kuyruk ve kanat tasarımlarının performans testlerini gerçekleştirmişlerdir. Uçak dizaynında aerodinamik performansın anlaşılabilmesi ancak prototipler üretilerek rüzgar tünellerinde test edilerek anlaşılabilmektedir. Fakat her bir tasarım için ayrı prototip üretip rüzgar testini yapmak hem çok fazla zaman hem de çok fazla maliyetli olduğundan bunun yerine yazılım aracılığı ile İHA'ların farklı konfigürasyonlarının, kanat üzeri Mach sayılarını kontur grafikleriyle göstermişlerdir.

Özsavaş ve arkadaşları [9] çalışmalarında, bilgisayarlı tomografi görüntülerinden akciğer konturlamasının radyasyon tedavi planlamasının vazgeçilmez bir parçası olduğundan bahsetmiştir. Fakat geniş kanserli alanların bulunduğu durumlarda, akciğer ve çevreleyen yapılar arasında kontrastın düşük olması nedeniyle, klasik akciğer konturlamasının başarısız olabildiğinden bahsetmiştir. Çalışmalarında, radyoterapi planlamalarında kullanılmak üzere bilgisayarlı tomografi görüntülerinden akciğeri tam olarak konturlayabilen ve sonrasında akciğerlerdeki geniş kanserli alanları tespit edebilen tam otomatik bir yöntem önerilmektedir. Böylece elde ettikleri sonuçlar, ağır hesaplama yükünden kaçınan ve hızlandırılmış konturlamayı sağlayan yöntemin radyoterapi planlamada kullanabileceği gösterilmiştir.

Sındırgı ve arkadaşları [10] çalışmalarında, Aydın-Kuşadası çalışma alanında yeraltı sularını inceleyerek tuzluluk ve kirlilik oranları üzerine bir çalışma yapmışlardır. Bölgeye

ait sondaj kuyularından topladıkları örneklerin analizi sonucu tuzluluk ve kirlilik verilerini elde etmişlerdir. Elde edilen ayrıntılı sonuçları (kimyasal ve bakteriyolojik) bir arada değerlendirilerek bölgedeki tuzlanma ve kirlilik boyutunu bölge haritası üzerinde konturlama yaparak göstermişlerdir.

Aslan [11] çalışmasında, farklı bölgeler için hem sürekli küresel yer belirleme sistemi (GPS) verileri hem de yıllık ortalama GPS verileri kullanarak 3 farklı veri seti ile çalışmanın birinci kısmında, Türkiye ve çevresini kapsayan bölge için 7 adet sürekli ölçümleri bulunan GPS istasyonu ile bir GPS ağı oluşturmuştur. İkinci kısmında Türkiye ve yakın çevresi için 1988-1997 yılları arasında yapılan yıllık GPS ortalaması kullanılarak 32 adet GPS istasyonunu içeren ve üçüncü çalışma alanı Marmara Denizi ve yakın çevresini kapsayan GPS ağını 28 adet istasyon kullanarak oluşturmuştur. Elde edilen GPS ağı verileri sonuçları ile ilgili çalışma alanlarının asal gerilme haritalarını konturlama yöntemi ile göstermiştir.

Görüldüğü gibi konturlama yöntemi farklı alanda toplanan verilerin yoğunlaştığı alanları göstermek amacıyla kullanılmaktadır. Bu çalışmada, mesafenin tersi enterpolasyon yöntemi kullanılarak gridleme yapılmış, gridleme yöntemiyle elde edilen verilerden kontur grafiği çizebilen, farklı çalışma alanlarında kullanılabilir bir yazılım geliştirilmiştir.

2. SAYISAL ARAZİ MODELİ KAVRAMI

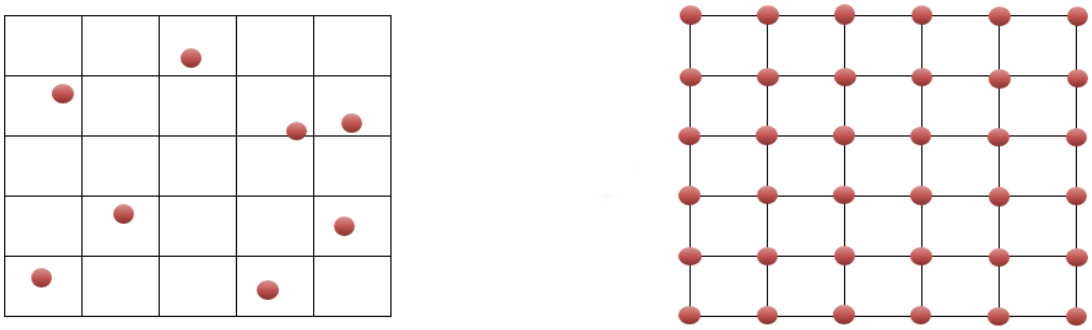
Geniş bir çalışma alanı olan sayısal arazi modeli kavramı, jeofizik haritalarında, meteorolojik bilgi sistemlerinde, tıp ve birçok alanda toplanan verilerin sayısal olarak temsilidir. Sayısal arazi modelleri, farklı çalışma alanları için toplanmış verilerin yoğunlaştığı veri kümelerini ifade etmek için kullanılır.

2.1. SAYISAL ARAZİ MODELLEME TEKNİKLERİ

Sayısal arazi modeli kurgulanırken hangi yöntemin kullanılacağına karar vermek oldukça önemlidir. Modelleme yaparken en kritik işlem çalışma alanındaki düğüm noktalarının doğru şekilde yerleştirilmesidir. Çalışma alanında grid (ızgaralama) ve üçgenleme en çok kullanılan yöntemlerdendir.

2.1.1. Grid Yöntemi

Araştırmalarda veriler toplanırken her zaman istenilen noktalardan düzenli veriler elde etmek mümkün olmayabilir. Örneğin; jeofiziksel bir harita için manyetik ya da sismik verilerin her 10 metrede bir toplanması hedeflenirken ölçüm yapılacak noktalara yerleştirilecek ölçüm cihazları arazi koşulları nedeniyle yerleştirilemeyebilir. Bu gibi durumlarda enterpolasyon yöntemleri, arazinin belirli noktalarında yapılan ölçümlerle arazinin ölçüm yapılamayan noktalarına tahmini değer atanması için yardımcı olacaktır. Enterpolasyon, bir çalışma alanının belirli noktalarından alınan ölçüm bilgilerinin, alanın ölçüm alınmayan diğer noktalarına tahmini değer atama işlemine denir. Şekil 2.1’ de görüldüğü gibi enterpolasyon yapılırken, toplanan üç boyutlu veriler aracılığı ile çalışma alanının tamamı gridlenmektedir. Bu grid çalışma alanının tamamını kapsayacak şekilde yapılmaktadır. Çalışma alanı bir sondaj alanı ise sondajlama noktalarının tamamını, bir sıcaklık dağılım haritası çıkarılacak ise sıcaklık ölçümlerinin yapıldığı alanın tamamını kapsayacak şekilde bir gridleme yapılır.

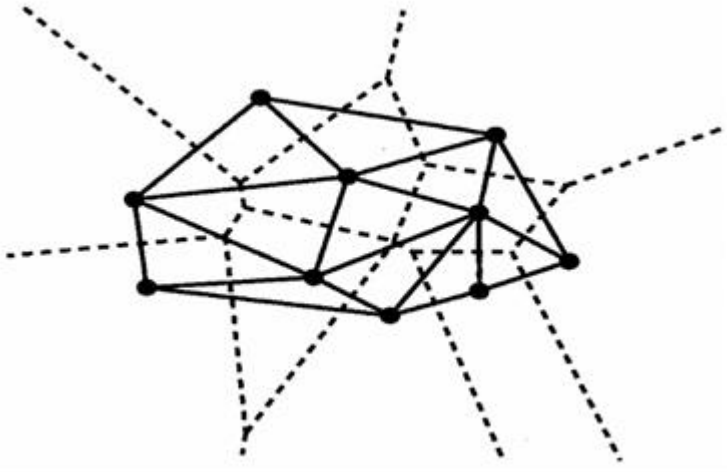


Şekil 2.1. Bilinen noktalardan yararlanılarak bilinmeyen noktalar üretilerek çalışma alanının gridlenmesi.

Gridleme yapılırken çeşitli enterpolasyon yöntemleri kullanılmaktadır. Çalışmalarda ki eksik değerlerin tahmin edilebilmesi için en çok mesafenin tersi yöntemi kullanılmaktadır [12], [13]. Bu çalışmada, gridleme yapılırken mesafenin tersi yöntemi kullanılmıştır. Bu yöntem, Sayısal Arazi Modellerinde Kullanılan Bazı Enterpolasyon Yöntemleri bölümünde açıklanmıştır.

2.1.2. Üçgenleme Yöntemi

Arazi modellemesi yapılırken yüzey tek bir fonksiyon yardımıyla sürekli olarak gösterilebileceği gibi üçgen, kare gibi geometrik şekillere bölünerek kısımlar halinde de gösterilebilmektedir. Eş karelere bölündüğünde gridleme yönteminde olduğu gibi çalışma alanı ızgaralanmaktadır. Çalışma alanında referans noktalarının düzensiz dağılım göstermesine bağlı olarak, arazi modellemesinde referans noktalarının enterpolasyon yöntemleri kullanılarak üçgenleme için düğüm noktaları oluşturulmaktadır [14]. Şekil 2.2' deki gibi arazi modellemesinde oldukça sık kullanılan üçgenleme yöntemi, Voronoi Diyagramı ve Delunay Üçgenlemesi üzerine kurulmuştur.



Şekil 2.2. Voronoi diyagramı ve Delunay üçgenlemesi [15].

Düzlemde yer alan sonlu nokta kümesine ait herhangi bir noktaya, kümedeki diğer noktalardan daha yakın konumda bulunan düzlem noktalarının geometrik yerine, o noktanın "Voronoi Çokgeni" denilmektedir. Kümedeki tüm noktaların Voronoi çokgenlerinin birleşimi, o kümenin Voronoi diagramını oluşturmaktadır [16].

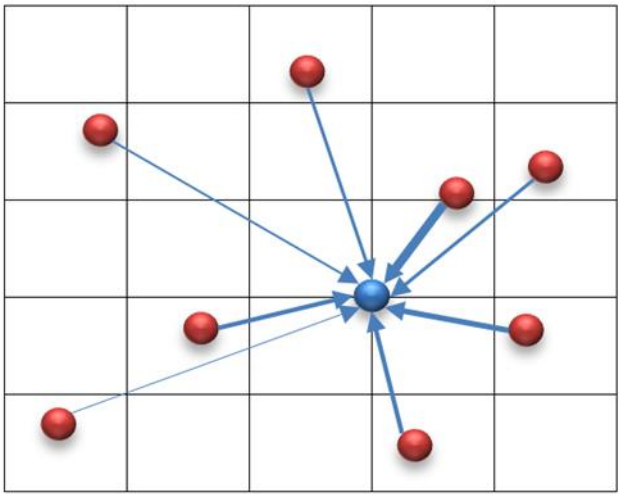
2.2. SAYISAL ARAZİ MODELLERİNDE KULLANILAN BAZI ENTERPOLASYON YÖNTEMLERİ

Düzgün olmayan yüzeyler ya da çalışma alanlarının matematiksel olarak ifadesi oldukça zordur. Çalışma alanı ya da yüzeyin tam olarak ifade edilebilmesi için tüm noktaların bilinmesi gerekir ki bu da pek çok çalışma alanı için mümkün değildir [17]. Sayısal arazi modellerinde kullanılan enterpolasyon yöntemleri de, çalışma alanlarındaki bilinen noktalardan bilinmeyen noktaları matematiksel olarak elde etmek amacıyla kullanılmaktadır.

Bu çalışmada sayısal arazi modelleme yöntemlerinden mesafenin tersi yöntemi kullanılarak uygulama geliştirilmiştir. Diğer enterpolasyon yöntemleri de kısaca anlatılmaya çalışılmıştır.

2.2.1. Mesafenin Tersi Yöntemi

Çalışma alanındaki bilinen noktalardan bilinmeyen grid noktaları elde etmek amacıyla kullanılmaktadır. Bilinmeyen bir grid noktasının değeri tahmin edilirken, bilinen tüm ölçüm noktalarının değerlerinin ardışık olarak o noktaya etkisi hesaplanmaktadır. Hesaplamada grid noktasına en yakın olan ölçüm noktasının değerinin etkisi o noktaya en fazla olurken tahmin edilecek grid noktasından uzaklaştıkça hesaba katılacak ölçüm noktalarının ağırlığı azalmaktadır [18]. Şekil 2.3' de çizgi kalınlığı bilinen bir noktanın bilinmeyen bir noktaya etki şiddetini göstermektedir. Mesafe açıldıkça çizgi incelmekte, mesafe daraldıkça çizgi kalınlaşmakta yani daha etkili olduğunu ifade etmektedir.

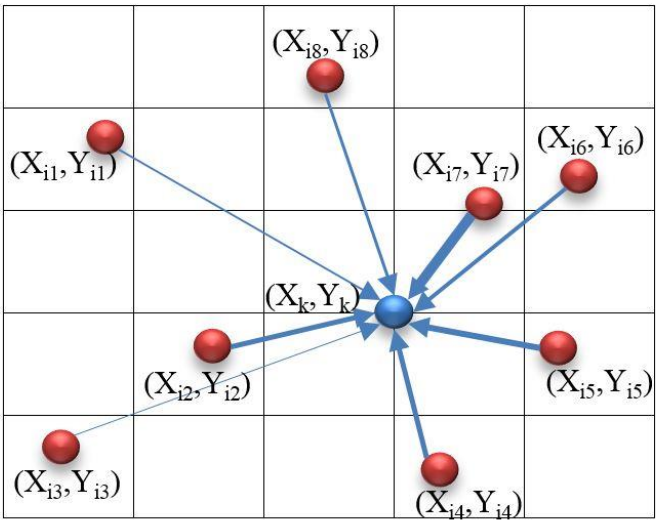


Şekil 2.3. Ölçüm noktalarının grid düğüm noktalarına uzaklığa bağlı etkileri.

Mesafenin tersi yöntemi, Şekil 2.1' deki gibi gridleme yapılırken, grid noktasındaki tahmin edilecek değerin hesaplanabilmesi için çevredeki bilinen ölçüm noktalarının değerini kullanan bir enterpolasyon yöntemidir. Bu hesaplama yöntemi Denklem (2.1) ve Denklem (2.2)' deki gibi ifade edilmektedir.

$$D_{ik} = \sqrt{(X_k - X_i)^2 + (Y_k - Y_i)^2} \tag{2.1}$$

Eşitlikte, X_k, Y_k grid düğüm noktasının koordinatlarını, X_i, Y_i ölçüm değeri bilinen noktanın koordinatlarını belirtmektedir. D_{ik} tahmin edilecek grid noktasının, ölçüm değeri bilinen her bir noktaya olan uzaklığını belirtmektedir. Bu uzaklığı bulmak için Şekil 2.4'de gösterildiği gibi grid düğüm noktası (X_k , Y_k) ile ölçüm noktası (X_i, Y_i) arasındaki doğrusal uzaklık hesaplanmaktadır.



Şekil 2.4. Düğüm noktası ile ölçüm noktalarının grid üzerinde gösterimi.

Denklem (2.1)'deki hesaplamalar tamamlandıktan sonra Denklem (2.2)'de yerine koyularak Z_k değeri hesaplanır.

$$Z_k = \frac{\sum_{i=1}^{n}(\frac{Z_i}{D_{ik}})}{\sum_{n=1}^{n}(\frac{1}{D_{ik}})} \tag{2.2}$$

Eşitlikte, Z_k tahmin edilecek grid noktasını belirtirken Z_i ise ölçüm değeri bilinen grid noktasının çevresindeki değerleri belirtmektedir.

2.2.2. Kriging Yöntemi

Bu yöntem birçok alanda kullanılan geoistatiksel bir enterpolasyon yöntemidir. Bu metod, D.G. Krige isimli Güney Afrikalı bir maden mühendisi tarafından geliştirilmiş ve mühendisin adını almıştır [19]. Bu enterpolasyon yöntemi Denklem (2.3)'de gösterildiği gibi, dayanak noktalarının ağırlıklı ortalaması alınarak hesaplanmaktadır.

$$Z_k = \sum_{i=1}^{n} D_i Z_i \tag{2.3}$$

Eşitlikte;

Z_k : k kontasının aranacak değerini,

D_i : Z_k değeri aranırken her bir Z_i'ye atanacak ağırlıkları,

Z_i : Z_k değeri aranırken kullanılan noktaların değişim değerlerini göstermektedir.

2.2.3. Minimum Eğrilik Yöntemi

Minimum Eğrilik Yöntemi birçok veri setleri için hızlı şekilde doğal, yumuşak yüzeyler üretir fakat veri bulunmayan alanlarda oldukça yüksek yapay yükseltiler oluşturabilmektedir. Çalışma alanı içinde ve sınır konturlarında düzgünleştirme oranlarının kontrol edilmesini sağlamaktadır. Minimum Eğri Yöntemi, Z verilerinin aralığının (en küçüğü ve en büyüğü) ötesinde değerler çıkarabilmektedir.

2.2.4. Değiştirilmiş Shepard's Yöntemi

Değiştirilmiş Shepard's Yöntemi, mesafenin tersi yöntemine benzer fakat mesafenin yönteminde olduğu gibi yuvarlak-boğa gözü konturlamalarına yatkın olmadığından genellikle kontur yumuşatma-düzeltme işlemlerinde kullanılmaktadır. Bu yöntem veri setindeki Z grid değerinin dışında ve tahmini Z değerleri üretebilmektedir.

2.2.5. Doğal Komşu Yöntemi

Doğal Komşu Yöntemi, veri setlerindeki yoğun verilerin olduğu bölgelerden daha az verilerin olduğu alanlar için daha iyi konturlar üretmek için kullanılmaktadır. Gridleme yaparken çalışma alanındaki veri içermeyen alanlar için veri üretilememektedir. Bu yöntemde veri setindeki Z grid değerinin dışına çıkılamamakta ve tahmini Z değeri üretilememektedir.

2.2.6. En Yakın Komşu Yöntemi

En Yakın Komşu Yöntemi, gridleme yaparken, grid alanındaki eksik noktaları XYZ veri setlerinden yararlanıp tahmini değerlerle grid üzerindeki eksik gözlem ya da ölçüm noktalarını doldurmak için kullanılmaktadır. Bu yöntemde veri setindeki Z grid değerinin dışına çıkılamamakta ve tahmini Z değeri üretilememektedir.

2.2.7. Polinomal Fonksiyon Yöntemi

Bu yöntemle arazi yüzeyi tek bir fonksiyonla ifade edilmektedir. Arazideki dayanak noktalarının oluşturduğu yüzey n'inci dereceden bir polinomla genel olarak Denklem (2.4)' deki gibi ifade edilmektedir [20].

$$Z(x,y) = \sum_{k=0}^{n} \sum_{i=0}^{k} j = k - i\, a_{ij} x^i y^i \qquad (2.4)$$

Eşitlikte; a_{ij}, denklemin bilinmeyen katsayılarını, n, yüzeyin derecesini, i ve j (x,y)

koordinatlarının üssü olan pozitif tamsayıları göstermektedir.

2.2.8. Radyal Bazal Fonksiyon Yöntemi

Radyal bazal fonksiyonları dayanak noktalı enterpolasyon metodları gibidir. Multikuadrik yöntemin, verilen noktalardan geçen bir yüzey üretmek için iyi olduğu düşünülmektedir [21]. Bu yöntem çeşitli çalışma alanlarındaki veri setlerinden Kriging yöntemine benzer sonuçlar ürettiği ve Kriging Yöntemine kıyasla daha iyi enterpolasyon yapıldığından oldukça fazla kullanılmaktadır.

2.2.9. Doğrusal Enterpolasyon İle Üçgenleme Yöntemi

Doğrusal Enterpolasyon ile Üçgenleme Yönteminde küçük veri setleri ile çalışılırsa grid noktaları arasında belirgin üçgen yüzeyler oldukça hızlı şekilde üretilmektedir. Bu yöntemde veri setindeki Z grid değerinin dışına çıkılamamakta ve tahmini Z değeri üretilememektedir.

2.2.10. Hareketli Ortalama Yöntemi

Hareketli Ortalama Yöntemi, büyük hatta çok büyük kirli veri setlerinde oldukça hızlı büyük ve orta derecede varyasyonlar çıkarmaktadır. Bu yöntem büyük veri setlerinde En Yakın Komşu Yöntemine alternatif olarak kullanılmaktadır.

2.2.11. Veri Metrik Yöntemi

Veri Metrikleri Yöntemi, verilerle ilgili bilgi gridleri oluşturmak için kullanılmaktadır. Çalışma alanı için oluşturulan griddeki tüm Z değerlerinin sayısı ve değerlerinin bilindiği gridlemelerde kullanılmaktadır.

2.2.12. Yerel Polinom Yöntemi

Yerel Polinom Yöntemi, çalışma alanındaki bölgesel veri setlerini kullanarak bölgesel pürüzsüzleştirme ya da daha düzgün kontur çizgileri üretmede kullanılmaktadır. Bu yöntemin hesaplama hızı veri setinin boyutuna çok fazla bağımlı olmamaktadır.

2.3. C# GRAFİK İŞLEMLERİ

Bilimsel, matematik ve mühendislik alanlarında ki rapor ve sunumlarda grafik verilerine oldukça fazla ihtiyaç duyulmaktadır. Çalışmalara eklenen tablo ve grafikler ile raporlar görsel ve en önemlisi anlaşılması kolay hale getirilmektedir. Microsoft' un nesneye dayalı

C Sharp (C#) programlama dili kullanılarak hem hesaplamalar yapıp veri setleri üretebilmekte, hem de üretilen bu veri setlerinden grafik aygıt arabirimi (GDI) kütüphanesini kullanılarak tablo ve grafikler oluşturulmaktadır.

Microsoft Visual C#, Microsoft .Net Framework kullanarak nesneye dayalı, hesaplamalar, haberleşme ve iletişim uygulamaları vb. geliştirmek amacıyla güçlü fakat basit bir programlama dili olarak tasarlanmıştır [22]. C# programlama dili, temel programlama yeteneklerine sahip kullanıcıların gelişmiş grafik ve tablolar üretmesini sağlayan çok yönlü, esnek bir grafiksel kullanıcı arayüzüne (GUI) sahiptir. C# ile geliştirilen grafik ve tabloların gelişmişlik ve karmaşıklık düzeyi ihtiyaca göre şekillenmektedir.

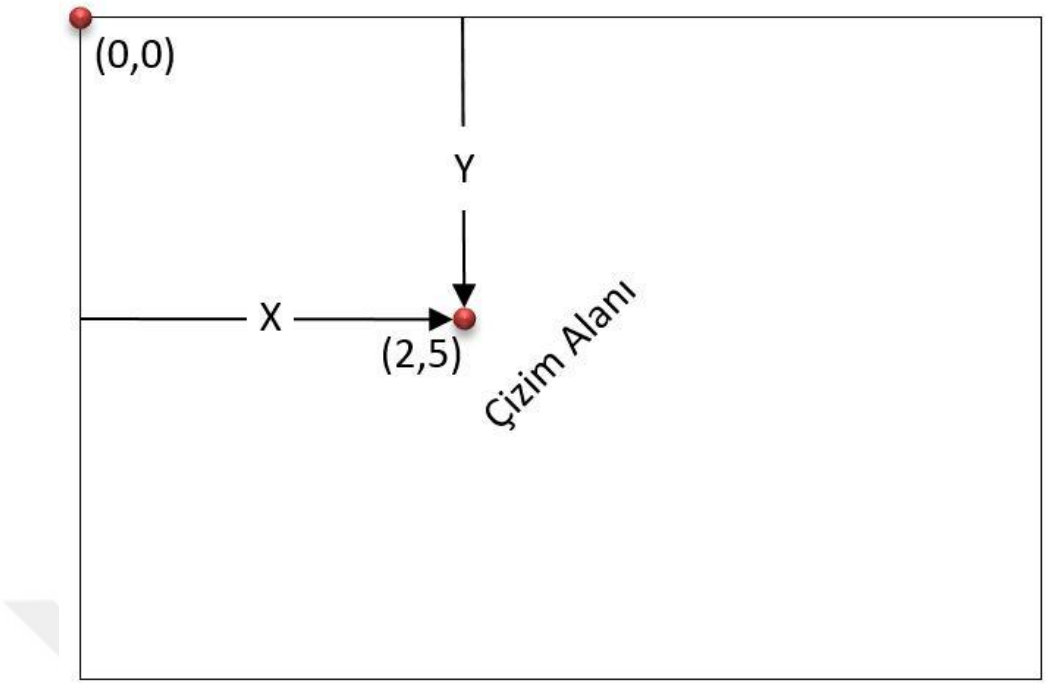
2.3.1. Koordinat Sistemi

Bir grafik nesnesi oluştururken grafik nesnesinin veya çizimin nerede oluşturulacağına karar verdikten sonra bunu yapabilmek için C#'ta grafik nesnelerinin koordinat sisteminde nasıl ölçülendirildiğinin bilinmesi gerekmektedir. Her bir Windows formunda ya da kontrol aracında bir X ve Y koordinatı bulunmaktadır.

C# ve GDI kütüphanesinde üç temel 2D koordinat sistemi mevcuttur. Bunlar dünya, sayfa ve cihaz koordinat sistemleri olarak adlandırılmaktadır.

- Dünya koordinat sistemi, belirli bir grafik dünyasını modellemek için kullanılan C#' taki metotlardandır.
- Sayfa koordinat sistemi, bir çizim yüzeyini tıpkı bir sayfa gibi ölçeklendirerek form veya kontrollerde kullanılmaktadır.
- Cihaz koordinat sistemi, fiziksel cihazlarda (ekran vb.) çizgi çizmek için ekranın ölçülerini kullanarak (x_1,y_1) noktasından (x_2,y_2) noktasına çizgi çizmek için kullanılmaktadır.

Genel olarak koordinat sistemi, dünyada mesafeyi ölçmek için kullanılan birimin cihaz ya da form elamanına göre uyarlanmasıyla oluşmaktadır. Bir nesne oluştururken ya da çizerken doğrudan dünyadaki koordinat sistemi kullanılamamaktadır. Ekranda bir grafik ya da nesne çizmeden önce koordinat dönüşümleri (Dünya-Sayfa, Dünya-Cihaz gibi) yapılması gerekmektedir. Her üç koordinat sisteminin de Şekil 2.5'de olduğu gibi orijin noktası (0,0) varsayılan olarak çizim noktasının sol üst köşesidir. Örneğin Şekil 2.5'de verilen koordinat sistemindeki nokta, (X,Y) noktasını işaret etmektedir.



Şekil 2.5. Koordinat sistemi ve başlangıç noktası.

Koordinat sistemlerinde kullanılan birimler farklılıklar gösterebilir. Grafik kütüphanesinin *GraphicsUnit* metodu çalışma alanının varsayılan birimi pikseldir, ihtiyaç duyulduğu durumda bu birim inç veya milimetre olarak değiştirebilir.

2.3.2. Pencere ve Görünüm

Genellikle uygulamalarda grafikler ya da çizimler belirli bölümlerde ya da alanlarda kullanılmaktadır. Ekranın tamamı grafik ya da çizimler için kullanılmamaktadır. Bunun için çalışma formunun belirli bir alanı pencere (Window) olarak ayrılmaktadır. Bu alanda çizgiler çizmek, şekiller oluşturmak için bir koordinat sistemi dünyası oluşturulmakta buna da görüntü alanı (Viewport) denmektedir. Bir grafik nesnesi oluşturulacağında, grafik penceresinde koordinat sistemi belirlenir ve şekil nerede oluşturulacaksa karar verilir ve böylece görüntüleme alanında belirlenen koordinatlara şekil ya da çizgi yerleştirilmektedir.

Grafik alanında nesnelerin boyutunu ve konumunu değiştirmek için Viewport (görüntü alanı) kullanılmaktadır. Görüntü alanının penceredeki boyutu değiştirildiğinde, nesnelerin boyutları görüntü alanına göre oranlanacağından çizilen çizgi ya da şekillerin boyutu değişmektedir. Bu özellik, aynı zamanda o nesne ya da çizgiye yakınlaşma-uzaklaşma efekti olarak da kullanılmaktadır.

2.3.3. Kalem ve Fırça

Grafik nesneleri uygulama yazılımı ile görüntüleme aygıtı arasında bir arayüz olarak görülebilir. Uygulamada bir grafik nesnesi oluşturduktan sonra bir çizgi çizebilir ya da bir şekil oluşturulabilir. Ancak grafik kütüphanesi şekiller oluşturmak ve çizgiler çizmek için bir platform sağlasa bile hala çizim yapmak için bir araca ihtiyaç duyulmaktadır. Burada da en çok kullanılan araçlar, kalem ve fırça (Pen-Brush) devreye girmektedir. Şekiller, çizgiler, eğriler ve konturları çizmek için kalem kullanılmaktadır. Renkler ve desenlerle şekilleri doldurmak için ise fırça kullanılmaktadır.

2.3.4. Temel Grafik İşlemleri

Windows Form'ları bir C# uygulamasındaki en büyük kullanıcı arayüzü birimidir. C# uygulamalarında grafikler uygulama içerisindeki en büyük arabirim olan Windows Form'lara çizdirilebileceği gibi form üzerine diğer denetimler eklenerek bu eklenen denetimler üzerine de çizdirilebilir.

Şekil 2.5'deki gibi (X,Y) koordinatları (2,5) olan bir çizgi içeren grafik nesnesinin Windows Form üzerine çizdirmek için Şekil 2.6'de verilen kod parçacığı kullanılmıştır.

```
Graphics grafik = this.CreateGraphics();
Pen kalem = new Pen(Color.Black, 5 );
grafik.DrawLine(kalem, 0, 0, 2, 5);
grafik.Dispose();
```

Şekil 2.6. Windows form üzerine düz çizgi çizdirme.

Şekil 2.5'de birinci satırda farklı bir sınıf içerisindeki metoda erişebilmek için Graphics sınıfından grafik adında bir nesne üretilmiştir. İkinci satırda Pen sınıfından kalem isimli nesne üretilmiştir. Kalem nesnesi grafik alanına çizilen çizginin özelliklerini tanımlar. Nesne tanımlanırken Color.Black ve 5 parametreleri ile kalem nesnesinin üreteceği çizginin siyah renkli ve 5 birim kalınlığında olacağı belirtilmiştir. Üçüncü satırda, grafik isminde oluşturulan nesnenin DrawlLine metodunu çağrılmıştır. Çizginin renk ve kalınlık bilgisini kalem nesnesinden aktararak, başlangıç ve bitiş noktalarını ise sırasıyla X_1, Y_1, X_2, Y_2 şeklinde parametreler göndererek form üzerine çizgi çizdirilmiştir. En son satırda Graphics sınıfından üretilen grafik nesnesi işlemlerini tamamladığından hafızada yer kaplamaması için Dispose metodu ile kapatılmıştır.

2.3.4.1. *Noktalar*

C# GDI kütüphanesinde iki nokta (Point) yapısı bulunmaktadır. Bunlar Point ve PointF dir. Point yapısı iki boyutlu düzlemdeki bir noktayı tanımlayan X,Y tamsayı koordinatlarını temsil etmektedir. PointF ise Point yapısına benzer fakat tamsayı yerine reel sayıları kullanmaktadır. Şekil 2.7'da Point sınıfından nokta nesnesi üretilerek tanımlama yapılmıştır.

```
Point nokta = new Point();
```

Şekil 2.7. Nokta nesnesi tanımlama.

2.3.4.2. *Çizgiler ve Eğriler*

Grafik kütüphanelerindeki çiziciler, çizgi ve eğri nesnelerini barındırmaktadır. DrawLine yöntemi ile koordinat sisteminde belirtilen iki nokta arasına düz bir çizgi Şekil 2.8' deki gibi oluşturulmaktadır.

```
Graphics grafik = this.CreateGraphics();
Pen kalem = new Pen(Color.Black, 5 );
grafik.DrawLine(kalem, x1, y1, x2, y2);
grafik.Dispose();
```

Şekil 2.8. Noktalar arası çizgi çizdirme.

Şekil 2.8'de Graphics sınıfından grafik nesnesi, Pen sınıfından kalem nesnesi türetilmiştir. Oluşturulan grafik nesnesinin DrawLine metoduna çizgi rengini, kalınlığını ve çizginin başlangıç bitiş noktalarını belirten parametreler gönderilerek düz bir çizgi çizdirilmiştir. Şekil 2.8'de (x_1,y_1) çizginin koordinat sistemindeki başlangıç noktasını ve (x_2,y_2) ise çizginin bitiş noktasını belirtmektedir. Başlangıç ve bitiş noktaları Point yapısı kullanılarak Şekil 2.9'deki gibi tanımlanabilir:

```
Graphics grafik = this.CreateGraphics();
Pen kalem = new Pen(Color.Black, 5 );
Point nokta1 = new Point(x1, y1);
Point nokta2 = new Point(x2, y2);
grafik.DrawLine(kalem, nokta1, nokta2);
grafik.Dispose();
```

Şekil 2.9. Başlangıç ve bitiş noktaları arasına eğri çizdirme.

Şekil 2.9'de Graphics sınıfından grafik nesnesi, Pen sınıfından kalem nesnesi, Point sınıfından nokta1 ve nokta2 nesneleri türetilmiştir. DrawLine metoduna çizilecek çizginin

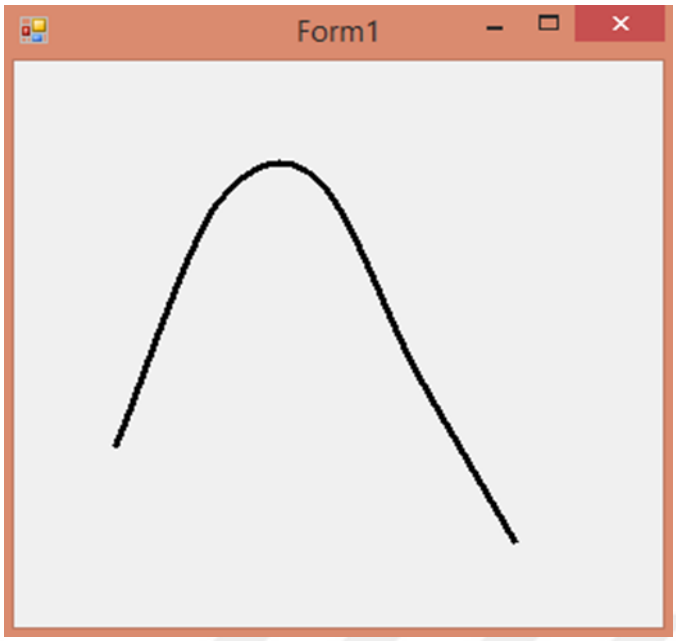
başlangıç ve bitiş koordinatları tek tek girilmek yerine nokta1 ve nokta2 nesneleri kullanılarak aktarılmıştır.

DrawCurve yöntemi ile bir nokta dizisindeki her noktadan geçen ve tüm noktaları birleştirecek şekilde birbirine bağlayan bir eğri oluşturulmaktadır. Bu noktalar Point yapısı kullanılarak aşağıdaki gibi oluşturulabilir:

```
Graphics grafik = this.CreateGraphics();
Pen kalem = new Pen(Color.Black, 5 );
Point nokta1 = new Point(50, 200);
Point nokta2 = new Point(100, 75);
Point nokta3 = new Point(150, 60);
Point nokta4 = new Point(200, 160);
Point nokta5 = new Point(250, 250);
Point[] noktalar = { nokta1, nokta2, nokta3, nokta4, nokta5 };
grafik.DrawCurve(kalem, noktalar);
grafik.Dispose();
```

Şekil 2.10. Noktalar kümesi oluşturma.

Şekil 2.10’ da Graphics sınıfından grafik nesnesi, Pen sınıfından kalem nesnesi, Point sınıfından nokta1, nokta2, nokta3, nokta4 ve nokta5 nesneleri türetilmiştir. Bir sonraki satırda Point sınıfı kullanılarak noktalar isminde dizi nesnesi üretilmiştir. Bu dizi nesnesine ait değerler nokta1, nokta2, nokta3, nokta4 ve nokta5 nesnelerinden aktarılmıştır. DrawCurve metoduna çizgi özellikleri için kalem, çizilecek eğrinin koordinatları ise noktalar nesnelerine ait değerler parametre olarak aktarılmıştır. Şekil 2.10’da ki örnek kodda belirtilen noktalar kümesi kullanılarak oluşturulan eğrinin grafiği Şekil 2.11’deki gibidir:



Şekil 2.11. Form üzerinde noktalar arası çizilen eğri.

2.3.5. Dynamic Data Display Kütüphanesi

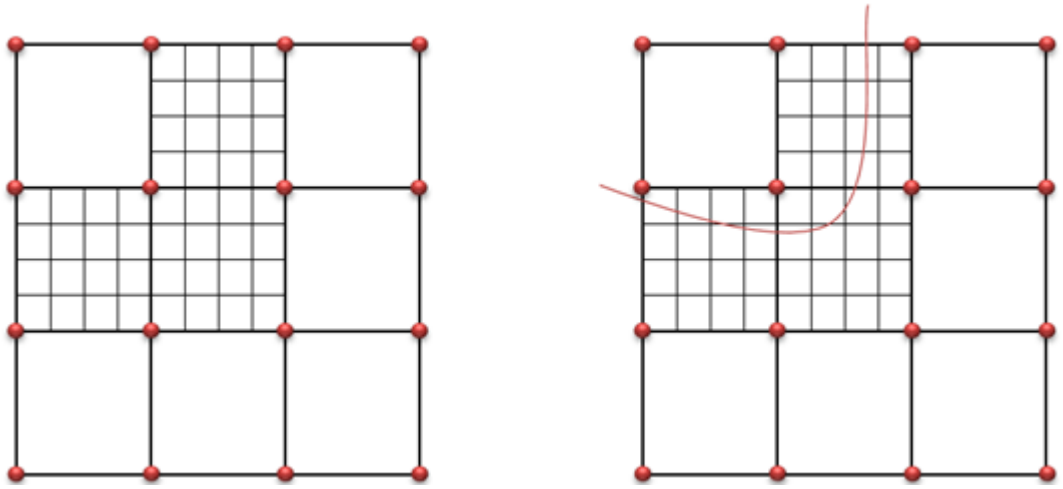
Microsoft tarafından, özellikle bilimsel uygulamalarda kullanılması amacıyla, çalışmalarda üretilen verilerden kolay ve hızlı bir şekilde çizgi grafikleri, kabarcık grafikleri, ısı haritaları ve diğer karmaşık 2D grafikler oluşturmak için Windows gösterim altyapısı (WPF) ve Silverlight kontrolleri kullanılarak hazırlanmış dinamik veri görüntüleme kütüphanesidir. Bu kütüphane .Net GDI kütüphanelerinden farklı olarak WPF ve Silverlight teknolojilerini kullandığından işlemleri merkezi işlemci birimi (CPU) yerine grafik işleme biriminde (GPU) yaparak gerçek zamanlı grafiklerde daha fazla görsellik ve performans sağlamaktadır.

Bu çalışmada da Dynamic Data Display kütüphanesi kullanılmıştır. Böylece çok büyük miktarlarda veriler için grafikler oluşturulurken CPU yerine GPU kullanıldığından normalden çok daha kısa sürede veri grafikleri elde edilmektedir.

2.4. KONTURLAMA

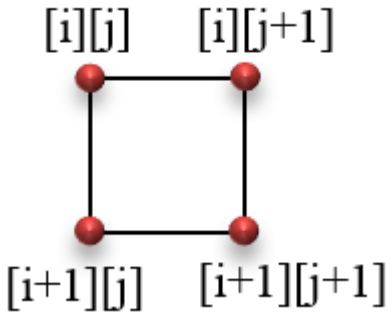
Kontur bir nesnenin, yüzeyin ya da bir verinin belirgin bir şekilde görünmesi sağlayan çizgi ya da çizgiler topluluğu olarak nitelendirilmektedir. Kontur çizgileri kullanıldığı alanlara göre adlandırılmaktadırlar. Örneğin; eş yükselti eğrileri gibi. Kontur çizgileri eğriler olarak adlandırılıp, eğriler olarak görülse de aslında küçük çizgi parçacıklarından oluşmaktadırlar. Tıpkı bir doğruyu oluşturan sonsuz noktalar gibi. Ancak Şekil 2.12'de

görüldüğü gibi kontur çizgisini oluşturan her bir çizgi parçacığı düzdür.



Şekil 2.12. Kontur eğrisini oluşturan düz çizgi parçacıkları.

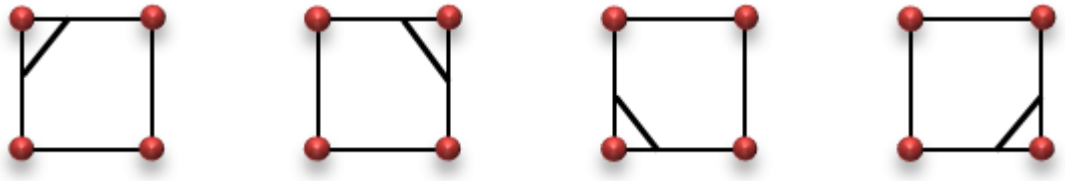
Kontur çizgileri çizilirken gridleme sırasında grid düğüm noktalarına atanan veri değerleri kullanılmaktadır. İki boyutlu dizi olarak gridlenmiş her XY ikilisi için bir Z değeri mesafenin tersi enterpolasyon yöntemiyle atanmıştır. Bu değerlerden XY grid üzerindeki koordinatları, Z değeri ise grid düğüm noktasının ağırlığını belirtmektedir. Kontur çizimine başlandığında Şekil 2.13'deki gibi gridlenmiş alanlar küçük karelere ayrılmaktadır.



Şekil 2.13. İki boyutlu diziden küçük kareler elde edilmesi.

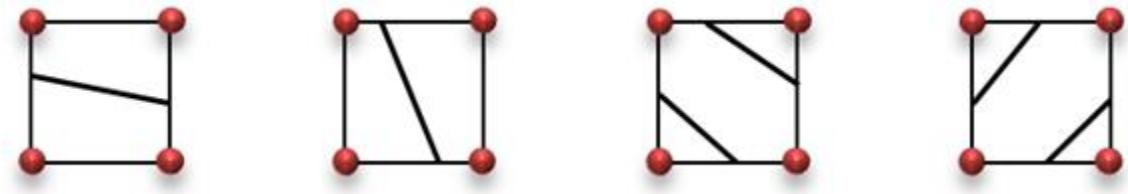
Şekil 2.13'deki gibi ayrılan her bir karenin kırmızı köşe noktaları iki boyutlu bir XY ikilisini ve bu ikiliye ait bir Z değerini belirtmektedir. Kontur çizgisinin seçilen kareden geçip geçmeyeceği kontur çizgisinin taşıdığı değer ve seçilen karenin köşe noktalarındaki değerlere bağlıdır. Seçilen karenin köşe noktalarındaki Z değeri kontur çizgisinin değerinden büyük ise bu kontur çizgisinin seçilen kareden geçeceği kesinleşmiştir. Kontur çizgisinin seçilen kareden geçeceği kesinleştikten sonra bu çizginin karenin hangi köşesinden girip hangi köşesinden çıkacağı belirlenmektedir. Kontur çizgisinin ağırlık

değeri ile kare köşesindeki Z değerleri tek tek karşılaştırılmaktadır. Seçilen karenin köşe değeri kontur çizgisinin değerinden büyük ise o köşe kontur çizgisiyle kesilerek kareden ayrılmaktadır.



Şekil 2.14. Kontur çizgisinin geçeceği kenarların belirlenmesi.

İşlem yapılan karede her zaman tek grid noktası kontur çizgisinin değerinden büyük olmayabilir. Eğer Şekil 2.14'den farklı olarak kontur çizgisinin değerinden büyük iki değer var ise bu iki değeri ayıracak şekilde kontur çizgisi çekilmektedir.



Şekil 2.15. Eş değer köşelerin kontur çizgisiyle ayrılması.

Kontur çizgisinin sahip olduğu ağırlık değerinden büyük olan tek bir grid noktası var ise Şekil 2.14'de olduğu gibi o köşe yalnız bırakılacak şekilde ayırılmaktadır. Eğer grid noktalarından ikisi büyük ise Şekil 2.15'de olduğu gibi büyük olan iki köşe kareden ayırılmaktadır. Grid düğüm noktalarının üçünün de değeri kontur çizgisinden büyük ise üç köşe diğer köşeden kontur çizgisi ile kesilerek ayrıldığında Şekil 2.14'deki durum oluşur. Bu durumların hiç birisi gerçekleşmezse, yani karenin düğüm noktalarının hepsi kontur çizgisinin değerinden büyük ise bu kareden kontur çizgisinin geçmeyeceği anlaşılmaktadır.

Bu işlem basamakları, Şekil 2.13'deki gibi gridlenmiş alanın karelere ayrılması ve her bir karenin tek tek bu işlem basamaklarından geçirilmesiyle tekrarlanmaktadır. Tekrarlanan bu işlemler sonucunda işlem gören her kare yan yana geldiğinde birleşen kontur çizgileri kontur eğrilerini, kontur eğrileri de kontur grafiklerini ya da haritalarını meydana getirmektedir.

3. MATERYAL VE YÖNTEM

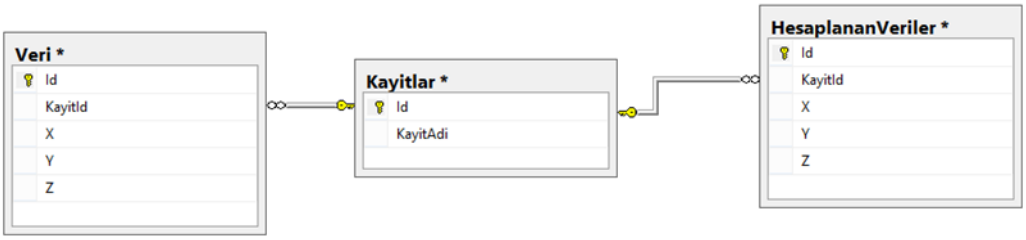
Çalışma kapsamında geliştirilen uygulama iki bölümden oluşmaktadır. Birinci bölümde bir veri tabanı tasarlanarak yazılım tarafından üretilen tüm verilerin saklanması amaçlanmıştır. İkinci bölümünde, nesneye dayalı metodolojiyle C# programlama dili kullanılarak yazılım geliştirilmiştir. Geliştirilen yazılım aracılığıyla mesafenin tersi yöntemi kullanılarak veriler enterpole edilmekte ve bu veriler kullanılarak kontur grafikleri çizdirilmektedir.

Çalışma kapsamında karşılaşılan sınırlılıklara sonuçlar ve öneriler bölümünde yer verilmiştir.

3.1. YAZILIM GELİŞTİRME AŞAMALARI

3.1.1. VERİ TABANI TASARIMI

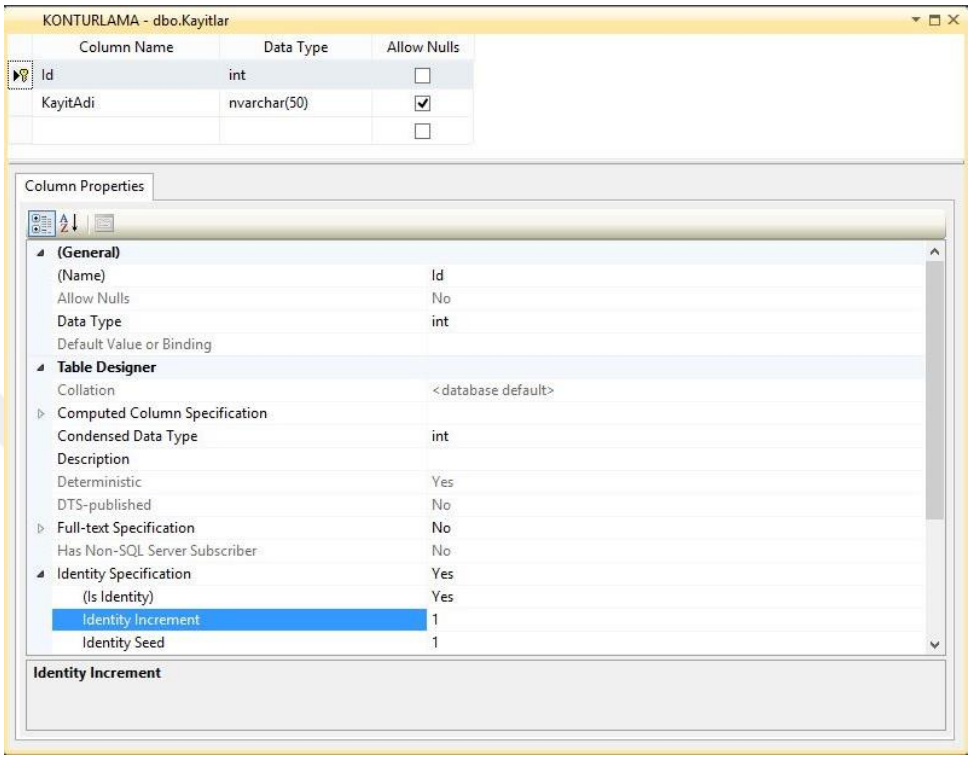
Geliştirilen yazılıma ait tüm veriler MSSQL Veri Tabanı kullanılarak saklanmaktadır. Tasarlanan veri tabanında üç adet tablo oluşturulmuştur. Bu tablolarda çalışma dosyaları, yüklenen veriler ve gridlenen veriler depolanmaktadır. Veri tabanı tablo tasarımı Şekil 3.1'deki gibidir.



Şekil 3.1. Veri tabanı tasarım diyagramı.

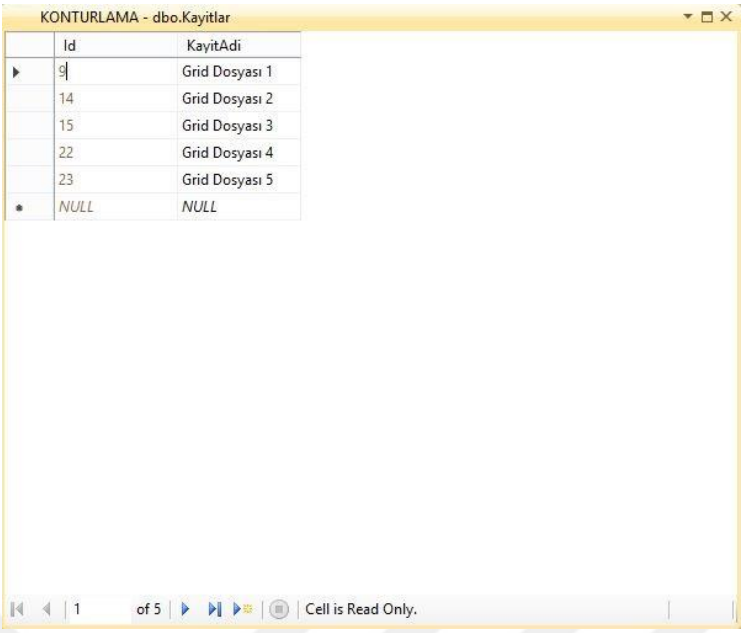
Şekil 3.1' de yer alan Kayıtlar adındaki tabloda, uygulama üzerinde oluşturulacak her bir çalışmanın Adı ve Id' si tutulmaktadır. Veri adındaki tabloda, uygulamaya dışarıdan yüklenen veriler tutulmaktadır. HesaplananVeriler adındaki tabloda ise Veri tablosundaki kayıtlar kullanılarak yapılan gridleme işleminin sonuçları tutulmaktadır. Kontur grafikleri çizilirken HesaplananVeriler tablosundaki kayıtlar kullanılmaktadır.

Uygulama üzerindeki tüm çalışmalar Kayitlar adındaki tabloda tutulmaktadır. Çalışmalara ait veriler bu tablodaki kayit adı ve id ile eşleşerek diğer tablolardan çağrılmaktadır. Kayitlar tablosu Şekil 3.2'de görüldüğü gibi iki adet veri alanından oluşmaktadır.



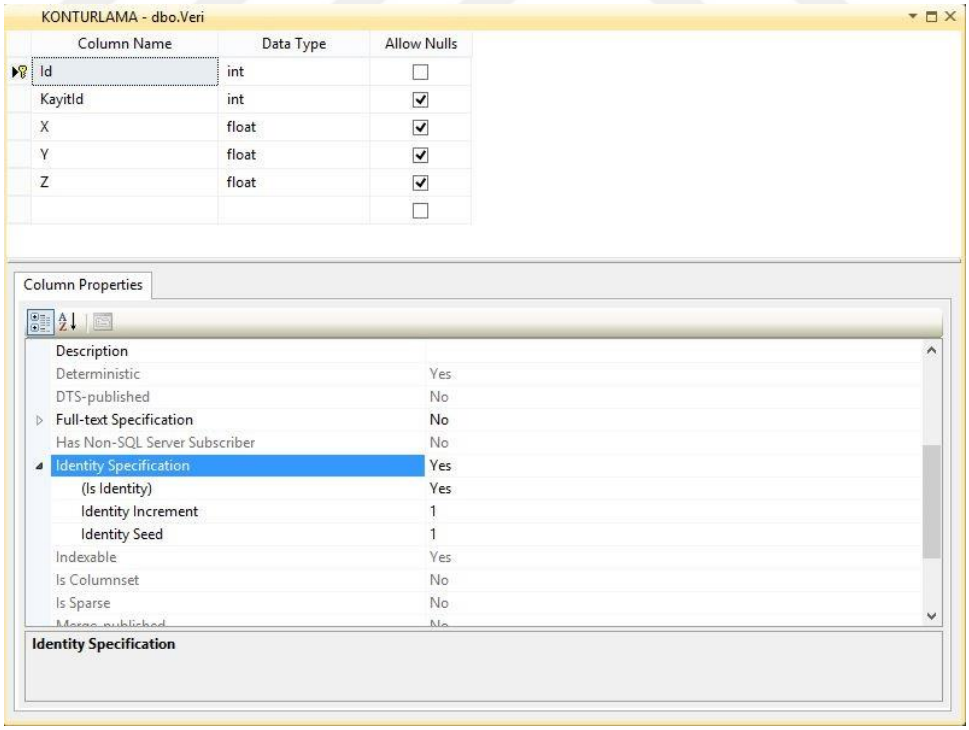
Şekil 3.2. Kayitlar tablosu tasarım görünümü.

Tablodaki Id veri alanının türü int olarak belirlenmiştir. Bu alan aynı zamanda tablonun birincil anahtarı olduğundan veri alanı özelliklerinden otomatik artan özelliği aktif edilmiştir. Böylece tabloya her yeni kayıt eklendiğinde Id alanının değeri bir artarak tabloda eşsiz alan olarak tanımlanmıştır. KayitAdi alanının veri türü nvarchar olarak belirlenmiştir. Bu tabloya ait örnek veriler Şekil 3.3'deki gibidir.



Şekil 3.3. Kayitlar tablosuna ait örnek veriler.

Kayıtlı çalışmalara ait ham veriler, Veri adındaki tabloda tutulmaktadır. Veri tablosu beş adet veri alanından oluşmaktadır. Şekil 3.4'de görüldüğü gibi X, Y ve Z adındaki veri alanlarının türü, tam sayı dışında da değerler alabileceği için float olarak belirlenmiştir.



Şekil 3.4. Veri tablosunun tasarım görünümü.

KayitId alanının veri türü int olarak belirlenmiştir. Bu alan Kayitlar tablosunun ikincil anahtarı olarak Veri tablosuna eklenmiştir. Böylece tablodaki verilerin hangi kayıtlarla ilişkili olduğu anlaşılabilmektedir. Id veri alanının türü int olarak belirlenmiştir. Bu alan aynı zamanda tablonun birincil anahtarı olduğundan veri alanı özelliklerinden otomatik artan özelliği aktif edilmiştir. Bu tabloya ait örnek veriler Şekil 3.5'deki gibidir.

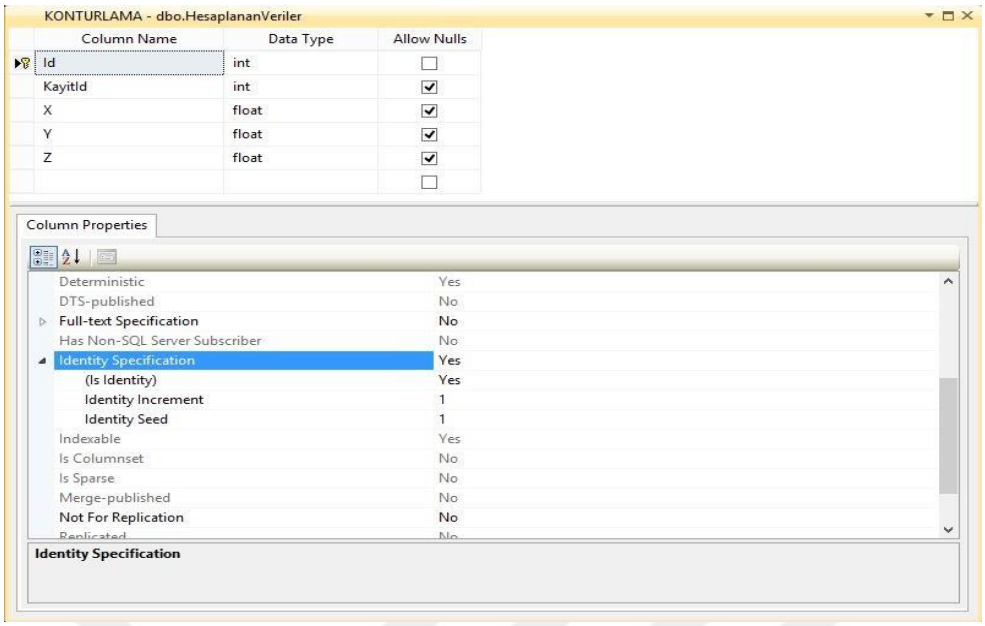
KONTURLAMA - dbo.Veri

Id	KayitId	X	Y	Z
1	9	1,8	1,2	5
2	9	2,5	4,5	3
3	9	3,5	2,4	1
56	14	1	1	55
60	14	2	1	30
61	14	3	1	80
63	23	0	0	0
71	14	3	2	75
72	14	1	2	45
73	14	2	2	35
74	14	3	3	60
75	14	1	3	50
76	14	2	3	40
94	22	1	4	4
95	22	2	3	3
96	22	3	2	4
97	22	4	1	5

1 of 17 Cell is Read Only.

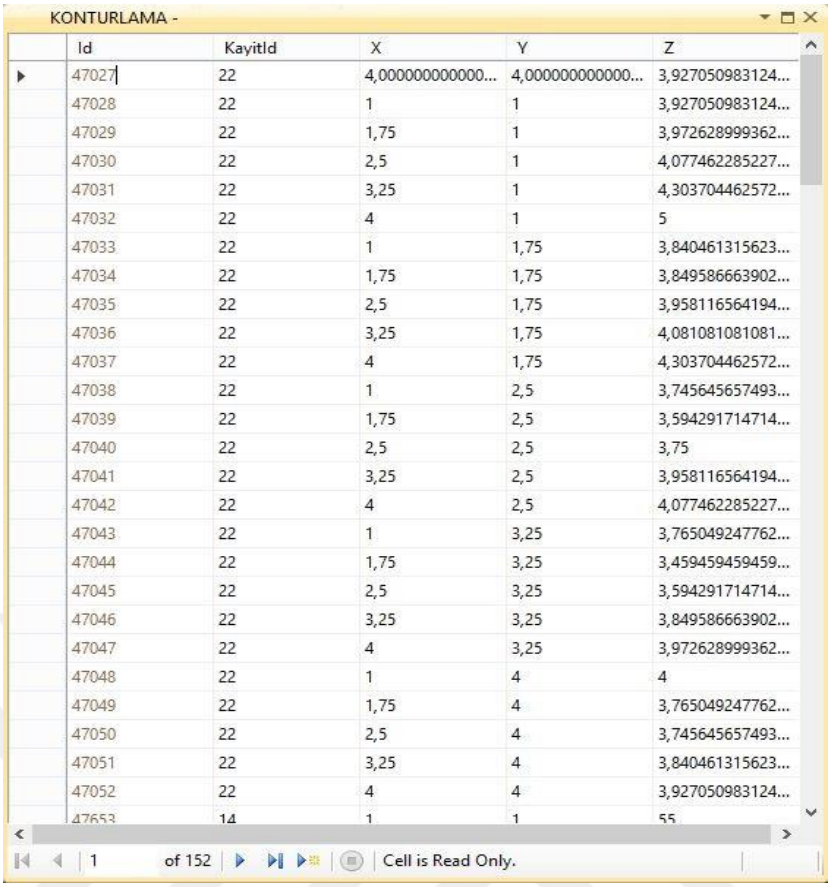
Şekil 3.5. Veri tablosuna ait örnek veriler.

Çalışmalara ait gridlenmiş veriler, HesaplananVeriler adındaki tabloda tutulmaktadır. Veri tablosundaki ham veriler mesafenin tersi yöntemine göre işlenerek HesaplananVeriler tablosuna aktarılmaktadır. Bu tabloda beş adet veri alanı bulunmaktadır. Şekil 3.6'da görüldüğü gibi X, Y ve Z adındaki veri alanlarının türü hem koordinat olarak hem de enterpolasyon işlemi sonrasında tam sayıdan farklı değerler alabileceği için float olarak belirlenmiştir.



Şekil 3.6. HesaplananVeriler tablosunun tasarım görünümü.

KayitId alanının veri türü int olarak belirlenmiştir. Bu alan Kayitlar tablosunun ikincil anahtarı olarak HesaplananVeriler tablosuna eklenmiştir. Böylece tablodaki hesaplanmış verilerin hangi kayıtlarla ilişkili olduğu anlaşılabilmektedir. Id veri alanının türü int olarak belirlenmiştir. Bu alan aynı zamanda tablonun birincil anahtarı olduğundan veri alanı özelliklerinden otomatik artan özelliği aktif edilmiştir. Bu tabloya ait örnek veriler Şekil 3.7'deki gibidir.



KONTURLAMA -

Id	KayitId	X	Y	Z
47027	22	4,000000000000...	4,000000000000...	3,927050983124...
47028	22	1	1	3,927050983124...
47029	22	1,75	1	3,972628999362...
47030	22	2,5	1	4,077462285227...
47031	22	3,25	1	4,303704462572...
47032	22	4	1	5
47033	22	1	1,75	3,840461315623...
47034	22	1,75	1,75	3,849586663902...
47035	22	2,5	1,75	3,958116564194...
47036	22	3,25	1,75	4,081081081081...
47037	22	4	1,75	4,303704462572...
47038	22	1	2,5	3,745645657493...
47039	22	1,75	2,5	3,594291714714...
47040	22	2,5	2,5	3,75
47041	22	3,25	2,5	3,958116564194...
47042	22	4	2,5	4,077462285227...
47043	22	1	3,25	3,765049247762...
47044	22	1,75	3,25	3,459459459459...
47045	22	2,5	3,25	3,594291714714...
47046	22	3,25	3,25	3,849586663902...
47047	22	4	3,25	3,972628999362...
47048	22	1	4	4
47049	22	1,75	4	3,765049247762...
47050	22	2,5	4	3,745645657493...
47051	22	3,25	4	3,840461315623...
47052	22	4	4	3,927050983124...
47653	14	1	1	55

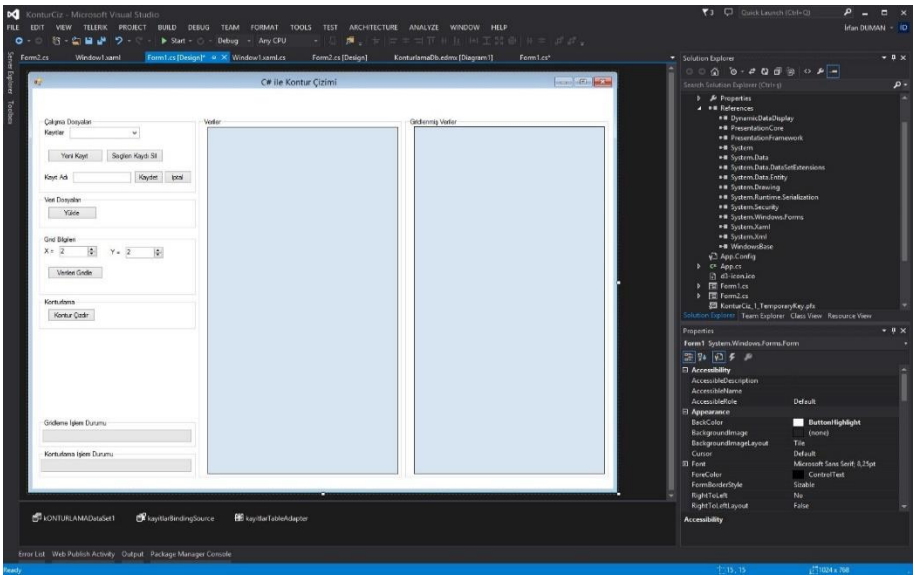
1 of 152 Cell is Read Only.

Şekil 3.7. HesaplananVeriler tablosuna ait örnek veriler.

Yazılım için tasarlanan bu üç adet veri tabanı tablosu, Kayıtlar tablosundan Veri tablosuna ve Kayıtlar tablosundan HesaplananVeriler tablosuna doğru birden çoğa ilişki türüyle birbirlerine bağlıdırlar. Tüm tablolarda Id alanları birincil anahtar olarak tanımlanmıştır. Veri ve HesaplananVeriler tablolarındaki KayıtId veri alanları ise Kayıtlar tablosunun ikincil anahtarı olarak tanımlanmıştır.

3.1.2. YAZILIM TASARIMI VE GELİŞTİRME

Bu çalışmada yazılım tasarımı ve geliştirilmesi sürecinde Şekil 3.8'de gösterildiği gibi Visual Studio 2013 ortamı ve C# programlama dili kullanılmıştır.



Şekil 3.8. Visual Studio uygulama geliştirme ve tasarım ortamı.

3.1.2.1. *Çalışma Dosyaları*

Uygulama içerisinde birden çok çalışma kontur grafiği çizdirilebilmektedir. Bu kontur grafiklerine ait toplanan veriler ve bu verilere ait gridlenmiş veriler SQL veri tabanında tutulmaktadır. Yeni bir çalışma için "Kayıt Adı" alanına çalışmanın adı yazılır ve "Kaydet" butonuna tıklanır ve Şekil 3.9'daki kodlar işletilir.

```
Kayitlar aKayit = new Kayitlar(); ;
aKayit.KayitAdi = textBox1.Text;
_db.AddToKayitlars(aKayit);
_db.SaveChanges();
```

Şekil 3.9. Yeni çalışma alanı kaydı için kod parçacığı.

Şekil 3.9'da Kayitlar sınıfından aKayit adında bir nesne oluşturulur. Kullanıcının textBox1 alanına yazdığı kayıt adı alınarak aKayit nesnesinin KayitAdi alanına aktarılır. Sonraki satırda _db adındaki varlık kümesinin AddToKayitlars metoduna parametre olarak aKayit nesnesi gönderilir. Son satırda _db nesnesinin SaveChanges metodu kullanılarak kullanıcıdan alınan kayıt adı veri tabanına kaydedilir. Şekil 3.9'daki kodlar işletildiğinde "Kayıtlar" adındaki tabloya yeni bir çalışma yapmak için kayıt açılmaktadır. Bu yeni açılan kaydın adı aynı zamanda açılabilir menüde de görüntülenmektedir. Böylece yeni açılan ve daha önce açılmış çalışma kayıtlarına açılabilir menü aracılığıyla erişilmektedir. Eğer herhangi bir çalışma dosyası verileri ile birlikte silinmek isteniyorsa, açılır menüden silinecek kayıt seçilir ve "Seçilen Kaydı Sil" butonuna tıklanır.

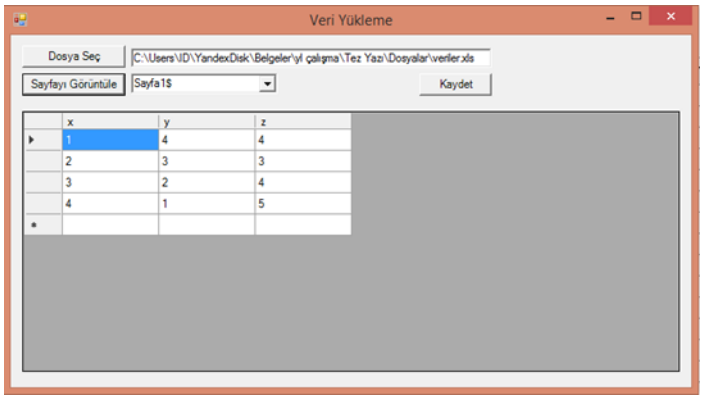
```
if(comboBox1.SelectedIndex !=null)
{
      int Id = Convert.ToInt32(comboBox1.SelectedValue.ToString());
      var dKayit = _db.Kayitlars.FirstOrDefault(i => i.Id == Id);
      _db.DeleteObject(dKayit);
      _db.SaveChanges();
      comboBox1.DataSource = _db.Kayitlars;
}
else
{
      MessageBox.Show("Silmek için bir kayıt seçmelisiniz!","Hata");
           }
```

Şekil 3.10. Seçim yapılan çalışma dosyasını silen kod parçacığı.

Şekil 3.10'da integer türünde Id isminde bir değişken tanımlanarak, comboBox1 adındaki açılır menüden seçilen değer aktarılmıştır. Sonraki satırda dKayit türünde nesne oluşturularak, Kayitlar tablosundaki seçilen kayıt bu nesneye aktarılmıştır. Sonraki satırda _db nesnesinin DeleteObject metoduna dKayit nesnesi parametre olarak gönderilmiş ve SaveChanges metoduyla seçilen kayıt veri tabanından silinmiştir. Sonraki satırda comboBox1 nesnesinin DataSource metodu kullanılarak veri tanının Kayitlar tablosu comboBox1 nesnesine bağlanmıştır. Böylece silme işlemi sonrasındaki değişikliğin açılır menüde gösterilmesi sağlanmıştır. Tüm bu kodlar If koşul bloğu içerisine alınarak kullanıcıdan kayıt seçebilmesi için mutlaka bir seçim yapması sağlanmıştır. Şekil 3.10'daki kodlar işletildiğinde açılır menüden seçilen kayda ait veriler "Kayıtlar" tablosundan silinmektedir.

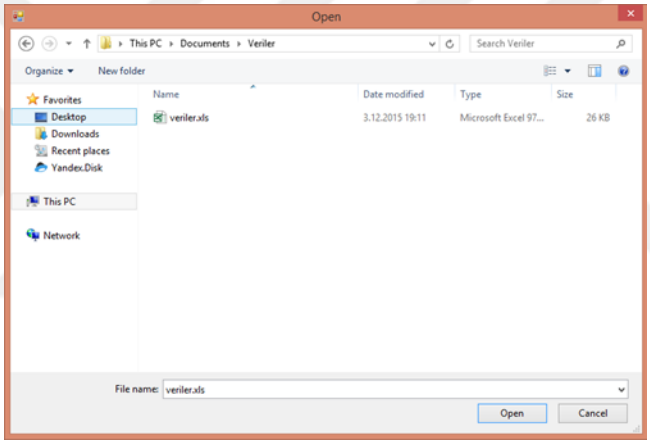
3.1.2.2. *Veri Dosyaları*

Çalışma dosyasına ait toplanmış veriler Excel ortamından yazılıma gridlenmek ve sonrasında kontur grafiği çizdirmek üzere yüklenmektedir. Yeni veriler yüklemek için "Veri Dosyaları" butonuna tıklanır ve Şekil 3.11'deki gibi veri yükleme formu açılır.



Şekil 3.11. Veri yükleme formu.

Dosya seç butonu tıklandığında openFileDialog nesnesi kullanılarak dizin gezinme aracı açılmaktadır. Şekil 3.12'de ki gibi açılan dizin gezinme formundan verilerin olduğu Excel formu seçilir.



Şekil 3.12. Dizin gezinme ve dosya seçme penceresi.

Dizin gezinme formundan verilerin bulunduğu Excel dosyası seçildikten sonra Şekil 3.13'de ki kodlar işletilir ve dosya okuma işlemi yapılır.

```
try
{
	string pathExcel = "Provider=Microsoft.Jet.OLEDB.4.0;" +
			"Data Source=" + textBox1.Text + ";" +
			"Extended Properties=\"Excel 8.0;HDR=Yes;\";";
	System.Data.DataTable dt = null;
	OleDbConnection conn = new OleDbConnection(pathExcel);
	conn.Open();
	dt = conn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, null);
	String[] excelSheets = new String[dt.Rows.Count];
	int i = 0;
	foreach (DataRow row in dt.Rows)
	{
	excelSheets[i] = row["TABLE_NAME"].ToString();
	i++;
	}
	for (int j = 0; j < excelSheets.Length; j++)
	{
		comboBox1.Items.Add(excelSheets[j].ToString());
	}
	conn.Close();
}
catch { MessageBox.Show("Görüntüleyecek sayfa bulunamadı!"); }
```

Şekil 3.13. Seçilen dosyanın sayfaların açılır menüde gösterilmesini sağlayan kod parçası.

Şekil 3.13'de try…catch bloğu kullanılarak dosya okuma işlemi sırasında hatayla karşılaşılması durumunda programın hata vererek çakılması önlenmiştir. Blok içerisinde pathExcel isminde değişken tanımlanmıştır. Bu değişkene OLEDB bağlantı cümleciği aracılığı ile okunacak dosyanın yolu belirtilmiştir. Sonraki satırda DataTable sınıfından dt adında boş bir nesne oluşturulmuştur. Sonraki satırda pathExcel değişkenindeki bağlantı cümleciği ile OleDbConnection sınıfından conn isimli bir bağlantı nesnesi oluşturulmuştur. Sonraki satırda conn nesnesinin Open metodu kullanılarak Excel dosyasına erişim için bağlantı açılmıştır. Sonraki satırda conn nesnesinin GetOleDbSchemaTable metodu aracılığıyla dosya içindeki tüm sayfalar dt nesnesine aktarılır. Sonraki satırda excelSheets adında dizi değişkeni oluşturulmuştur. Oluşturulan dizi değişkeninin boyutu sayfa içerisindeki satır sayısı olacak şekilde tanımlama yapılmıştır. Sonraki satırda foreach döngüsü, satır satır okuma işlemi yapılarak dosya içerisindeki sayfaların adı excelSheets değişkenine atanmıştır. Sonraki satırda yer alan for döngüsü, dizi içerisindeki sayfa adlarını satır satır comboBox1 adında ki açılır menüye eklemek için kullanılmıştır. Sonraki satırda conn nesnesinin Close metodu çağrılarak bağlantı sonlandırılmıştır.

Şekil 3.11' de ki form ara yüzündeki açılır menüye Excel dosyasının içindeki çalışma

sayfalarının isimleri yüklenmektedir. Açılır menüden verilerin yer aldığı sayfa seçildikten sonra "Sayfayı Görüntüle" butonu tıklanır. Şekil 3.14' deki kodlar işletildiğinde Excel dosyasına ait seçilen sayfadaki verilerin ön izlemesi görüntülenmektedir. Ön izleme sırasında veriler "Kaydet" butonuna tıklanarak ilgili çalışma dosyasına ait "Veri" tablosuna eklenmektedir.

```
try
{
       string pathExcel = "Provider=Microsoft.Jet.OLEDB.4.0;" +
                     "Data Source=" + textBox1.Text + ";" +
                     "Extended Properties=\"Excel 8.0;HDR=Yes;\";";
       OleDbConnection conn = new OleDbConnection(pathExcel);
       string cmd = "Select * from [" + comboBox1.Text + "]";
       OleDbDataAdapter myAdpt = new OleDbDataAdapter(cmd, conn);
       DataTable myDt = new DataTable();
       myAdpt.Fill(myDt);
       dataGridView1.DataSource = myDt;
}
catch { MessageBox.Show("Veri Yok!"); }
```

Şekil 3.14. Dosyanın okunup ön izlemesini sağlayan kod parçacığı.

Şekil 3.14'de try…catch bloğu kullanılarak dosya okuma işlemi sırasında hatayla karşılaşılması durumunda programın hata vererek çakılması önlenmiştir. Blok içersinde kaln kodlarda çalışma zamanı hatası oluşması durumunda catch bloğu içerisinde kalan MessageBox sınıfı ile kullanıcıya "Veri Yok!" uyarısı vererek bilgilendirmektedir. Blok içerisindeki ilk satırda pathExcel ismindeki değişkene OLEDB bağlantı cümleciği aracılığı ile okunacak dosyanın yolu aktarılmıştır. Sonraki satırda pathExcel değişkenindeki bağlantı cümleciği ile OleDbConnection sınıfından conn isimli bir bağlantı nesnesi oluşturulmuştur. Sonraki satırda conn nesnesinin Open metodu kullanılarak Excel dosyasına erişim için bağlantı açılmıştır. Sonraki satırda cmd adında string değişken tanımlanmıştır. Bu değişken içerisinde açılır menüden seçilen sayfa adı kullanılarak SQL sorgu cümlesi yazılmıştır. Sonraki satırda OleDbDateAdapter sınıfı kullanılarak myAdpt adında nesne oluşturulmuştur. Sonraki satırda DataTable sınıfı kullanılarak myDt adında bir nesne oluşturulmuştur. Sonraki satırda myAdpt nesnesinin Fill metodu kullanılarak myDt nesnesi myAdpt sınıfından çekilen verilerle doldurulmuştur. Sonraki satırda form üzerindeki dataGridView1 nesnesinin DataSource metodu kullanılarak myDt nesnesindeki veriler dataGridView1 nesnesine aktarılmıştır. Bu satırların tamamı işletildiğinde açılır menüden seçilen her sayfa için GridView nesnesi üzerinde ön izleme yapılmaktadır.

3.1.2.3. *Grid Bilgileri/Boyutu ve Gridleme*

Toplanan veriler dosyalardan uygulama veri tabanına aktarıldıktan sonra çalışma alanının boyutları "Grid Bilgileri" bölümündeki "X" ve "Y" parametreleri ile belirtilmektedir. Çalışma alanının boyutlar belirlendikten sonra "Verileri Gridle" butonuna tıklanır. Şekil 3.15' deki kodlar işletilmeye başlandığında kullanıcıya üzerinde çalışılan dosya ile ilgili daha önce yapılan grid hesaplamalarına ait verilerin silineceği mesajı verilmektedir.

```
if (MessageBox.Show("Daha önce gridlenmiş verileriniz silinecektir.
Devam etmek istiyor musunuz?", "Uyarı", MessageBoxButtons.YesNo) ==
System.Windows.Forms.DialogResult.Yes)
{
       var dGridData = _db.HesaplananVerilers.Where(i => i.KayitId ==
pKayitId);
      if (dGridData != null)
      {
            foreach (var item in dGridData)
            {
                  _db.DeleteObject(item);
            }
            _db.SaveChanges();
      }
⋮
```

Şekil 3.15. Daha önce gridlenmiş verileri silen kod parçacığı.

Kullanıcı onayından sonra If bloğunun altındaki kodlar işletilmektedir. Bu blokta dGridData nesnesi oluşturularak çalışma dosyalarına ait verilerin tutulduğu "Kayıt" tablosundan üzerinde işlem yapılan çalışmaya ait "Id" kullanılarak oluşturulan yeni nesneye aktarılmıştır. Sonraki satırda dGridData nesnesinin içerisinde veri olup olmadığı If kontrol yapısı ile kontrol edilmiştir. Eğer dGridData nesnesinde veri varsa Foreach kod bloğu işletilerek veriler satır satır veri tabanından silinmiştir. İşlem yapılan "Id" ile ilişkili veriler hem Şekil 3.8' deki form üzerindeki "Gridlenmiş Veriler" adındaki DataGrid nesnesinden hem de veri tanındaki "HesaplananVeriler" adındaki tablodan silinmektedir.

Eski gridleme işlemine ait veriler tablodan temizlendikten sonra Şekil 3.16' daki kodlar işletilmektedir. Bu kodlarda form ara yüzünden alınan gridleme boyut bilgilerini kullanılarak değişkenler tanımlanmaktadır.

```
int boyutX = Convert.ToInt32(numericUpDown1.Value) - 1;
int boyutY = Convert.ToInt32(numericUpDown2.Value) - 1;
double[, ,] grid_XYZ = new double[boyutX + 2, boyutY + 2, 3];
double grid_X_artis = 0;
double grid_Y_artis = 0;
HesaplananVeriler hVeriler = new HesaplananVeriler();
```

Şekil 3.16. Grid hesaplamaları için değişkenlerin tanımlandığı kod parçacığı.

Şekil 3.16'da boyutX ve boyutY değişkenleri tanımlanarak bu değişkenlere form üzerindeki grid boyut bilgilerini gösteren numericUpDown nesnelerinin değerleri aktarılmıştır. Sonraki satırda grid_XYZ adında üç boyutlu double türünde bir değişken tanımlanmıştır. Sonraki iki satırda grid_X_artis ve grid_Y_artis adında iki değişken tanımlanarak değerleri sıfıra eşitlenmiştir. Sonraki satırda HesaplananVeriler adındaki veri tabanı varlık nesnesi kullanılarak hVeriler adında nesne türetilmiştir. Değişken tanımlamalarından sonra gridleme işlemi başlamaktadır.

Gridleme işlemi mesafenin tersi yöntemi kullanılarak yapılmaktadır. Denklem (2.1) ve (2.2)'deki formüller C# kodlarına uyarlanarak Şekil 3.17'deki kodlar geliştirilmiştir.

```
grid_X_artis = (_db.Veris.Where(i => i.KayitId == pKayitId).Max(i =>
i.X).Value - _db.Veris.Where(i => i.KayitId == pKayitId).Min(i =>
i.X).Value) / boyutX;
grid_Y_artis = (_db.Veris.Where(i => i.KayitId == pKayitId).Max(i =>
i.Y).Value - _db.Veris.Where(i => i.KayitId == pKayitId).Min(i =>
i.Y).Value) / boyutY;
grid_XYZ[0, 1, 0] = _db.Veris.Where(i => i.KayitId == pKayitId).Min(i
=> i.X).Value - grid_X_artis; //X
grid_XYZ[0, 0, 1] = _db.Veris.Where(i => i.KayitId == pKayitId).Min(i
=> i.Y).Value - grid_Y_artis; //Y
for (int i = 1; i <= boyutY + 1; i++) //Y
{
      grid_XYZ[0, i, 1] = grid_XYZ[0, i - 1, 1] + grid_Y_artis;//Y
      for (int j = 1; j <= boyutX + 1; j++) //X
      {
            grid_XYZ[j, i, 0] = grid_XYZ[j - 1, 1, 0] +
      grid_X_artis;//X
            grid_XYZ[j, i, 1] = grid_XYZ[j - 1, i, 1];//Y
            double qX = grid_XYZ[j, i, 0];
            double qY = grid_XYZ[j, i, 1];
            if (_db.Veris.Count(id => id.X == qX && id.Y == qY &&
      id.KayitId == pKayitId) > 0)
            {
                  grid_XYZ[j, i, 2] = _db.Veris.FirstOrDefault(id =>
            id.X == qX && id.Y == qY && id.KayitId ==
            pKayitId).Z.Value;
            }
            else
            {
                  grid_XYZ[j, i, 2] =
            dugum_noktasi_degeri(grid_XYZ[j, i, 0], grid_XYZ[j, i,
            1]);
            }

                  hVeriler.X = grid_XYZ[j, i, 0];
                  hVeriler.Y = grid_XYZ[j, i, 1];
                  hVeriler.Z = grid_XYZ[j, i, 2];
                  hVeriler.KayitId = pKayitId;
                  _db.AddToHesaplananVerilers(hVeriler);
                  _db.SaveChanges(false);

            }…
```

Şekil 3.17. Grid hesaplamalarını yapan kod parçacığı.

Şekil 3.17'de grid_X_artis ve grid_Y_artis değişkenlerine, veri tabanındaki Veri tablosunda işlem yapılacak dosyaya ait en büyük ve en küçük X değerlerinin farkları alınarak gridleme boyutuna bölünüp aktarılmıştır. Geliştirilen algoritmaya göre iç içe For döngüleriyle her bir Y değeri için tüm X değerleri taranarak grid_XYZ adındaki üç boyutlu dizi doldurulmuştur. Üç boyutlu dizideki her bir değer hVeriler.X, hVeriler.Y ve hVeriler.Z nesnelerine aktarılmıştır. Döngü sonunda Veri tablosundan çekilerek geliştirilen algoritmaya göre hesaplanan değerler HesaplananVeriler tablosuna _db

adındaki veri tabanı varlık nesnesi kullanılarak kaydedilmiştir. Veri tabanına kaydedilen değerler aynı zamanda uygulama ara yüzünde ki "Gridlenmiş Veriler" adındaki DataGrid' e aktarılmıştır.

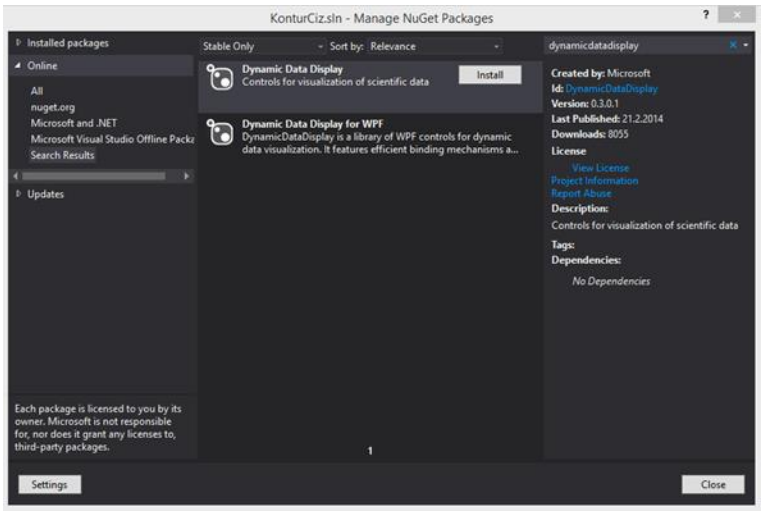
Gridleme işleminin süresi çalışma alanının boyutuna göre farklılık göstereceğinden, grid hesaplamaları yapılırken iç içe oluşturulan döngüler içine yerleştirilen Şekil 3.18'deki kod aracılığı ile ProgresBar nesnesi gridleme süresi boyunca güncellenmekte, gridleme işleminin durumu ara yüz aracılığı ile kullanıcıya yansıtılmaktadır.

```
pBarValue(100 * i / (boyutY + 1));
```

Şekil 3.18. Grid hesaplama durum çubuğu için kod parçacığı.

3.1.2.4. *Konturlama/Kontur Grafiği Çizdirme*

Bir sorunu çözmek için yazılmış kod parçasını tekrar tekrar ve farklı uygulamalarda kullanabilmek için kaynak kodu kütüphane oluşturularak saklanmaktadır. C# programlama dilinde de .Net Framework içinde bulunan kütüphaneleri kullanarak hızlı ve kolay bir şekilde uygulamalar geliştirebilir. Grafik işlemlerinde ise genellikle .Net Framework içesinde yer alan GDI kütüphanesi kullanılmaktadır. Bu kütüphane aracılığıyla basit çizgi ve şekiller çizdirilebilmekte ya da resim üzerinde işlemler kolayca yapılabilmektedir. DynamicDataDisplay, Visual Studio ortamında kod geliştirenler için Microsoft firması tarafından hazırlanmış GDI'dan daha görsel ve hızlı grafikler oluşturulabilecek bir kütüphanedir. Şekil 3.19'da görüldüğü gibi DynamicDataDisplay kütüphanesi Microsoft' tun web sayfasından indirilerek ya da Visual Studio penceresi üzerinde, araçlar sekmesinden "Paket Yöneticisi" kullanılarak projeye dahil edilip kullanılmaya başlanabilir.



Şekil 3.19. Dynamic Data Display kütüphanesinin projeye eklenmesi.

Proje dosyasına kütüphane eklendikten sonra Şekil 3.20'deki gibi namespace tanımlaması yapılarak kullanılmaktadır.

```
using IsolineSampleApp;
```

Şekil 3.20. Kütüphaneye ait namespace tanımlaması.

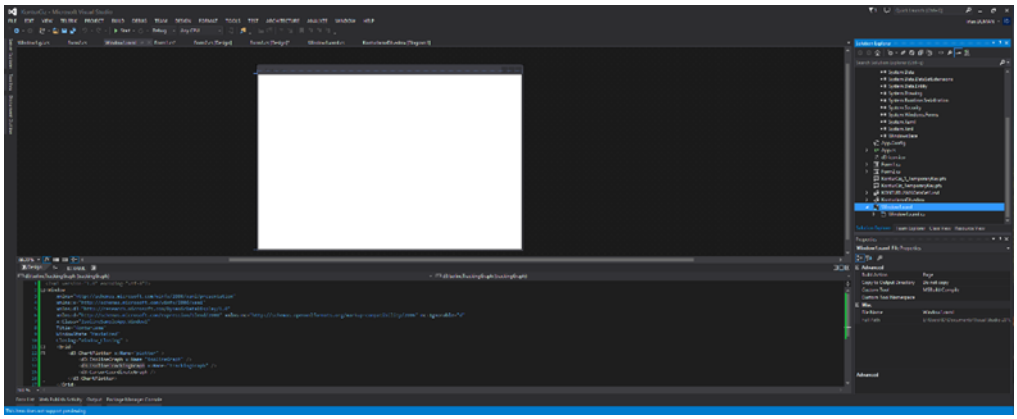
Projeye dahil edilen DynamicDataDisplay kütüphanesini, namespace tanımlama alanında IsolineSampleApp anahtar sözcüğünü kullanılarak tanımlanmaktadır.

Grid hesaplamaları tamamlanmış verilere ait kontur grafiğini çizdirmek için Şekil 3.8'deki "Kontur Çiz" butonu tıklanır.

```
Window1 kontur_formu = new Window1();
kontur_formu.Show();
```

Şekil 3.21. Çizim alanı nesnesi oluşturma.

Kullanılan kütüphane içerisindeki Window1 sınıfı Şekil 3.21'deki gibi çağrılarak kontur_formu adında bir nesne oluşturulmaktadır. Bu nesneye ait Show method u kullanılarak Şekil 3.22'deki kontur grafiklerinin çizdirileceği çizim alanı ekranda gösterilmektedir.



Şekil 3.22. Kontur grafiği çizim alanı (tasarım anı).

Çizim alanı nesnesi Show method aracılığı ile çağrıldığında bu nesneye tanımlanan Şekil 3.23'deki kodlar çalıştırılmaya başlanmıştır.

```
KONTURLAMAEntities _db = new KONTURLAMAEntities();
pKayitId = Form1.pKayitId;
```

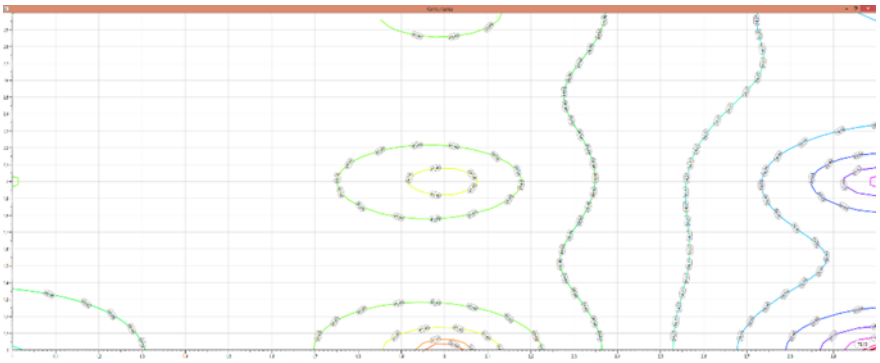
Şekil 3.23. Çalışma dosyasına ait grid verilerini çeken kod parçacığı.

Çalışma dosyasına ait "Id" değerine göre "HesaplananVeriler" tablosundan üzerinde çalışılan dosyanın verileri çekilerek iki boyutlu dizilere aktarılmaktadır. Dizilere aktarılan değerler koordinatlar ve bu koordinatlara ait Z değerleridir. Dizilere aktarılan değerler Şekil 3.24'deki kod parçacıkları kullanılarak kontur grafiğine dönüştürülür.

```
WarpedDataSource2D<double> konturegrisi = new
WarpedDataSource2D<double>(zdegeri, noktalar);
this.isolineGraph.DataSource = konturegrisi;
this.trackingGraph.DataSource = konturegrisi;
Rect gridBounds =
IDataSource2DExtensions.GetGridBounds<double>(konturegrisi);
this.plotter.Viewport.Visible = gridBounds;
```

Şekil 3.24. Izgara, kontur ve çerçeve nesnelerinin oluşturulmasını sağlayan kod parçacığı.

Şekil 3.24'de WarpedDataSource sınıfıyla konturegrisi adında nesne oluşturulmaktadır. Bu nesne oluşturulurken kontur grafiğine ait koordinat ve bu koordinatlara ait Z değerleri de nesneye aktırılmaktadır. Bu nesne kontur grafiği çizdirecek isolineGraph sınıfı için veri kaynağı niteliği taşımaktadır. Aynı zamanda kontur grafiğinin üstüne oturtulacağı ızgarayı çizen gridBounds sınıfı ve kontur eğrilerinin Z değerlerini kontur eğrileri üzerine yazacak tracikingGraph sınıfı içinde veri kaynağı olacaktır.



Şekil 3.25. Örnek kontur grafiği.

Kontur çizdirme işlemi tamamlandığında Şekil 3.25'de görüldüğü gibi çizim alanı açılmakta ve çizim alanın üzerine ızgara, kenar çizgileri ve kontur eğrisi çizdirilerek kontur grafiği çizdirme işlemi tamamlanmaktadır.

4. BULGULAR VE TARTIŞMA

Bu çalışmada, mesafenin tersi yöntemini kullanarak enterpolasyon yapabilen ve enterpolasyon sonucuna göre kontur grafiği çizebilen bir yazılım geliştirilmiştir. Geliştirilen yazılımın sonuçlarını karşılaştırabilmek için aynı işlemleri yapabilen Golden Software firmasına ait Surfer yazılımı kullanılmıştır. Karşılaştırmak için kullanılan ticari yazılım ile veriler mesafenin tersi yöntemi kullanılarak gridlenebilir ve kontur grafikleri çizdirilebilir. Çalışmada hem mesafenin tersi yöntemi kullanılarak gridleme yapıldığından hem de gridlenmiş verilere ait kontur grafikleri çizdirildiğinden Surfer yazılımı, çalışma için geliştirilen yazılım ile karşılaştırmaya uygun bir yazılım olarak görülmektedir.

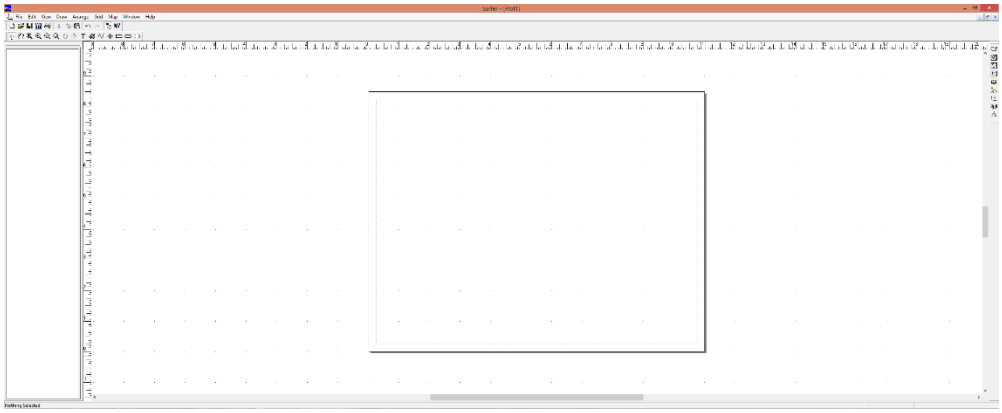
Çalışma için geliştirilen uygulamada ve karşılaştırmak için Surfer yazılımında Çizelge 4.1'deki aynı örnek veri seti kullanılmıştır. Bu veri seti küçük bir çalışma alanını örneklemektedir ve her iki yazılım içinde enterpolasyon yöntemi olarak mesafenin tersi kullanılmıştır. Her iki yazılım içinde örnek veri setleri mesafenin tersi yöntemiyle gridlendikten sonra gridlenmiş veriler ve bu gridlenmiş verilerle çizdirilen kontur grafikleri karşılaştırılmıştır.

Çizelge 4.1. Örnek veri seti.

X	Y	Z
1	1	55
2	1	30
3	1	80
3	2	75
1	2	45
2	2	35
3	3	60
1	3	50
2	3	40

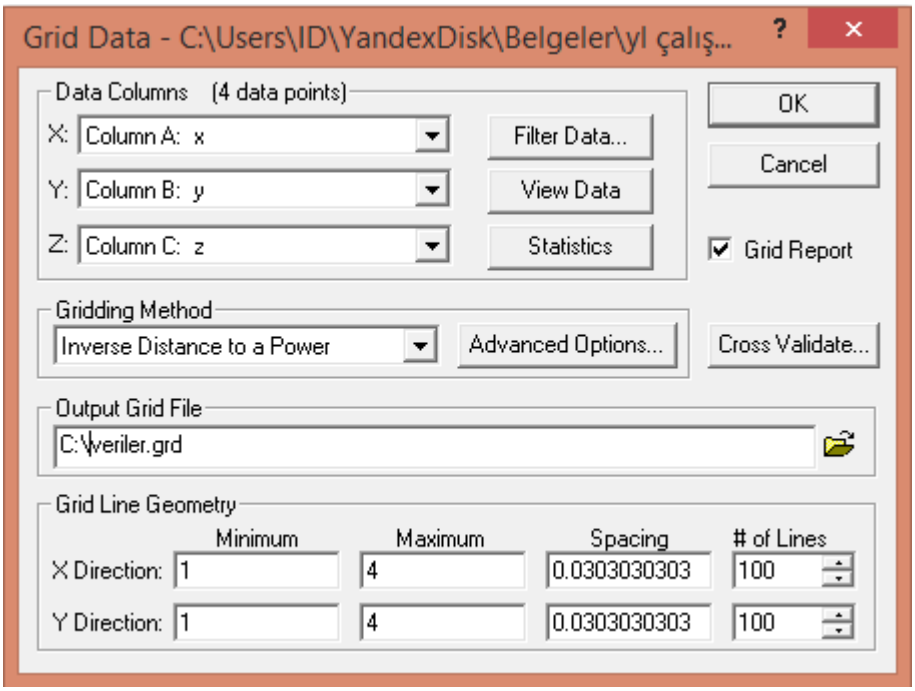
Örnek veri setine mesafenin tersi enterpolasyon yöntemi uygulanmakta ve bilinen noktalardan bilinmeyen noktalar için Z değerleri oluşturulup örnek veri seti gridlenmektedir.

Örnek veri seti .txt uzantılı bir dosyaya ya da .xls uzantılı bir Excel dosyasına kaydedilip kaydedilmektedir. Surfer uygulaması çalıştırıldığında Şekil 4.1'deki gibi boş bir çalışma alanı açılmaktadır. Bu çalışma alanı üzerine kontur grafikleri çizdirilmektedir.



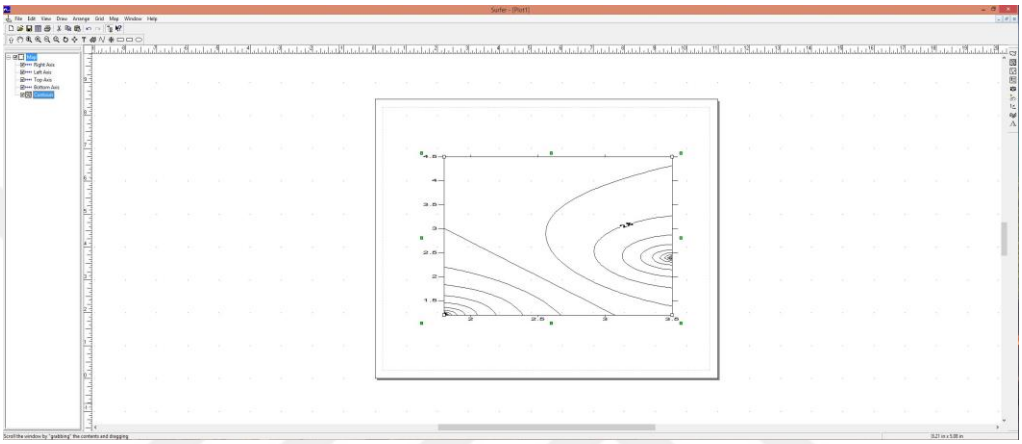
Şekil 4.1. Surfer yazılımı çalışma alanı.

Kontur grafiği çizdirebilmek için çalışma alanına ait tüm noktaların verileri olmalıdır. Eğer çalışma alanına ait verilerin tamamı yoksa ya da çalışma alanının belli kesitlerinde veri toplanamıyorsa eldeki veriler enterpolasyon yöntemleriyle gridlenerek bilinen noktalardan bilinmeyen noktalar elde edilmeye çalışılır. Çalışma alanına ait verileri Surfer uygulamasına yükleyebilmek için Grid menüsünün altında yer alan Data butonu tıklanır. Gridlenecek verilerin bulunduğu dosya seçildikten sonra Şekil 4.2' deki gibi bir pencere açılmaktadır.



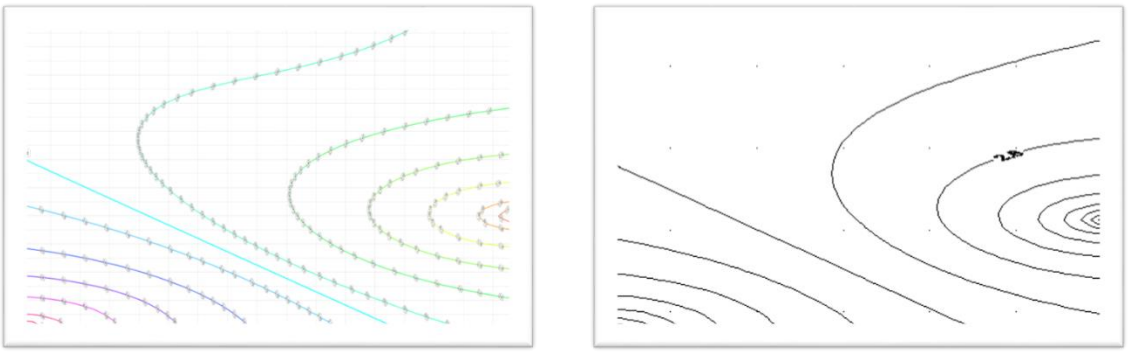
Şekil 4.2. Surfer yazılımı gridleme parametleri.

Bu pencere üzerinde enterpolasyon yöntemi, grid boyutu ve gridlenmiş verilerin nerede saklanacağı belirledikten sonra tamam butonuna basıldığında gridleme işlemi başlatılmaktadır. Gridleme işlemi tamamlandığında seçilen konuma .grd uzantılı dosyada gridlenmiş veriler kaydedilerek saklanmaktadır. Çalışma alanı üzerine gridlenmiş verilere ait kontur grafiği çizdirebilmek için açılır menülerden Map tıklanır Contour Map seçeneği seçilir ve altındaki New Contour Map seçeneği tıklanarak çalışma alanı üzerine kontur grafiği Şekil 4.3'deki gibi çizdirilir.



Şekil 4.3. Surfer yazılımında çalışma alanına kontur çizdirme.

Bu çalışma kapsamında geliştirilmiş yazılım ve Golden Software firmasına ait Surfer yazılımı üzerinde aynı veri seti kullanılarak gridleme işlemi yapılmış ve gridlenmiş veriler kullanılarak Şekil 4.4'deki gibi her iki yazılım üzerinde kontur grafikleri çizdirilmiştir.



a. b.

Şekil 4.4. Kontur grafikleri (a) Tez kapsamında geliştirilen yazılım b) Surfer yazılımı).

Şekil 4.4.a'da bu çalışma kapsamında geliştirilen yazılım aracılığı ile gridlenmiş ve bu gird verileri kullanılarak çizdirilen kontur grafiği görülmektedir. Şekil 4.4.b'de aynı veri seti kullanılarak Golden Software firmasına ait Surfer yazılımı kullanılarak gridlenmiş ve

bu grid verileri kullanılarak çizdirilen kontur grafiği görülmektedir. Ayrıca iki ayrı veri seti daha kullanılarak tez kapsamında geliştirilen yazılım ve Surfer yazılımında kontur grafikleri çizdirilerek Ek-2 ve Ek-3'de sunulmuştur. Şekil 4.4, Ek-2 ve Ek-3'de de görüldüğü üzere geliştirilen yazılım ile Surfer yazılımının oluşturduğu kontur grafikleri benzerlik göstermektedir. Bu benzerliğe dayanarak kullanılan enterpolasyon yönteminin hesaplamalarının doğru yapıldığı ve çizdirilen kontur haritasının da buna bağlı olarak doğru bir şekilde oluşturulduğu söylenebilir.

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada, farklı çalışma alanları için toplanan verilerin mesafenin tersi enterpolasyon yöntemi kullanılarak gridlenip, elde edilen gridlenmiş verilerle kontur grafiği çizebilen bir yazılım geliştirilmesi amaçlanmıştır. Bu kapsamda mesafenin tersi yönteminde kullanılan formüller için algoritmalar geliştirilmiştir. Bu algoritmalar C# programlama diline çevrilerek Visual Studio geliştirme ortamında yazılım geliştirilmiştir. Geliştirilen uygulamanın sonuçları, örnek veri setlerine mesafeye bağlı ters ağırlık yöntemi uygulanıp gridleme işlemi yapılmıştır. Elde edilen gridlenmiş yeni veri seti çalışma kapsamında geliştirilen yazılım kullanılarak kontur grafiği oluşturulmuştur. Aynı veri seti Golden Software firmasına ait Surfer programında da kullanılarak oluşturulan kontur grafiği karşılaştırılmış ve hesaplama sonuçlarının doğru, buna bağlı olarak kontur grafiğinin de doğru bir şekilde oluşturulduğu görülmüştür. Örnek veri setleriyle yapılan diğer karşılaştırmalarda da benzer sonuçlar elde edilmiştir. Bu sonuçlarda Ek kısmında sunulmuştur. Karşılaştırma sonuçları birbirleriyle benzerlik gösterdiğinden, farklı çalışma alanları için piyasada ücretli olarak sunulan yazılımların yerine geçebileceği düşünülmektedir.

Çalışmaya ait sınırlılıklar belirlenirken maliyet ve süre gözüne alınmıştır. Çalışmanın sınırlılıklarından bir tanesi, sunucu maliyetleri gözetilerek yazılımın masaüstü uygulaması olarak geliştirilmiş olmasıdır. Fakat uygulamanın C# programlama diliyle yazılmış olması sonraki çalışmalarda web tabanlı sürümünün de geliştirilmesinde kolaylık sağlayacaktır. Çalışmanın diğer sınırlılığı, geliştirilen yazılımda enterpolasyon yöntemlerinden sadece birinin kullanılıyor olmasıdır. Yazılım, nesneye dayalı programlama metodolojisiyle geliştirildiğinden sonraki çalışmalarda farklı enterpolasyon yöntemleri de eklenerek yazılımın birden fazla enterpolasyon yöntemi için hesaplamalar yaparak kontur grafiği çizdirmesi sağlanabilir.

6. KAYNAKLAR

[1] A. Sezer, Coğrafya öğretim teknolojileri ve materyal tasarımı, 1.Baskı, Ankara, Türkiye: Pegem Akademi, 2017.

[2] T. Emre, Harita çizimi, *Ders Notları*, İzmir, 2003.

[3] K. Dirik, Yeraltı harita çeşitleri, *Ders Notları*, Ankara, 2013.

[4] P. Manuel and D. Maidment, "Influence of DEM interpolation methods in drainage analysis," *Gis Hydro*, vol. 4, 2004.

[5] Q. Liang, S. Nittel, J. C. Whittier and S. d. Bruin, "Real-time inverse distance weighting interpolation for streaming sensor data," *Transactions in GIS*, vol. 22, no. 5, pp. 1179-1204, 2018.

[6] A. Menteşe, "Ultrason görüntülerinde karaciğer damarlarının otomatik sınıflandırılması," Yüksek lisans tezi, Bilgisayar Mühendisliği Bölümü, Gazi Üniversitesi, Ankara, Türkiye, 2018.

[7] F. Biltekin, "Lokal ileri evre serviks kanseri tedavisinde adaptif brakitterapi uygulaması: Bir olgu sunumu eşliğinde tedavi planlama ve değerlendirilmesi," *Turkish Journal of Oncology*, c. 32, s. 1, ss. 87-92, 2017.

[8] U. Baykara ve N. Alemdaroğlu, "Yüksek hızlı sahte hedef İHA tasarımı," *Sürdürülebilir Havacılık Araştırmaları Dergisi*, c. 1, s. 2, ss. 55-65, 2016.

[9] E. E. Özsavaş, Z. Telatar, B. Dirican ve Ö. Sağer, "Akciğerler ve geniş kanserli alanların tam otomatik olarak bölütlenmesi," *2014 IEEE 22nd Signal Processing and Communication Application Conference*, Trabzon, Türkiye, 2014, ss. 606-609.

[10] P. Sındırgı, O. Pamukçu ve Ö. Akdemir, "Yeraltısuyu tuzluluk ve kirlilik çalışmasında özdirenç yöntemi ve güzelçamlı(Adın-Kuşadası) uygulama alanı," *DEÜ Mühendislik Fakültesi Fen ve Mühendislik Dergisi*, c. 12, s. 2, ss. 65-74, 2010.

[11] G. Aslan, "GPS ölçümleri ve fay haraketlerinde deprem ilişkisi," Yüksek lisans tezi , İnşaat Mühendisliği Bölümü, İstanbul Teknik Üniversitesi, Ankara, Türkiye, 2012.

[12] T. Ramesh S.V. and C. V., "Improved weighting methods, deterministic and stochastic data-driven models for estimation of missing precipitation records," *Journal of Hydrology*, vol. 312, no. 1-4, pp. 191-205, 2005.

[13] S. Kazancı ve E. Kayıkçı, "Konumsal enterpolasyon yöntemleri uygulamalarında optimum parametre seçimi: Doğu karadeniz bölgesi günlük ortalama sıcaklık verileri örneği," *15. Türkiye Harita Bilimsel ve Teknik Kurultayı*, Ankara, Türkiye, 2015.

[14] M. Yalanak, "Transformation of ellipspid heights to local leveling heights," *Journal Of Surveying Engineering*, vol. 127, no. 3, pp. 90-103, 2001.

[15] M. Mcallister and J. Snoeyink, "Medial Axis Generalization of River Networks," *Cartography and geographic information science*, vol. 27, pp. 129-138, 2000.

[16] M. Yalanak, "Sayısal arazi modellerinde hacim hesaplarında en uygun enterpolasyon yönteminin araştırılması," Doktora tezi, Geomatik Mühendisliği, İstanbul Teknik Üniversitesi, İstanbul, Türkiye, 1997.

[17] M. Yalanak, "Sayısal arazi modellerinde yükseklik enterpolasyonu," *Harita ve*

Kadastro Mühendisliği Dergisi, c. 1, s. 128, ss. 23-38, 2002.

[18] C. S. Yang, S. P. Kao, F. B. Lee and P. S. Hung, "Twelve different interpolation methods: A case study of surfer 8.0," *In Prooceedings of The XXth ISPRS Congress*, Istanbul, Turkey, 2004, pp. 778-785.

[19] T. Üstüntaş, "Sayısal arazi modellerinde bazı enterpolasyon yöntemlerinin karşılaştırılması," *Selçuk-Teknik Dergisi*, c. 5, s. 2, ss. 41-48, 2006.

[20] S. Köroğlu, "Farklı enterpolasyon yöntemlerinin hacim hesabına etkisinin araştırılması," Yüksek lisans tezi, Jeodezi ve Fotogrametri Mühendisliği, İstanbul Teknik Üniversitesi, İstanbul, Türkiye, 2006.

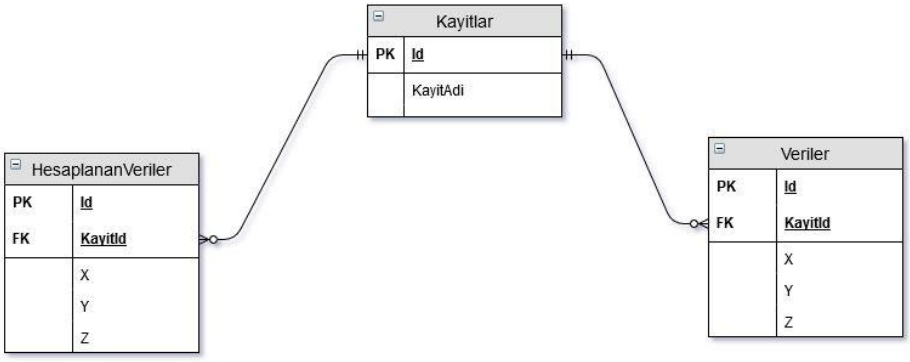
[21] A. Soycan ve M. Soycan, "Yol projelerinde sayısal arazi modellerinin kullanılması," *Selçuk Üniversitesi Jeodezi ve Fotogrametri Mühendisliği 30.Yıl Sempozyumu*, Konya, Türkiye, 2002.

[22] J. Sharp, Adım adım microsoft visual C#, 1. Baskı, Ankara, Türkiye: Arkadaş Yayınevi, 2010.

7. EKLER

7.1. EK 1: ER DİYAGRAMI

Veri tabanı modellemesi yapılırken Crow's Foot gösterimi dikkate alınarak Şekil 7.1'deki ER diyagramı hazırlanmıştır.



Şekil 7.1. Veri tabanı ER diyagramı.

7.1.1. İlişkisel Şema

Şekil 7.1'deki diyagrama ait ilişkisel şema aşağıdaki gibidir.

Kayitlar (Id: int, KayitAdi: nvarchar(50))

Veriler (Id: int, KayitId: int, X: float, Y:float, Z:float)

HesaplananVeriler (Id: int, KayitId: int, X: float, Y: float, Z: float)

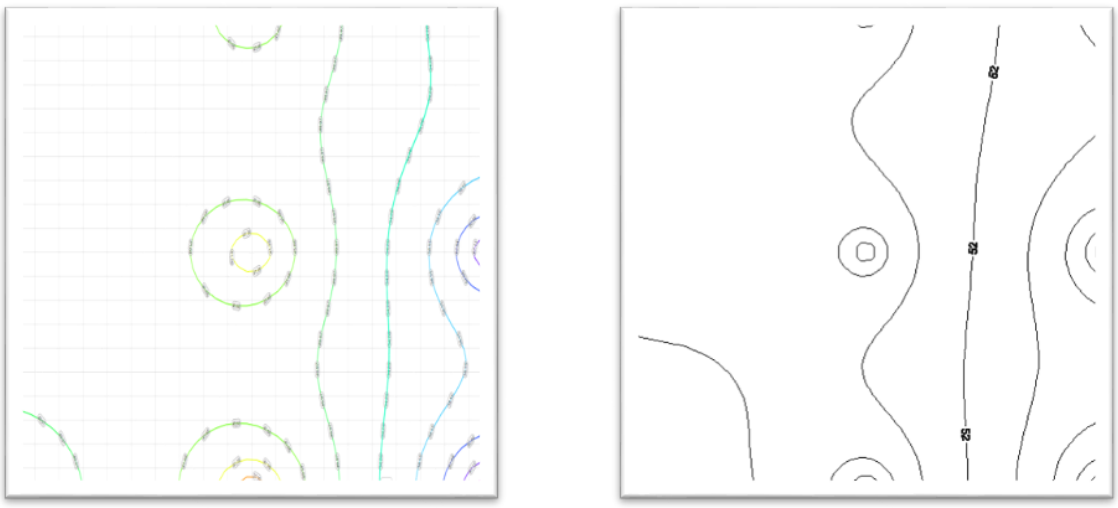
7.2. EK 2: ÖRNEK VERİ SETİ 2 VE KONTUR GRAFİĞİ

Tez kapsamında geliştirilen yazılım ve Surfer yazılımında Çizelge 7.1'deki veri seti kullanılmıştır.

Çizelge 7.1. Örnek Veri Seti 2.

X	Y	Z
1	1	55
2	1	30
3	1	80
3	2	75
1	2	45
2	2	35
3	3	60
1	3	48
2	3	40

Şekil 7.2'deki kontur grafikleri, Çizelge 7.1' deki veri seti kullanılarak oluşturulmuştur. Tez kapsamında geliştirilen yazılım kullanılarak önce gridleme işlemi gerçekleştirilmiş ardından Şekil 7.2.a'daki kontur grafiği oluşturulmuştur. Aynı işlem basamakları Surfer yazılımında da yapılarak Şekil 7.2.b'deki kontur grafiği oluşturulmuştur.



a. b.

Şekil 7.2. Kontur grafikleri (a) Tez kapsamında geliştirilen yazılım b) Surfer yazılımı).

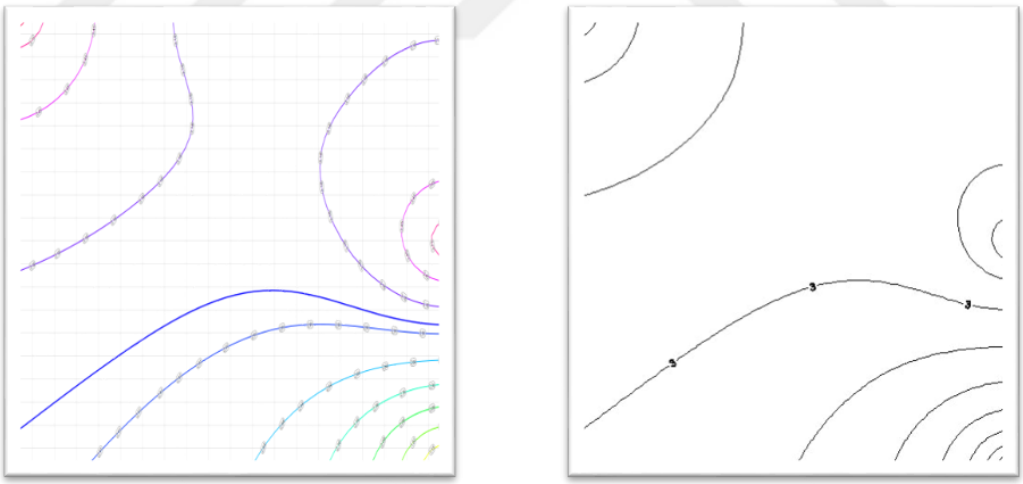
7.3. EK 3: ÖRNEK VERİ SETİ 3 VE KONTUR GRAFİĞİ

Tez kapsamında geliştirilen yazılım ve Surfer yazılımında Çizelge 7.2'deki veri seti kullanılmıştır.

Çizelge 7.2. Örnek Veri Seti 3.

X	Y	Z
1	3	4
2	3	3
3	2	4
3	1	1

Şekil 7.3'deki kontur grafikleri, Çizelge 7.1' deki veri seti kullanılarak oluşturulmuştur. Tez kapsamında geliştirilen yazılım kullanılarak önce gridleme işlemi gerçekleştirilmiş ardından Şekil 7.3.a'daki kontur grafiği oluşturulmuştur. Aynı işlem basamakları Surfer yazılımında da yapılarak Şekil 7.3.b'deki kontur grafiği oluşturulmuştur.



a. b.

Şekil 7.3. Kontur grafikleri (a) Tez kapsamında geliştirilen yazılım b) Surfer yazılımı).

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İrfan DUMAN

Doğum Tarihi ve Yeri : 27.06.1985 Kars

Yabancı Dili : İngilizce

E-posta : irfanduman@beun.edu.tr

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Yüksek Lisans	Elektirik-Elektronik ve Bilgisayar Mühendisliği	Düzce Üniversitesi	2018
Lisans	BÖTE	Gazi Üniversitesi	2009
Lise	Bilişim Teknolojileri	G.A.M.P.M.T.A. Lisesi	2003

YAYINLAR

1. İ. Duman, R. Kara, ve E. Çetiner, "C# Kullanarak Mesafeye Bağlı Ters Ağırlık Yöntemi ile Gridleme ve Kontur Çizimi," *Karaelmas Fen ve Mühendislik Dergisi*, c. 6, s.1, ss. 16-21, 2016