

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**İTÜ TRİSONİK LABORATUARI AERODİNAMİK DENEY SONUÇLARININ
ANALİZİ VE GÖRSELLEŞTİRİLMESİ**

YÜKSEK LİSANS TEZİ

Ali BAŞEL

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

HAZİRAN 2013

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**İTÜ TRİSONİK LABORATUARI AEORODİNAMİK DENEY SONUÇLARININ
ANALİZİ VE GÖRSELLEŞTİRİLMESİ**

YÜKSEK LİSANS TEZİ

**Ali BAŞEL
(511881027)**

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Duygu ERDEM

HAZİRAN 2013

İTÜ, Fen Bilimleri Enstitüsü'nün 511881027 numaralı Yüksek Lisans öğrencisi **Ali BAŞEL**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “**İTÜ TRİSONİK LABORATUARI AERODİNAMİK DENEY SONUÇLARININ ANALİZİ VE GÖRSELLEŞTİRİLMESİ**” başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Yrd. Doç. Dr. Duygu ERDEM**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. Filiz BAYTAŞ**
İstanbul Teknik Üniversitesi

Yrd. Doç. Dr. Hayri ACAR
İstanbul Teknik Üniversitesi

Teslim Tarihi : **17 Nisan 2013**
Savunma Tarihi : **05 Haziran 2013**

ÖNSÖZ

Bu tezin hazırlanmasında özellikle başlangıç kısmında yardımcı olan ve yön gösteren arkadaşım Uçak Mühendisi Aziz Bekkine'ye teşekkür ederim.

Tüm tez çalışmam boyunca yönlendiren ve yardımlarını esirgemeyen danışmanım Yrd. Doç. Dr. Duygu Erdem'e teşekkür ederim.

Nisan 2013

Ali Başel
(Uçak Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ.....	v
İÇİNDEKİLER	vii
KISALTMALAR	ix
ÇİZELGE LİSTESİ.....	xi
ŞEKİL LİSTESİ.....	xiii
ÖZET.....	xvii
SUMMARY	xix
1. GİRİŞ	1
1.1 Tezin Amacı	1
2. DENEY SİSTEMLERİ.....	3
2.1 Sıcak Tel Anemometresi	3
2.1.1 Sıcak tel sensörleri	5
2.1.2 Sıcak tel anemometre donanımı	7
2.1.3 Güçlü yanlar, sınırlar ve lazer hızölçer ile karşılaştırma	7
2.1.4 Veri formatı	10
2.2 Parçacık görüntü hızölçeri	10
2.2.1 Veri formatı	13
2.3 Çok delikli basınç probu	13
2.3.1 Veri formatı	15
3. ALGORİTMALAR.....	17
3.1 Akım Çizgileri.....	17
3.1.1 Akım çizgisi fonksiyonu	17
3.1.2 Akım çizgisi algoritması	23
3.1.2.1 Belirli bir akım çizgisinin çizilmesi	25
3.1.2.2 Seçilen bir noktadaki akım çizgisinin çizilmesi	27
3.1.2.3 Tüm akım çizgilerinin çizilmesi	27
3.2 Girdaplılık Çizgileri	28
3.2.1 Girdaplılık fonksiyonu	28
3.2.2 Girdaplılık çizgisi algoritması	33
3.3 Program Algoritmaları	35
3.3.1 Grid veri yapısı	35
3.3.2 Ölçüm noktalarının grid matrisine yerleştirilmesi	36
3.3.3 Katı cisim tanımlama	38
3.3.4 Katı cisim yüzey konturunun belirlenmesi	38
3.3.5 Cisim Konturu içinde Kalan Noktaların Belirlenmesi	41
3.3.6 Grid hücresinin köşelerinin belirlenmesi	41
3.3.7 Seçilen bir noktanın bulunduğu hücrenin belirlenmesi	42
3.3.8 Seçilen bir noktanın hücre içinde bulunduğu üçgen'in belirlenmesi	43
3.3.9 Katı cisim sayma ve numaralandırma	44
4. PROGRAM.....	47
4.1 Genel Çalışma Prensibi ve Çalışma Adımları	47

4.2 Genelleştirilmiş Parametrik Yapı ve Çizim Fonksiyonları	49
4.3 Programın Doğrulaması.....	51
4.3.1 Potansiyel akım örneği ile kıyaslama.....	51
4.3.2 Potansiyel akım hız vektörleri ve akım çizgileri ile kıyaslama.....	53
4.3.3 Mevcut uygulamalarla kıyaslama.....	57
4.3.4 Girdaplılık Çizimi Doğrulaması.....	59
4.4 Derleme ve Kurulum	61
4.4.1 Kullanılan kütüphaneler	61
4.4.2 Programın derlenmesi	62
4.4.3 Programın kurulması ve çalıştırılması.....	64
4.5 Programın Kullanımı	65
4.5.1 Katı cisim tanımlama	68
4.5.1.1 Daire Tanımlama.....	68
4.5.1.2 Dikdörtgen Tanımlama	69
4.5.1.3 Üçgen tanımlama.....	69
4.5.1.4 Denklem ile tanımlama	70
4.5.1.5 Şekil dosyası ile tanımlama.....	70
4.5.2 Cisimi yakınlaştırma, uzaklaştırma ve kaydırma işlemleri	71
4.5.3 Hesaplamaların yaptırılması.....	71
4.5.4 Grafiklerin çizdirilmesi	72
4.5.5 Analiz modu	74
4.5.6 Çizgi ekleme.....	78
4.5.7 Çizgi silme.....	78
4.5.8 Akım çizgileri referans doğrusu yerinin değiştirilmesi.....	78
4.5.9 Seçilen grid noktalarının katı cisim yapılması	79
4.5.10 Büyüklük kontur çizgilerinin tekrar çizdirilmesi	79
4.5.11 Ekran tuşları yardım sayfası.....	80
4.5.12 Hakkında penceresi	81
4.6 Deneysel Çalışma	81
4.6.1 Deney verileri.....	85
4.6.2 Deney verilerinin demavis ile çizilmesi	85
5. SONUÇ VE ÖNERİLER.....	91
5.1 Çalışmanın Uygulama Alanı	91
KAYNAKLAR.....	93
EKLER.....	95
ÖZGEÇMİŞ	106

KISALTMALAR

İTÜ	: İstanbul Teknik Üniversitesi
CTA	: Constant Temperature Anemometer
CCA	: Constant Current Anemometer
PIV	: Particle Image Velocimetry

ÇİZELGE LİSTESİ

	<u>Sayfa</u>
Çizelge 2.1 : CTA3D veri formatı.	10
Çizelge 2.2 : PIV3D veri formatı.....	13
Çizelge 2.3 : Aeroprobe3D veri formatı.	15
Çizelge 3.1 : Grid noktası veri yapısı.	35
Çizelge 3.2 : Bir grid noktasında hesaplanabilecek büyüklükler.	36
Çizelge 4.1 : Çizim veri yapısı.	50
Çizelge 4.2 : Potansiyel akım veri örneği.	53
Çizelge 4.3 : Programda kullanılan kütüphaneler.....	62
Çizelge 4.4 : Deney verilerinden bir kesit.	85

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : Sabit akım anemometre devresi (Perry, 1982).	3
Şekil 2.2 : Tipik bir sabit sıcaklık tel anemometre şeması (Goldstein, 1996).	4
Şekil 2.3 : Tele bağlı koordinat sistemi ve hız bileşenleri (Bruun, 1995).	5
Şekil 2.4 : Sıcak Tel probu (Kurtuluş, 2010).	6
Şekil 2.5 : Sıcak tel anemometresi prob örnekleri (Goldstein, 1996).	6
Şekil 2.6 : Film prob.	6
Şekil 2.7 : Fiber Film prob.	7
Şekil 2.8 : PIV sisteminin deneysel düzeneği (Raffel, 1998).	11
Şekil 2.9 : Travers üzerinde PIV lazer kaynağı.	11
Şekil 2.10 : PIV Deney odası altındaki iki adet CCD kamera.	12
Şekil 2.11 : PIV sistemi kontrol üniteleri.	12
Şekil 2.12 : 7 Delikli basınç probu.	14
Şekil 2.13 : Hızlı cevaplı prob, bağlantı kablosu ve destek modülü.	14
Şekil 2.14 : Hızlı cevaplı prob örnekleri.	15
Şekil 3.1 : İki farklı akım elemanının izledikleri yollar (Anderson, 2007).	17
Şekil 3.2 : Akım çizgileri ve hız vektörleri (Anderson, 2007).	18
Şekil 3.3 : Daimi akışta akım çizgisi ve yörüngesi (Anderson, 2007).	18
Şekil 3.4 : Akımın iki boyutlu kartezyen düzleminde denklemi.	19
Şekil 3.5 : Farklı akım çizgileri.	20
Şekil 3.6 : Δn boyunca debi Δy ve $-\Delta x$ arasındaki debi toplamına eşittir.	21
Şekil 3.7 : Deney verilerinin oluşturduğu grid.	23
Şekil 3.8 : B ve D noktalarındaki değerlerin hesaplanması.	24
Şekil 3.9 : Hücre içinde akım çizgisinin çizilmesi.	25
Şekil 3.10 : İki nokta arasında enterpolasyon.	26
Şekil 3.11 : Bir hücredeki akım çizgisi.	26
Şekil 3.12 : Üçgen eleman üzerinde yüzey enterpolasyonu.	27
Şekil 3.13 : Akım çizgileri ve referans değerleri.	28
Şekil 3.14 : Bir akışkan elemanın dönmesi (Anderson, 2007).	28
Şekil 3.15 : Hız vektörleri ve bunlara dik potansiyel eğrileri.	32
Şekil 3.16 : Akım çizgileri ve bunlara dik potansiyel eğrileri.	33
Şekil 3.17 : Grid hücresinde girdaplılık hesabı için kullanılacak büyüklükler.	33
Şekil 3.18 : Ölçüm noktalarını gösteren tipik bir grid.	35
Şekil 3.19 : Tekil X ve Y zincir yapıları.	37
Şekil 3.20 : Zincir veri yapısı.	37
Şekil 3.21 : Ölçüm noktalarının grid matrisine yerleştirilmesi.	37
Şekil 3.22 : Daire konturu.	38
Şekil 3.23 : Daire konturunun belirlenmesi.	39
Şekil 3.24 : Dikdörtgen konturu ve küçülmüş bir hücre elemanı.	40
Şekil 3.25 : Üçgen konturu ve küçülmüş bir hücre elemanı.	40
Şekil 3.26 : Elips denklem konturu.	40
Şekil 3.27 : Cisim konturu içindeki noktaların belirlenmesi.	41

Şekil 3.28 : Genel hücre formu.	42
Şekil 3.29 : Hücresinin üçgenlere bölünmesi.	43
Şekil 3.30 : Noktayı çevreleyen üçgen.	43
Şekil 3.31 : Her yöne tarama yöntemi ile numaralandırma.	44
Şekil 4.1 : Silindir etrafındaki potansiyel akışa ait hız vektörleri.	54
Şekil 4.2 : Demavis tarafından çizilmiş potansiyel akışa ait hız vektörleri.	55
Şekil 4.3 : Silindir etrafındaki potansiyel akımda akım çizgileri.	56
Şekil 4.4 : Demavis programının oluşturduğu akım çizgileri.	57
Şekil 4.5 : TechPlot programı tarafından oluşturulmuş akım çizgileri.	58
Şekil 4.6 : Demavis programı tarafından oluşturulmuş akım çizgileri.	59
Şekil 4.7 : Girdap akışına ait hız vektörleri ve eş hız eğrileri.	60
Şekil 4.8 : Girdap konturları.	61
Şekil 4.9 : Demavis programının derlenmesi.	63
Şekil 4.10 : Demavis programının çalıştırılması.	64
Şekil 4.11 : Demavis boş açılış ekranı.	65
Şekil 4.12 : Demavis'in ana menüsü.	65
Şekil 4.13 : Kullanılacak veri tipinin seçilmesi.	66
Şekil 4.14 : Kullanılacak verinin tanıtıldığı "Data Settings" penceresi.	66
Şekil 4.15 : Veri dosyası seçimi için tarayıcı pencere.	66
Şekil 4.16 : Ölçüm ve katı cisim noktalarının çizimi.	67
Şekil 4.17 : Katı cisimler için tanımlama satırları.	67
Şekil 4.18 : Cisim sayısının kullanıcı tarafından girilmesi.	67
Şekil 4.19 : Katı cisim tipleri.	68
Şekil 4.20 : Demavis daire tanımlama ekranı.	68
Şekil 4.21 : Demavis dikdörtgen tanımlama ekranı.	69
Şekil 4.22 : Demavis üçgen tanımlama ekranı.	69
Şekil 4.23 : Demavis denklem ile cisim tanımlama ekranı.	70
Şekil 4.24 : Demavis denklem dosyası okuma ekranı.	70
Şekil 4.25 : Hesaplama işlemlerinin sonu.	71
Şekil 4.26 : Demavis çizilebilecek grafikler menüsü.	72
Şekil 4.27 : Programda varsayılan olarak çizilen büyüklükler.	73
Şekil 4.28 : Hız vektörleri ve akım çizgileri bir arada.	74
Şekil 4.29 : Seçilen noktadaki akım çizgisi ve hız alanı eklenmiş görünüm.	75
Şekil 4.30 : Hücre referans noktasına ait tüm büyüklükler.	75
Şekil 4.31 : Bilgi menüsü.	76
Şekil 4.32 : Grid genel bilgileri.	76
Şekil 4.33 : Program içindeki değişkenlerin izlenmesi.	77
Şekil 4.34 : Akım çizgileri referans noktaları doğrusu.	78
Şekil 4.35 : Grid noktası silme modu.	79
Şekil 4.36 : Büyüklük kontur çizim parametreleri.	80
Şekil 4.37 : Tuş kısayol ekranı.	80
Şekil 4.38 : Demavis hakkında penceresi.	81
Şekil 4.39 : Demavis geliştirici ekranı.	81
Şekil 4.40 : Sıcak tel anemometre ölçüm ve traverse kontrol ünitesi.	82
Şekil 4.41 : Jet deneyinde kullanılan eliptik lüle.	82
Şekil 4.42 : Sıcak tel probu kalibrasyon ünitesi.	83
Şekil 4.43 : Deney düzeneğine alttan bakış.	83
Şekil 4.44 : Deneyde kullanılan hava jet masası.	84
Şekil 4.45 : Deneyde kullanılan sıcak tel sensörü.	84
Şekil 4.46 : Deneyde kullanılan travers sistemi.	85

Şekil 4.47 : Deney verilerinin Demavis'e tanıtılması.....	86
Şekil 4.48 : Deney hız dağılımı ve hız alanı renkleri.	86
Şekil 4.49 : Deney hız konturları.....	87
Şekil 4.50 : Deney türbülans konturları.....	88
Şekil 4.51 : Deney info çıktısından kesit.....	88
Şekil 4.52 : XZ düzleminde hız alanı ve hız büyüklükleri.	89

İTÜ TRİSONİK LABORATUARI AEORODİNAMİK DENEY SONUÇLARININ ANALİZİ VE GÖRSELLEŞTİRİLMESİ

ÖZET

Bu çalışmada, İTÜ Trisonik laboratuvarında yapılan rüzgar tüneli deneylerinde elde edilen veriler kullanılarak incelenen akışa ait büyüklüklerin çizimlerinin oluşturulması için algoritmalar ve bunları uygulayan bir çizim programı geliştirmek hedeflenmiştir.

Laboratuar ortamında ölçülen veri bir düzlem yada uzaysal bir hacim içerisinde tanımlanmış noktalara karşılık ölçülen büyüklüklerden oluşmaktadır. Bu çalışmada ele alınan büyüklükler; ölçülen hız, toplam basınç, statik basınç ve türbülans verileri ve hız alanının yapısına bağlı olarak elde edilen akım ve girdaplılık değerleridir.

Geliştirilen programda hız vektörleri, eş hız eğrileri, eş basınç eğrileri, akım çizgileri ve girdaplılık çizimleri çizilmekte ve analiz modunda bunların noktasal incelemeleri görsel ve interaktif olarak gerçekleştirilmektedir.

Bu tez kapsamında öncelikle akım ve girdaplılık fonksiyonları kullanılarak çizim algoritmaları geliştirilmiştir. Çizim algoritmasında temel olarak ölçümün yapıldığı noktalarda analitik fonksiyona uyumlu şekilde akım ve girdaplılık değerleri hesaplanmıştır. İki noktalı lineer ve üçgen elemanlarla da yüzey enterpolasyonları yapılarak çizimler yapılmıştır.

Programda kullanılan algoritmalarla elde edilen çizimler mevcut programlarla ve potansiyel akış kullanılarak elde edilen teorik akışla kıyaslanmış ve uyumlu sonuçlar elde edildiği görülmüştür.

Ayrıca, bu çalışma kapsamında sabit sıcaklık anemometresi ile bir sesaltı jet akışı içerisinde hız ve türbülans ölçümleri gerçekleştirilerek bir örnek veri seti elde edilmiştir. Sonuçlar çalışmada geliştirilen çizim programı vasıtasıyla incelenmiştir.

Demavis programı hız, performans, çeşitli platformlarda çalışabilirlik, geliştirmeye açıklık, kolay kurulum ve kullanım özellikleri gözetilerek geliştirilmiştir. Bu amaçlar için 64 bit'lik açık kaynak kodlu geliştirme araçları ve standart C dili tercih edilmiştir. Derleyici olarak 64 bit GNU C kullanılmıştır. Çizimler OpenGL ile gerçekleştirilmiştir, Kullanıcı arayüzü olarak GTK kütüphanesi kullanılmıştır. Elde edilen program 64 bit işlemcili Linux ve Windows makinalarda yönetici yetkisi gerektirmeden kopyalama yoluyla çalışmaktadır. Uygulama yaklaşık 9900 satır kod'dan oluşmakta ve binary büyüklüğü 300 KB civarındadır.

Program çalıştırıldığında seçilen yüzey için tüm büyüklüklere ait çizim noktalarını hesaplamakta ve hızlı işleyiş için hafızada tutmaktadır. Sonrasında, yaklaştırma, kaydırma, çizimleri birlikte ve ayrı gösterme vb çeşitli fonksiyonallikleri OpenGL ve diğer kütüphaneler vasıtası ile interaktif olarak hızlı bir şekilde gerçekleştirilmektedir.

Bu çalışma ile deney verilerinde incelenen akım ve girdaplılık çizgilerinin bilgisayar ortamında oluşturulması için kullanılabilir algoritmalar ve bunların yanısıra diğer büyüklüklerin detaylı incelenebileceği bir uygulama elde edilmiştir. Bu programın yapılan deneyleri görsel ve sayısal olarak incelemekte faydalı olacağı düşünülmektedir, ayrıca ihtiyaç duyulan diğer fonksiyonaltelerın eklenmesi için de bir altyapı oluşturulmuştur.

ANALYSIS AND VISUALISATION OF İTÜ TRISONIC LABORATORY AERODYNAMIC EXPERIMENTS

SUMMARY

In this work, it was aimed to develop algorithms to visualise physical properties of the values in a flow obtained in the experiments performed at İTÜ Trisonic Wind Tunnel Laboratories, and to use these algorithms in a computer program to draw their graphics.

The data measured at laboratory conditions consist of corresponding values of the measurements of the points defined in a surface or three dimensional volume. The values considered at this work are measured velocity, total pressure, static pressure and turbulence values and the values of stream and vorticity obtained according to the structure of the flow area.

Velocity vectors, velocity contours, total and static pressure contours, streamlines and vorticity contours are drawn by the program. Further analysis of these drawings can be made in analysis mode visually and interactively.

The experiments systems which are examined in this work are constant temperature anemometer (CTA), particle image velocimetry (PIV) and multi-hole aeroprope. CTA system consists of a probe which is made usually by tungsten or platinum. This probe is the resistance of an electrical circuit and heated at a constant temperature by the electrical current. The temperature of the hot-wire sensor decreases by the air flow, resulting a voltage decrease in the circuit. The amount of electrical voltage required to compensate this decrease is used to define the velocity of the flow. The fast response frequency makes this system suitable for accurate air flow velocity and turbulence measurement. This system must be calibrated before each measurement and the actual measurement must be done at the same environment temperature with calibration in order to obtain reliable and accurate results.

At this coverage, stream and vorticity functions were used to develop algorithms in order to use in drawings. In the drawing algorithm, it was calculated the stream and vorticity values at the grid points where measurements were performed, in a similar way with the analytical functions. Two points linear and three points triangular area interpolations are used in order to perform drawings.

In order to use the numerical algorithms, some other geometrical algorithms were also required such as determination of object's shape and grid points intersection points which are called contour of the object, determination of grid points inside the object boundaries, determination of grid cell corner points and determination of the smallest triangular area covering the point in question. The contour points of the object's shape were determined by substituting one of the grid coordinates in the shape function of the object and comparing if the result value is between two adjacent grid points. If this condition was met then it was determined that the point in interrogation is a contour point. The same process repeated for other coordinate in order to determine the other contour point along the other axis. All the

grid points are tested to determine all the contour points of the object shape. The same approach is used to determine the grid points which are inside the object shape. Again by substituting the coordinate of a grid point it was obtained two values. If the grid point is between these values then this point is inside the shape function.

This approach defined for algebraic shape functions especially used for circular and elliptic objects, was applied for other geometrical objects such as rectangle and triangle.

The results obtained with the program were compared with the present applications at the Laboratory, with a stream obtained by using a theoretical potential flow around a cylinder and with a vortex flow. The results obtained with potential flow are very accurate and the max. error observed as 0.01%. Similar streamlines obtained with current applications, and it was obtained some differences at edges of the vortex flow which was caused by the sudden changes in this flow.

It was also obtained an example experimental result set of velocity and turbulence values of a subsonic jet flow by using a hot wire anemometer. Results are examined with the developed drawing software.

The program Demavis was developed with the objectives of speed, performance, portability, easy installation and user friendliness. To reach these objectives, 64 bits open source development tools and standart C language was preferred. 64 bits GNU C compiler was used. Drawings were performed with high performance OpenGL libraries. User interface was developed with GTK library. The resulting code runs on 64 bits Linux and Windows operating systems without requiring administrative rights and simply by copying. The code consists of 9900 lines and binary size is 300 KBytes.

While using the program Demavis, the user first chooses the data type that will be examined among the data types that program currently supports: PIV2/3D, CTA2/3D and Aeroprobe2/3D. If the data is for an experiment for 3 dimensions then the interrogation surface and section is choosen by using the dynamically changing user interface with mouse. Then the actual data file is choosen with a standart file browser. The grid and the obstacle points are displayed at the screen. If the data file belongs to a PIV experiment then objects in the flow are automatically determined by Demavis. For other data types user informs the number of object in the experiment. Solid objects in the experiments can be defined by a circle, rectangle, triangle and algebraic equation. The reference values of these objects are given into the application and on screen interactive displacements, enlargement and diminution of the shape can also be done.

When the program starts, the drawing points of all properties were calculated for the selected surface and they remain in memory for fast drawing operations. Then zoom, pan, together and standalone drawings, and other functionalities are performed with OpenGL libraries quickly and interactively.

The program can make 34 different drawings, they can be seen separately or together as desired: grid points, obstacle, shape, velocity vectors, streamlines, resultant velocity contours, velocity areas, u velocity contours, v velocity contours, w velocity contours, vortex contours, turbulence contours, turbulence values, static pressure contours, total pressure contours. The interactive drawings of these values can be made by the “selected” versions of them at the render menu. The “analysis” option of these values permits the detailed and interactive drawings. All the values of grid

point can be seen at the analysis mode and number, min. and max. values of contour lines can be changed again in analysis mode. Further lines can also be added to default lines. All the general properties of experiment such as min, max values and size of the values can be seen by using the info menu of Demavis. Program displays informational messages at every C function when entering and leaving; this facilitates to follow what the program does at each stage and can help other developers to easily add new functionalities.

With this work, it was developed usable algorithms for visualising on computer several flow properties i.e. streamlines, vorticity lines, velocity contours and pressure contours and a program that use these algorithms with the functionality of studying with details all the properties of the flow.

This program can be used to study the experimental results in a visual and interactive way. Other features that may be needed can be developed by using this work as the base.

1. GİRİŞ

İTÜ Trisonik laboratuvarında çeşitli yöntemlerle akım karakteristiklerinin gözlenmesi ve ölçülmesi için deneyler yapılmaktadır. Burada elde edilen verilerin değişik uygulamalarla analizi ve çizimleri yapılmakta fakat karşılaşılan çeşitli kısıtlar uygulama zorluğu oluşturmaktadır. Bu zorlukların aşılması için alternatif yöntemler geliştirilmesi analiz sürecini daha efektif hale getirecektir.

1.1 Tezin Amacı

İTÜ Trisonik laboratuvarında yapılan rüzgar tüneli deneylerinde elde edilen verilerin bilgisayar ortamında çizimlere dönüştürülmesi için ilgili analitik fonksiyonlar kullanılarak algoritmalar oluşturulması ve bunların bir bilgisayar programında etkin ve kullanıcı etkileşimli olarak kullanılacağı bir uygulama geliştirilmesi hedeflenmiştir.

2. DENEY SİSTEMLERİ

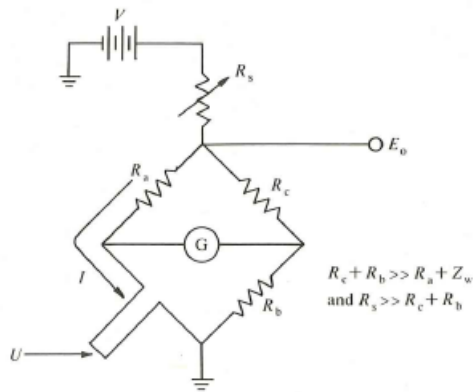
İTÜ Trisonik Laboratuvarında kullanılmakta olan deney ölçüm sistemlerinden, bu çalışma kapsamında verileri değerlendirilen Parçacık Görüntü Hızölçeri (PIV: Particle Image Velocimetry), Sıcak Tel Anemometresi ve Çok Delikli Basınç Ölçüm Probu (Multi-hole Aeroprobe) hakkında bilgiler aşağıda verilmiştir.

2.1 Sıcak Tel Anemometresi

Termal anemometre, akımın içine yerleştirilmiş elektrikle ısıtılan küçük bir sensörde akışkan akımı nedeniyle oluşan ısı transferine bağlı olarak akım hızının belirlenmesi prensibine göre çalışmaktadır. Bu sistemin başlıca iki yöntemi vardır;

- Prob sıcaklığının değiştiği sabit akım modu CCA (Constant Current Anemometer).
- Prob sıcaklığının sabit kaldığı voltajın değiştiği yöntem CTA (Constant Temperature Anemometer).

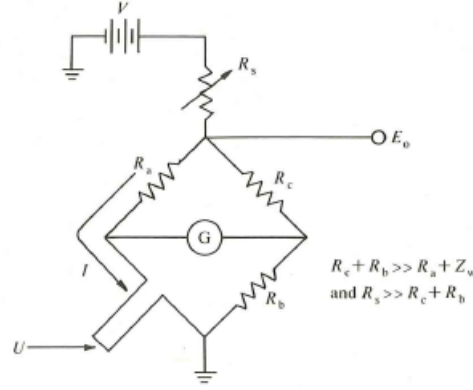
CCA yapısında kullanılan tipik bir Wheatstone köprüsü şeması **Şekil 2.1**'de gösterilmiştir.



Şekil 2.1 : Sabit akım anemometre devresi (Perry, 1982).

Transistörlü devrelerin gelişimine kadar sabit akım devreli sıcak tel anemometreleri yaygın olarak kullanılmaktayken, 1960 yıllarından itibaren sıcak tel anemometresi C yaygınlaşmaya başlamıştır. Bu sistemde, akışkan akımı telde ısı transferine yol

açarak telin sıcaklığının düşmesine neden olmaktadır. Buna bağlı olarak da direnç düşmekte ve elektrik voltajında azalma olmaktadır. Tel aynı sıcaklıkta tutulacak şekilde voltaj arttırılmaktadır. Hız ölçümü bu voltaj değerleri kullanılarak yapılmaktadır. **Şekil 2.2**'de tipik bir sabit sıcaklık tel anemometresinin prensip şeması görülmektedir.



Şekil 2.2 : Tipik bir sabit sıcaklık tel anemometre şeması (Goldstein, 1996).

Bu sistemde yüksek kazançlı amplifikatörden gelen geri besleme girişi verilerek sensör sıcaklığı ve dolayısıyla direncin sabit kalması sağlanmaktadır. Sistemin adım adım işleyişi aşağıda verilmiştir;

- Sıcak tel üzerinden geçen akışkanın hız artışı sıcak telin soğumasına, sıcaklığının düşmesine, R direncinin azalmasına ve I noktasında voltajın azalmasına yol açar.
- Amplifikatörün negatif girişindeki voltaj düşümü U_{12} giriş voltajının artmasına yolaçar.
- Amplifikatörün yükselmiş U_{12} giriş voltajı amplifikatörün U çıkış voltajını yükseltir.
- Köprüdeki yükselmiş U voltajı sensöre doğru olan akımı yükseltir.
- Yükselmiş akım sensörü ısıtır, buna bağlı olarak sistem tekrar dengeye gelene kadar U_{12} voltajı düşer.

Tüm bu süreç neredeyse anlık olarak gerçekleşir, böylelikle hızdaki bir artış U voltajında da bir artış olarak görülür. Bu voltaj genel olarak anemometrenin çıkışı olarak kullanılır.

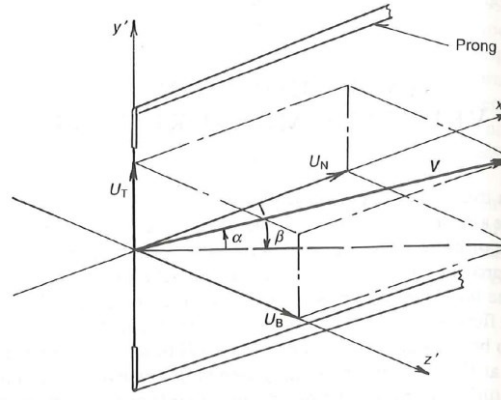
Sıcak tel sensörünün çıkış voltajı E basit güç yasası kanunu formunda

$$E^2 = A + BV_e^n \quad (2.1)$$

ifade edilebilir (Bruun, 1995). Burada V_e etkin hızı belirtmektedir, A ve B kalibrasyon sabitleridir. Etkin hız **Şekil 2.3**'te gösterilen hız bileşenlerinden

$$V_e^2 = U_N^2 + k^2 U_T^2 + h^2 U_B^2 \quad (2.2)$$

ifadesiyle elde edilmektedir. K ve h sıklıkla sensörün sapma ve yunuslama katsayıları olarak tanımlanmaktadır.

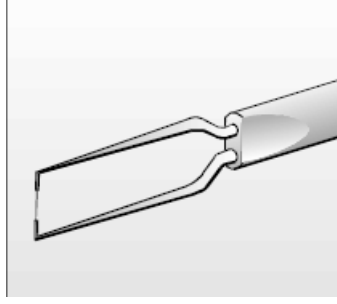


Şekil 2.3 : Tele bağlı koordinat sistemi ve hız bileşenleri (Bruun, 1995).

Sıcak tel anemometresinde sensörlerin küçük boyutları ve yüksek frekanslı cevapları bu sistemi özellikle türbülanslı akış detaylarının incelenmesine elverişli hale getirmektedir. CTA sistemleri kullanılırken her ölçümden önce kalibrasyon yapılması gerekmektedir. Kalibrasyon ortam sıcaklığı ile deney ortam sıcaklığının aynı veya çok yakın olması durumunda deney sonuçları güvenilirdir (Gümüş, 2002).

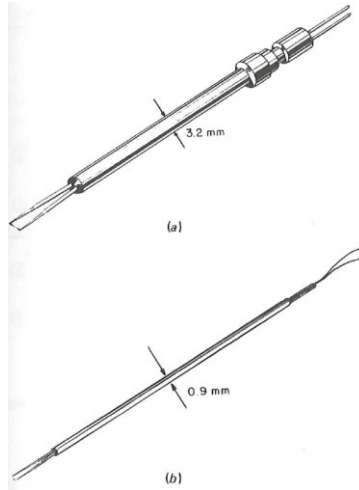
2.1.1 Sıcak tel sensörleri

Yaygın olarak sıcak tel sensörleri tungsten, platin ve platin-iridium alaşımlarından yapılmaktadır. **Şekil 2.4**'te şematik bir sıcak tel probu görülmektedir. Tungsten yüksek sıcaklık direnç katsayısı ve yüksek dayanıklılık özellikleri nedeniyle tercih edilmektedir. Ana dezavantajı özellikle 300 derecenin üstündeki sıcaklıklarda oksitlenme oranıdır. Tungsten teller ticari olarak 2,5 μm çaptan başlayarak sunulmaktadır. Platin 0,5 μm gibi küçük çaplardan başlayarak bulunabilmektedir. İyi bir sıcaklık katsayısına sahiptir ve oksitlenmemektedir. Yüksek sıcaklık ve yüksek hava hızlarında tek başına aerodinamik sürüklenme kuvveti platin telin limitlerinde gerilmeler oluşturmaktadır.



Şekil 2.4 : Sıcak Tel probu (Kurtuluş, 2010).

Platin-iridium oksitlenmeyen, platine göre daha yüksek dayanımlı fakat düşük sıcaklık direnç katsayısına sahiptir. Kullanım alanı, tel sıcaklığı tungsten için çok yüksek ve platinin çok zayıf kaldığı uygulamalar olarak ortaya çıkmaktadır. **Şekil 2.5'**de sıcak tel anemometresi prob örnekleri görülmektedir.



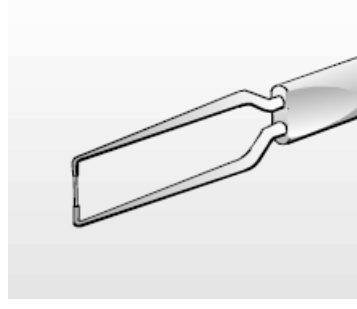
Şekil 2.5 : Sıcak tel anemometresi prob örnekleri (Goldstein, 1996).

Film sensörler 1955 yılında ortaya çıkmış ve sensör teknolojisinde büyük bir gelişme olmuştur. Film sensörler parçaçık içeren sıvı ve gazların ve özellikle taşınım yapan su gibi sıvıların ölçümlerinde faydalı olmaktadır. **Şekil 2.6'**da film sensör görülmektedir (Kurtuluş, 2010).



Şekil 2.6 : Film prob.

Şekil 2.7'de fiber film prob sensör görülmektedir.



Şekil 2.7 : Fiber Film prob.

2.1.2 Sıcak tel anemometre donanımı

İTÜ Trisonik laboratuvarında kullanılan sıcak tel anemometresi aşağıdaki ünitelerden oluşmaktadır:

1. StreamLine 90N10 ana ünite
2. StreamLine 90N10 CTA modülü
3. StreamLine 90H10 kalibrasyon sistemi
4. StreamLine 90H01 kalibrasyon modülü
5. StreamLine 90H02 akış ünitesi
6. StreamLine 90H04 hava filtreleme ünitesi
7. StreamLine 55P11 genel amaçlı prob
8. StreamLine 90S10 StreamWare yazılım programı
9. StreamLine 41T43 3D travers sistemi

2.1.3 Güçlü yanlar, sınırlar ve lazer hızölçer ile karşılaştırma

Bir akışkan içinde hız detaylarını ölçebilmek için ideal enstrümanın aşağıdaki özelliklere sahip olması gerekmektedir:

1. Hız geçişlerini yüksek hassasiyetle takip edebilmek için yüksek frekanslı cevaba sahip olmalıdır.
2. Noktasal ölçüm için boyut olarak küçük olmalıdır.
3. Geniş bir hız aralığını ölçebilmelidir.
4. Yalnızca hızı ölçmeli ve geniş bir sıcaklık, yoğunluk ve karışım aralığında çalışabilmelidir.

5. Hız bileşenlerini ölçebilmeli ve ters akışları tespit edebilmelidir.
6. Yüksek doğruluğa sahip olmalıdır.
7. Yüksek çözünürlüğe ve düşük gürültü oranına sahip olmalıdır.
8. Minimum akım bozulmasına yol açmalıdır.
9. Maddi olarak ucuz olmalıdır.
10. Kullanımı kolay olmalıdır.

Uzun yıllar boyunca sadece sıcak tel anemometresi türbülans çalışmalarını derinlemesine yapmak için bu kriterlerinden yeterincesini sağladı. Pitot problemleri, akım görüntüleme ve diğer teknikler tamamlayıcı olarak kullanıldılar, fakat sıcak tel kadar detay sağlayamadılar.

Pekçok uygulama için 4, 5, 6 ve 10. maddeler bu yöntemin belki de en zayıf noktalarıdır. Esasen, ilk pratik limit, kırılma ve kirlenmeye karşı duyarlılık olarak ortaya çıkmaktadır. Teorik olarak hızın doğrudan değil de çıkarımsal olarak ölçülmesi ve konveksiyon dışındaki ısı transferleri de hataya yol açmaktadır.

Aşağıda sıcak tel anemometresi ve lazer hızölçer (lazer kullananılan, lazer dopler anemometre ve parçacık görüntü hızölçeri birlikte değerlendirilmiştir) aşağıdaki 10 temel kriter açısından karşılaştırılmıştır.

1. Yüksek frekanslı cevap: Mevcut uygulamalarda, sıcak tel anemometresi kesinlikle daha yüksek frekanslara sahiptir. Yüzlerce kilohertz'e ulaşmak kolaylıkla mümkün ve 1 Mhz'e çıkmak da mümkündür.
Teorik olarak lazer hızölçerinde sıcak tel anemometresinin hızlarına ulaşmak mümkün olsa da pratikte 30 kHz ölçülmüştür.
2. Boyut: Bir sıcak tel sensörü tipik olarak $5\mu\text{m}$ çapında 2 mm uzunluğundadır.
 $1\mu\text{m}$ çap ve 0,2 mm uzunluk elde etmek mümkündür.
Lazer hızölçerinde ölçüm alanı $50\mu\text{m} \times 0,25\text{mm}$ ebatındadır. Çok küçük test kesitlerinde $5 \times 5\mu\text{m}$ ölçüm hacmi elde etmek mümkündür. Ölçüm noktası odaklama merceğinden 40 mm'den daha uzakta olduğunda küçük ölçüm hacimleri elde etmek zordur.

3. Hız aralığı: İki teknik de çok geniş hız yelpazesine sahiptir. Lazer çok düşük hızlarda avantaja sahiptir. Sıcak teli etkileyen serbest taşınım etkisi genellikle bir problem değildir.
4. Yalnızca hızı ölçmeli ve geniş bir sıcaklık, yoğunluk ve karışım aralığında çalışabilmelidir: Lazer hızölçer yalnızca parçacıkların hızını ve ancak belirli bir yönde ölçebilir.
İki sistem de geniş bir sıcaklık yelpazesinde çalışabilir. Sıcak telin maksimum sıcaklığı sensörün maksimum çalışma sıcaklığı ile sınırlanır.
Yüksek yoğunlukta ölçüm iki sistem için de zordur.
5. Hız bileşenlerini ölçebilmeli: Sıcak tel anemometresinde akış alanındaki bir, iki ve üç bileşeni de ölçebilmek için sırasıyla bir, iki ve üç sensör kullanılabilir. Film sensörler herhangi bir türbülans yoğunluğunu ölçme potansiyeli sağlamakta fakat mevcut tekniklerin karmaşıklığı ve prob boyutu uygulamayı kısıtlamaktadır.
Lazer hızölçer bileşenleri ölçebilir, frekans kaydırma ile ters akımları da tespit edebilir. Üçüncü bileşeni elde etmek zor olsa da bunu sağlayabilen sistemler mevcuttur.
6. Yüksek doğruluk: Sıcak tel sonuçları tekrar üretilebilir, sonuçların kesinliği kalibrasyon şartlarına ne kadar yaklaşıldığının bir fonksiyonudur. Pratikte, kirlilik, sıcaklık değişiklikleri ve diğer faktörler doğruluğu yüzde birkaç puan etkilemektedir.
Lazer hızölçer dikkatle kontrol altında tutulan deneylerde %0,1 gibi yüksek doğruluk sağlamaktadır. Fakat pratik olarak ölçümlerde, kırılma indeksi değişimi, sinyal işleme limitleri doğruluğun %1 mertebesinde gerçekleşmesine yol açmaktadır.
7. Çözünürlük: Sıcak tel, çok düşük gürültü seviyesine sahip olabildiği için net olarak daha yüksek bir çözünürlük sağlamaktadır. 10.000'de bir çözünürlüğü kolaylıkla elde edilebilmekteyken, lazerde 1000'de bir mevcut teknolojiyle zordur.

8. Akım bozulması: Sadece ışığın akıma girmesi gerektiğinden lazer net olarak daha iyidir. Kullanılması gereken parçacık boyut ve yoğunluğu akış alanında ölçülebilecek bir değişiklik yapmamaktadır.
9. Maliyet: Sıcak tel halen önemli bir mertebe daha düşük maliyetlidir. Fakat, yeni teknolojiler durumu ileride değiştirebilir.
10. Kullanım kolaylığı: Mevcut durumda, lazer hızölçerin kurulması ve geçerli veri alınmaya başlanması muhtemelen daha zordur. Kurulum yapıldıktan sonra ise, hassas, kirlenebilecek, kırılabilir veya yeniden kalibrasyon gerektirecek bir sensör olmadığı için lazer hızölçerin kullanımı daha kolaydır. Lazer hızölçerin verilerinin değerlendirilmesinde ayrı ölçümlerin neden olduğu kesintili çıktılar bir zorluk oluşturmaktadır.

Genel bir kural olarak, Pitot tüpleri veya basınç dönüştürücüler gibi enstrümanlar gereken ölçüm detaylarını sağlayamıyorsa, sıcak tel anemometresinin kullanımı değerlendirilmelidir. Yüksek sıcaklıklar, akış içinde hareketli objeler, akışta kirlilik, yüksek türbülans yoğunluğu, vb. problemler sıcak tel anemometre kullanımını zor veya imkansız kılabilir, bu durumlarda lazer hızölçeri kullanmak değerlendirilmelidir.

2.1.4 Veri formatı

Bu sistemde elde edilen nihai verinin 3 boyutlu koordinatlardaki yapısı **Çizelge 2.1**'de belirtilmiştir.

Çizelge 2.1 : CTA3D veri formatı.

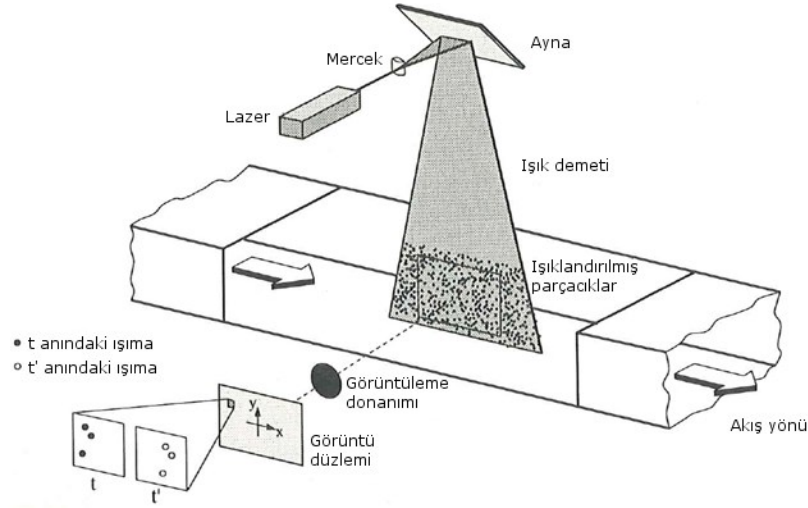
x[mm]	y[mm]	z[mm]	hız[m/s]	Türbülans
-------	-------	-------	----------	-----------

Veriler sırasıyla ölçüm noktası koordinatları, hız bileşenleri ve ölçüm noktası durum bilgisi değerlerinden oluşmaktadır.

2.2 Parçacık görüntü hızölçeri

Bu başlıkta “Particle Image Velocimetry” (PIV) olarak adlandırılan yöntem hakkında bilgiler verilecektir. PIV sisteminin deneysel kurulumu çeşitli alt sistemlerden oluşmaktadır. Uygulamaların çoğunluğunda takip edilecek parçacıkların akışa eklenmesi gerekmektedir. Bu parçacıkların bir akış düzleminde, kısa bir zaman

aralığında en az iki kez ışıklandırılması gerekmektedir. Parçacıklardan yayılan ışın tek bir fotoğrafta veya fotoğraf kareleri dizisinde kaydedilmelidir. Parçacık görüntülerinin iki ışıma arasındaki hareketleri PIV kayıt işlemi sırasında tespit edilmelidir. PIV tekniği kullanarak büyük bir miktardaki verinin işlenebilmesi için karmaşık ve yetkin son işleme gerekmektedir. **Şekil 2.8**'de bir rüzgar tünelineki deney için tipik bir PIV sistemi görülmektedir.



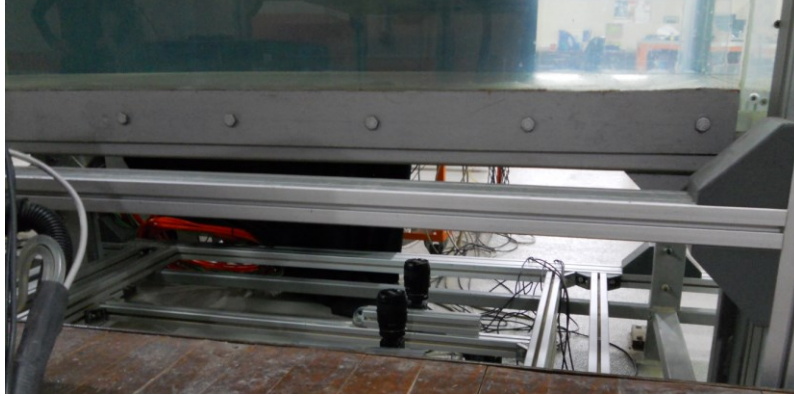
Şekil 2.8 : PIV sisteminin deneysel düzeneği (Raffel, 1998).

Küçük takip parçacıkları akışa eklenir. Akım içinde bir düzlem (ışık demeti) lazer ile iki kere (ışımalar arasındaki zaman aralığı akımın ortalama hızına ve akım şiddetine göre ayarlanır) aydınlatılır. **Şekil 2.9**'da laboratuarda bulunan deney düzeneği ve bir travers üzerinde sabitlenmiş lazer kaynağı görülmektedir.



Şekil 2.9 : Travers üzerinde PIV lazer kaynağı.

Takip parçacıklarının iki ışına arasında yerel akım hızıyla ilerledikleri kabul edilir. Takip parçacıklarından yayılan ışınlar yüksek kalitede bir objektif ile bir film negatifine veya özel çapraz korelasyon CCD (charge coupled device) sensöründe iki ayrı çerçevede kaydedilir. PIV fotoğraf filminin görüntüye dönüştürülmesinden sonra kayıt bir tarayıcı ile sayısallaştırılır. CCD sensörün ise çıktısı doğrudan gerçek zamanlı olarak bir bilgisayarın hafızasına kaydedilir. **Şekil 2.10**'da deney odasının altındaki kameralar görülmektedir.



Şekil 2.10 : PIV Deney odası altındaki iki adet CCD kamera.

Sayısal PIV kayıdı işlenmek üzere “sorgulama alanları” adı verilen küçük alt alanlara bölünür. Takip parçacıklarının her sorgulama alanında birinci ve ikinci ışınımdaki resimleri için yer değişim vektörleri istatistiki yöntemlerle (otomatik ve çapraz ilinti) belirlenir. Bir sorgulama alanındaki tüm parçacıkların iki ışına arasında homojen olarak ilerlemekte oldukları kabul edilmektedir. Yerel akım hız vektörünün ışık demeti düzlemine projeksiyonu iki ışına arasındaki zaman farkı ve görüntüdeki büyütme dikkate alınarak yapılmaktadır. **Şekil 2.11**'de PIV sistemi kontrol üniteleri görülmektedir.



Şekil 2.11 : PIV sistemi kontrol üniteleri.

Sorgulama prosesi PIV kayıdındaki tüm sorgulama alanları için tekrarlanır. Modern video kameralarla (1000x1000 sensör elemanlı) dakikada 100'den fazla PIV kaydı yapmak mümkündür. 3600 ani hız vektörlü bir video kaydının (kayıt boyutuna ve sorgulama alanlarının büyüklüklerine bağlı olarak) işlenmesi standart bilgisayarlarla birkaç saniye mertebesinde sürmektedir. Anında izleme için daha çok verinin hazır olması gerektiğinde bu proses için adanmış işlemcili donanımlar ticari olarak bulunmakta ve benzer sonucu bir saniyenin kesirlerinde yapabilmektedirler (Raffel, 1998).

2.2.1 Veri formatı

Bu sistemde elde edilen 3 boyutlu koordinatlarda verinin yapısı **Çizelge 2.2**'de belirtilmiştir.

Çizelge 2.2 : PIV3D veri formatı.

x[mm]	y[mm]	z[mm]	u[m/s]	v[m/s]	w[m/s]	Status
-------	-------	-------	--------	--------	--------	--------

Veriler sırasıyla ölçüm noktası koordinatları, hız bileşenleri ve ölçüm noktası durum bilgisi değerlerinden oluşmaktadır.

2.3 Çok delikli basınç probu

Akışkanlar mekaniğinde çeşitli çok delikli problar kullanılmaktadır. En yaygın kullanılmakta olan statik pitot tüpüdür, pitot tüpü ile yönü önceden belirlenmiş olan akımın hızı ölçülebilir. Üç delikli prob ile akım açısı ve akım hızı ölçülebilir. Beş delikli problar yaygın olarak hızın büyüklüğü, yönü, yerel statik ve toplam basınç ölçümünde kullanılırlar, fakat prob eksenine göre $55^{\circ} - 60^{\circ}$ dereceden küçük eğimlerde kullanılabilmektedirler. Yedi delikli problar da aynı amaçla fakat $70^{\circ} - 75^{\circ}$ derecelik açılara kadar kullanılabilmektedir (Aeroprobe, 2003).

Çok delikli problar dayanıklıdır ve üretim prosesinin sonunda her akım için bir kalibrasyona ihtiyaç duyarlar. Uygulama tekdüzedir ve akım hızında tipik olarak %1'den, akım açısında da %0,5'den daha iyi sonuçlar verir. Fakat, probların standart cevap frekansları tipik olarak 50 Hz'den daha düşüktür. Bununla birlikte, operasyonun doğası gereği akımda enterferans da oluştururlar. Çok delikli problar sağladıkları hassasiyetle akım incelemede maliyet açısından en uygun araçlardır.

Şekil 2.12’de Trisonik Laboratuar’ında kullanılmakta olan 7 delikli bir basınç probu görülmektedir.



Şekil 2.12 : 7 Delikli basınç probu.

Çok delikli problar tipik olarak prinç veya paslanmaz çelikten yapılmış uçlara sahiptir. Prob ucu 30°’lik yarım koni ve yarım küre şeklinde üretilirler. Tüpler arasındaki boşluklar hareketi engellemek için epoksi doldurulmaktadır. Probe desteği gövdesiyle bütün olacak şekilde imal edilmektedir. Kalibrasyon ve test sırasında probun sabitlenmesi önemlidir.

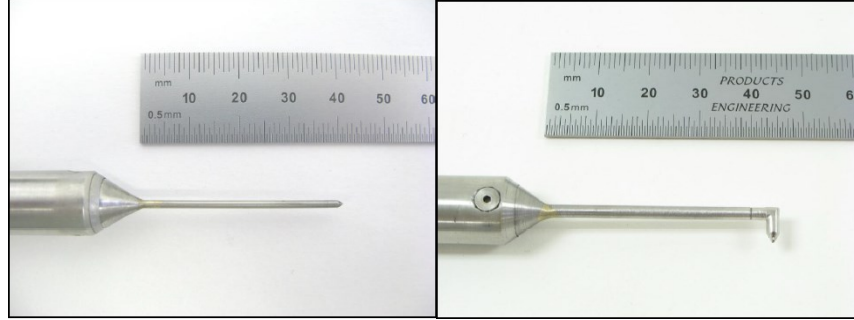
Hiçbir prob birbiriyle aynı üretilmeceği için tüm problemlerin kalibre edilmeleri gerekmektedir. Kalibrasyon sırasında prob rüzgar tünelinin içinde iki derecelik serbest açılarla pozisyonlama yeteneğine sahip mekanizmaya bağlanır. Prob ucu test alanında sabit kalacak şekilde kısıtlanır. Adımlama motoru probu serbest akım içinde daha önceden belirlenmiş yönlenmelere konumlandırır. Bu konumlandırmaların herbirinde çok delikli probun basıncı serbest akımın toplam basıncıyla birlikte kaydedilir. Toplanan veri prob kalibrasyon haritasını oluşturmada kullanılır. Bir kalibrasyon akım hızında prob pek çok açıda kalibre edilir. Şekil 2.13’de hızlı cevaplı prob, bağlantı kablosu ve destek modülünün resmi görülmektedir.



Şekil 2.13 : Hızlı cevaplı prob, bağlantı kablosu ve destek modülü.

İlave akış hızlarındaki kalibrasyon test verisi çeşitli Mach ve Reynolds sayılarındaki kalibrasyon haritaları ile enterpole etmek için kullanılabilir. Çoklu kalibrasyon için

önerilen Mach sayısı aralığı $\Delta M=0,1-0,15$ 'dir. Eğer probe düşük akım hızlarında (tipik olarak 17m/s'den küçük) kullanılacaksa ilave kalibrasyonlar Reynolds sayısı etkisini gidermek için gerekebilir. 5KHz civarında cevap süresi olan problar hızlı cevaplı prob olarak anılmaktadır. **Şekil 2.14**'de değişik tipte hızlı cevaplı prob resimleri görülmektedir (Aeroprobe, 2010).



Şekil 2.14 : Hızlı cevaplı prob örnekleri.

2.3.1 Veri formatı

Bu sistemde elde edilen 3 boyutlu verinin yapısı **Çizelge 2.3**'de belirtilmiştir.

Çizelge 2.3 : Aeroprobe3D veri formatı.

x[mm]	y[mm]	u[m/s]	v[m/s]	w[m/s]	Po[Pa]	Ps[Pa]	PVRedErr
-------	-------	--------	--------	--------	--------	--------	----------

Veriler sırasıyla ölçüm noktası koordinatları, hız bileşenleri, toplam ve statik basınçlar ile okuma hatası değerlerinden oluşmaktadır.

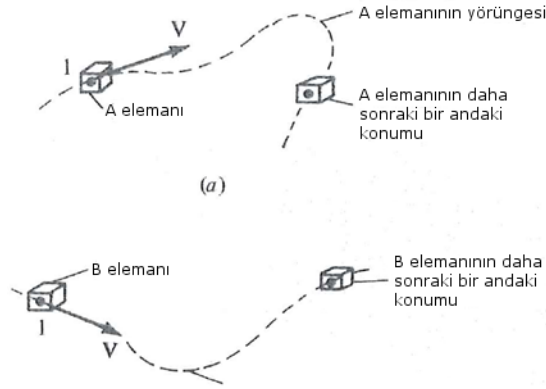
3. ALGORİTMALAR

Bu bölümde programda kullanılacak algoritmalara temel teşkil eden akım analitik fonksiyonları, bunlardan türetilmiş akım ve girdaplılık çizgileri hesaplama algoritmaları ve programlamalarında kullanılan uygulama algoritmaları anlatılacaktır.

3.1 Akım Çizgileri

3.1.1 Akım çizgisi fonksiyonu

Akımın ne şekilde ilerlediğini göstermek için akım yörüngeleri veya akım çizgileri çizilmektedir. Akım yörüngeleri akış elemanlarının farklı zamanlarda izledikleri yolları göstermektedir. **Şekil 3.1**'de iki farklı akım elemanının farklı zamanlarda izledikleri yollar görülmektedir.



Şekil 3.1 : İki farklı akım elemanının izledikleri yollar (Anderson, 2007).

Akım daimi olmadığı için 1 noktasındaki (ve akıştaki tüm diğer noktalarda) hız vektörü zamanla değişmektedir. Bu nedenle A ve B elemanlarının yol çizgileri aynı noktadan geçmiş olmalarına rağmen farklı eğriler şeklindedir. Genel olarak daimi olmayan akışta farklı akım elemanlarının aynı noktadan geçen yörüngeleri farklıdır.

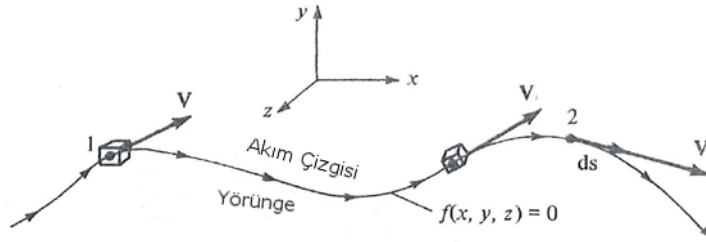
Akım çizgileri ise tanım olarak, herhangi bir noktada eğimi hız vektörü ile o noktada aynı yönde olan eğridir. **Şekil 3.2**'de bir kanat profili üzerindeki akım çizgileri ve hız vektörleri görülmektedir.



Şekil 3.2 : Akım çizgileri ve hız vektörleri (Anderson, 2007).

Daimi olmayan akıslarda, hız vektörleri zamana göre şiddet ve yön olarak değıştikleri için akım çizgisi yolları da değışik zamanlarda farklıdır. Daimi akımlarda ise bu değışkenlik bulunmadığı için akım çizgileri hep aynı akım yolundadır.

Genel olarak akım çizgileri yörüngelerinden farklıdır. Fakat daimi akıslarda tüm noktalarda hız vektörlerinin büyüklük ve yönleri zamanla değışmemektedir, bu nedenle aynı noktadan geçen farklı akım elemanlarının yörüngeleri aynıdır. Üstelik yörüngeleri ve akım çizgileri de eştir. Bu nedenle daimi akısta yörüngesi ve akım çizgisi arasında **Şekil 3.3**'te gösterildiği şekilde bir farklılık yoktur, bunlar aynı eğrilerdir. Bu akım çizgisi uzayda bir eğridir, bu nedenle $f(x,y,z)=0$ şeklinde bir denklem olarak yazılabilir.



Şekil 3.3 : Daimi akısta akım çizgisi ve yörüngesi (Anderson, 2007).

Bu denklemi elde etmek için ds elemanının **Şekil 3.3**'te 2 noktasında gösterildiği şekilde yönlenmiş bir akım elemanı olduğu kabul edilsin. 2 noktasındaki hız vektörü $\mathbf{V}$ olsun, bu durumda akım çizgisinin tanımından $\mathbf{V}$, $d\mathbf{s}$ 'ye paralel olur. Çapraz vektör çarpımı kullanılarak bu ifade

$$d\mathbf{s} \times \mathbf{V} = 0 \quad (3.1)$$

şeklinde yazılır ve geçerli bir akım çizgisi denklemidir. Kartezyen koordinatlarda genişletilerek vektörel çarpımı yapılarak

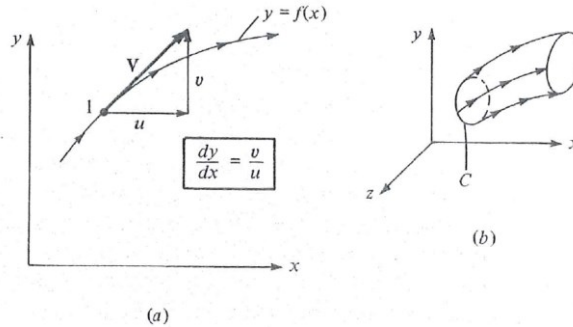
$$\begin{aligned}
d\mathbf{s} &= dx\mathbf{i} + dy\mathbf{j} + dz\mathbf{k} \\
\mathbf{V} &= u\mathbf{i} + v\mathbf{j} + w\mathbf{k} \\
d\mathbf{s} \times \mathbf{V} &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ dx & dy & dz \\ u & v & w \end{vmatrix} \\
d\mathbf{s} \times \mathbf{V} &= \mathbf{i}(w dy - v dz) + \mathbf{j}(u dz - w dx) + \mathbf{k}(v dx - u dy) = 0
\end{aligned} \tag{3.2}$$

bağıntıları elde edilir. Bu vektör sıfıra eşit olduğu için tüm terimlerinin sıfıra eşit olması gerekmektedir.

$$\begin{aligned}
w dy - v dz &= 0 \\
u dz - w dx &= 0 \\
v dx - u dy &= 0
\end{aligned} \tag{3.3}$$

Bu denklemler akım çizgisi için diferansiyel denklemlerdir. u , v ve w , x , y ve z 'nin fonksiyonu olarak bilindiğinde akım çizgisi denklemi olarak $f(x,y,z)=0$ şeklinde integre edilebilir.

(3.3) denkleminin fiziksel anlamını vurgulamak için **Şekil 3.4**'te gösterilen iki boyutta bir akım çizgisi ele alınsın. Bu akım çizgisinin denklemi $y=f(x)$ şeklinde yazılabilir.



Şekil 3.4 : Akımın iki boyutlu kartezyen düzleminde denklemi.

Akım çizgisinin 1 noktasındaki eğimi dy/dx 'dir. V , x ve y bileşenleri sırasıyla u ve v olmak üzere akım çizgisine 1 noktasında teğettir. Akım çizgisinin eğimi **Şekil 3.4**'te görüldüğü üzere v/u 'dur. Bu nedenle

$$\frac{dy}{dx} = \frac{v}{u} \tag{3.4}$$

eşitliği yazılabilir. Bu ifade bir akım çizgisi için iki boyutta bir diferansiyel denklemdir.

$$vdx - udy = 0 \quad (3.5)$$

(3.4)'ten elde edilen (3.5) denkleminin (3.3) ile aynı olduğu görülmektedir. Bu nedenle (3.3) denklemi matematiksel olarak hız vektörünün akım çizgisine teğet olduğunu ifade etmektedir.

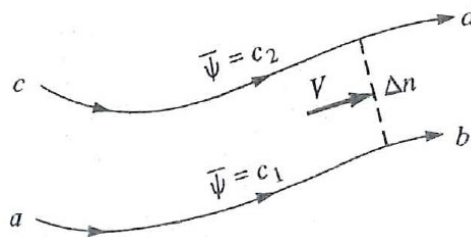
Eğer u ve v , x ve y 'nin fonksiyonu olarak bilinirse (3.4) denklemi akım çizgisi için (3.6)'da gösterilen şekilde cebirsel bir denklem elde edilecek şekilde integre edilebilir.

$$f(x, y) = c \quad (3.6)$$

Burada c integral sabitini göstermektedir. c 'nin değişik değerleri farklı akım çizgilerini gösterecektir. Eşitlik (3.6)'daki x, y fonksiyonu ψ ile gösterilip denklem tekrar yazıldığında elde edilen fonksiyona akım fonksiyonu denir:

$$\psi(x, y) = c \quad (3.7)$$

Şekil 3.5'te iki farklı akım çizgisi gösterilmiştir. ab ve cd akım çizgileri sırasıyla $\psi=c_1$ ve $\psi=c_2$ değerleri için gösterilmiştir.



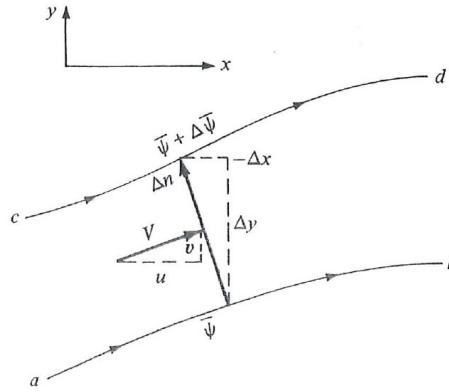
Şekil 3.5 : Farklı akım çizgileri.

(3.7)'deki c sabitinin değerinin bulunması için Şekil 3.5'teki ψ 'nin sayısal değeri $\Delta\psi$, $\psi=c_2$ cd akım çizgisi ve $\psi=c_1$ ab akım çizgisi arasındaki debiye eşit olsun. Şekil 3.5 iki boyutlu bir akış olduğu için, iki akım çizgisi arasındaki debi sayfaya dik birim kalınlık başına tariflenir. Bu da bu iki akım çizgisi için Δn kere birim kalınlıktır. Bu yüzden ab ve cd akım çizgileri arasındaki sayfaya dik birim kalınlık için debi eşitlik (3.8) sonucunu verir.

$$\Delta\psi = c_2 - c_1 \quad (3.8)$$

Bu sonuç halen c sabitlerinin değerlerinin bulunmasını sağlamamakta fakat biraz daha netleştirmektedir. ψ =sabit olması ve $\Delta\psi$ 'nin akım çizgileri arasında birim kalınlık başına debi değerine sahip olması doğaldır. Daimi bir akışta, bir akım tüpü içinde tüp boyunca akım sabittir. Tüpün herhangi bir kesitindeki debi de aynıdır. $\Delta\psi$ tanım olarak debiye eşit olduğu için $\Delta\psi$ 'nin kendisi de akım tüpü boyunca sabittir.

Akım fonksiyonunun en önemli özelliği, türevinin akım alanı hızlarını vermesidir. Bu ilişkinin elde edilmesi için **Şekil 3.6**'daki ab ve cd akım çizgileri dikkate alınacaktır gösterilmiştir.



Şekil 3.6 : Δn boyunca debi Δy ve $-\Delta x$ arasındaki debi toplamına eşittir.

Bu akım çizgilerinin birbirlerine çok yakın yani Δn 'nin çok küçük olduğu kabul edilmektedir, bu durumda akım hızı V Δn boyunca sabittir. Akım tüpünden sayfaya dik birim üniteden geçen debi

$$\Delta\psi \equiv \rho V \Delta n \quad (3.9)$$

şeklinde ifade edilebilir. Bu ifade oran şeklinde yazılarak

$$\frac{\Delta\psi}{\Delta n} = \rho V \quad (3.10)$$

elde edilir. Bu eşitlikte $\Delta n \rightarrow 0$ limiti alındığında da

$$\rho V = \lim_{\Delta n \rightarrow 0} \frac{\Delta\psi}{\Delta n} \equiv \frac{\partial\psi}{\partial n} \quad (3.11)$$

eşitliği ψ bilindiği duruma ρV çarpımının ψ 'nin V 'ye dik yönde kısmi diferansiyelinin alınarak bulunabileceğini ifade etmektedir. Bu ifadenin kartezyen koordinat sisteminde daha pratik bir formunu elde etmek için **Şekil 3.6**'daki gösterim dikkate alınacaktır. Burada yönelmiş Δn dik uzaklığı, önce y yönünde Δy kadar yukarı gidilmesi ve sonra x yönünde sola doğru $-\Delta x$ kadar gidilmesiyle aynıdır. Kütlelen korunumu nedeniyle, birim ünite kalınlığı için Δn boyunca debi, Δy ve $-\Delta x$ yönlerindeki debiye eşittir. Bu eşitlik

$$\Delta\psi = \rho \mathbf{V} \Delta n = \rho u \Delta y + \rho v (-\Delta x) \quad (3.12)$$

şeklinde ifade edilebilir. cd akım çizgisi ab çizgisine yaklaştığında (3.12) eşitliği limitte

$$d\psi = \rho u dy - \rho v dx \quad (3.13)$$

şeklinde ifade edilir. Bununla birlikte, $\psi = \psi(x, y)$ olduğu ve diferansiyelin zincir kuralı

$$d\psi = \frac{\partial \psi}{\partial x} dx + \frac{\partial \psi}{\partial y} dy \quad (3.14)$$

ifadesinde gösterilmektedir. (3.14) ifadesi (3.13)'te kullanılarak

$$\rho u = \frac{\partial \psi}{\partial y} \quad (3.15)$$

ve

$$\rho v = -\frac{\partial \psi}{\partial x} \quad (3.16)$$

eşitlikleri elde edilir. (3.15) ve (3.16) eşitlikleri önemlidir. Bir akış alanına ait $\psi(x, y)$ bağıntısı bilindiğinde, akış içinde herhangi bir noktada ρu ve ρv çarpanları ψ sırasıyla u ve v 'ye dik yönlerde alınacak diferansiyel ile bulunabilir. Ψ 'nin boyutu dik birim kalınlık başına debiye eşittir. SI birim sisteminde, ψ kg/(s.m) olarak bulunur.

Yukarıda tanımlanan ψ akım fonksiyonu sıkıştırılabilir ve sıkıştırılamaz akışlara uygulanabilir, (3.11) ifadesi

$$\mathbf{V} = \frac{\partial(\psi / \rho)}{\partial n} \quad (3.17)$$

şeklinde düzenlenerek sıkıştırılamaz akışlar da dikkate alındığında ρ sabit olduğundan (3.17) ifadesinde ρ yok olur ve

$$\mathbf{V} = \frac{\partial \psi}{\partial n} \quad (3.18)$$

bağıntısı elde edilir. Bu durumda (3.15) ve (3.16) ifadeleri

$$\begin{aligned} u &= \frac{\partial \psi}{\partial y} \\ v &= -\frac{\partial \psi}{\partial x} \end{aligned} \quad (3.19)$$

hızın bileşenlerini verecek şekile dönüşür.

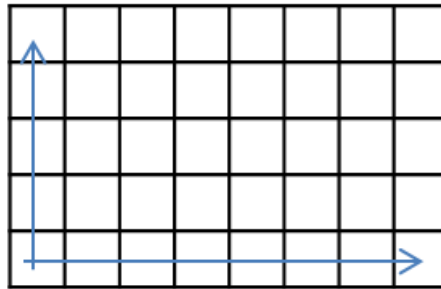
3.1.2 Akım çizgisi algoritması

Akım çizgilerinin oluşturulması algoritmasında eşitlik (3.19)'daki akım fonksiyonu bağıntısı kullanılacaktır. Akım fonksiyonundaki değişim

$$\begin{aligned} \Delta \psi &= u \Delta y \\ \Delta \psi &= -v \Delta x \end{aligned} \quad (3.20)$$

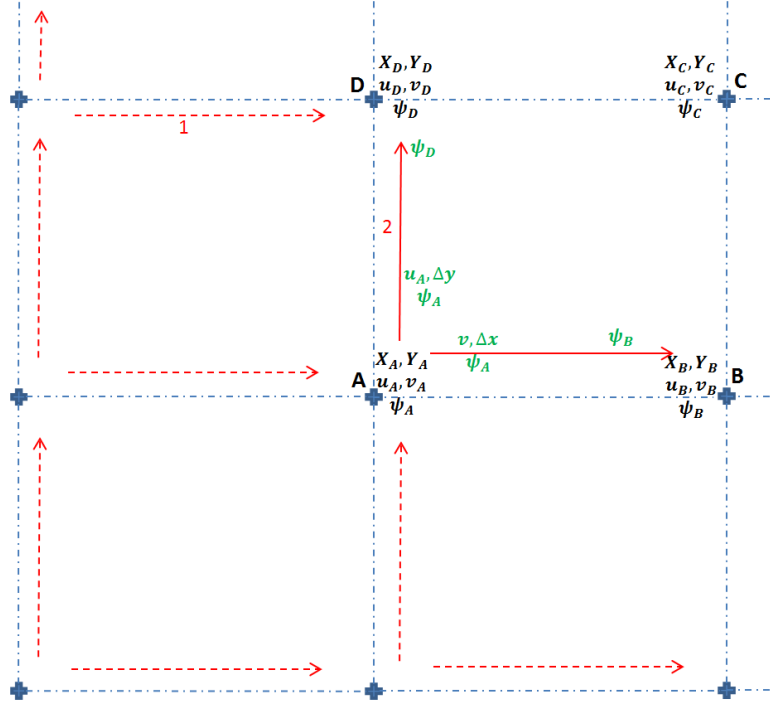
şeklinde ifade edilebilir.

Akım fonksiyonu değerlerini hesaplamak için Şekil 3.7'te görüldüğü gibi sol alt köşedeki noktadan başlanarak sütun sütun ilerlenerek tüm grid taranır.



Şekil 3.7 : Deney verilerinin oluşturduğu grid.

Seçilen her bir grid noktası (A) için bunun üstündeki (D) ve sağındaki (B) noktalarındaki ψ değerleri Şekil 3.8’te görüldüğü gibi A noktasındaki değer baz alınarak hesaplanır.



Şekil 3.8 : B ve D noktalarındaki değerlerin hesaplanması.

Hesaplama eşitlik (3.20)’den elde edilen

$$\begin{aligned}\psi_B &= \psi_A - v\Delta x \\ \psi_D &= \psi_A + u\Delta y\end{aligned}\quad (3.21)$$

bağıntıları kullanılır. Fakat (D) noktalarında (1) ile gösterildiği şekilde bir önceki sütun taramasında da değer atanmaktadır, bu değer ψ'_D olarak adlandırılmak üzere, mevcutsa bu değer de dikkate alınmalıdır, bu nedenle ψ_D için eşitlik (3.21)’deki atama

$$\psi_D = \begin{cases} \psi_A + u\Delta y \\ \frac{\psi'_D + (\psi_A + u\Delta y)}{2} \leftarrow \psi'_D \text{ mevcutsa} \end{cases} \quad (3.22)$$

ifadesi ile hesaplanır.

Ayrıca, akım içindeki cisime yakın hücrelerin bazı noktaları cisimin üzerine denk geleceği için bu hücrelerin sınırları cisim yüzeyiyle kesişme noktaları belirlenerek

yeniden saptanacak ve değer hesaplaması bu noktalarda yapılacaktır. Bu durum **Bölüm 3.3.4**'teki kontur tanımlamada anlatılacaktır.

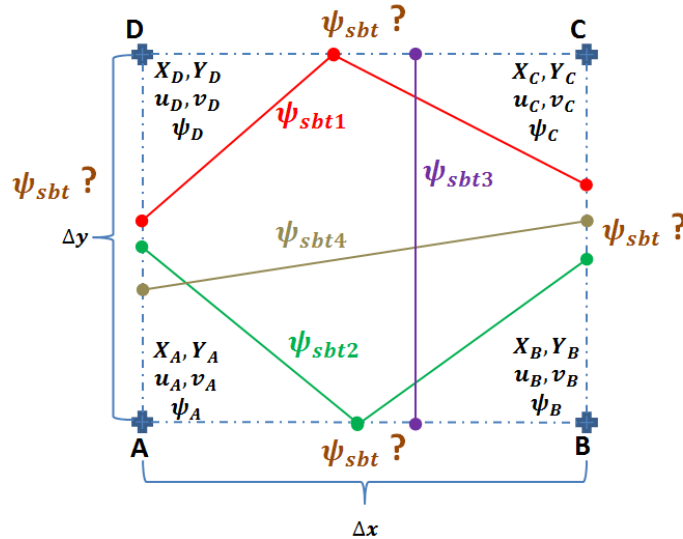
Sınır şartları olarak aşağıda belirtilen şartlar uygulanacaktır:

- Başlangıç noktasında ψ_A değeri sıfır kabul edilecektir.
- Cisim yüzeyin üzerinde $\psi = \text{sabit}$ olarak işlem yapılacaktır. Bunun için cisme sol alttan ilerleyerek ulaşılan ilk noktadaki değer tüm cisim yüzey noktaları için kullanılacaktır.
- Katı cisim noktalarında ψ hesabı yapılmayacak ve bu noktalardan akım çizgisi geçmeyecektir.

3.1.2.1 Belirli bir akım çizgisinin çizilmesi

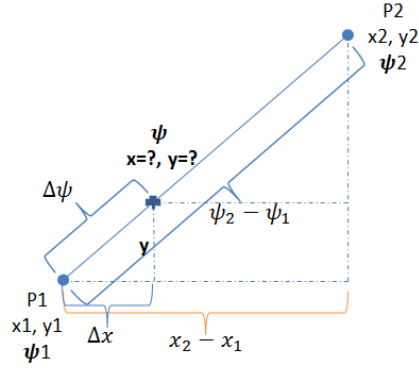
Belirlenen bir $\psi = \text{sabit}$ değerine karşılık gelen akım çizgisinin cisimle kesişimi olmayan tipik bir grid hücresinde oluşturulması aşağıda gösterilmiştir. Cisimle kesişimi olan hücrenin şekli yüzey kontur noktaları ile belirlenmektedir, bu durum yüzey konturunun belirlenmesi kısmında anlatılmaktadır.

Bu işleme başlamadan önce **Bölüm 3.1.2**'de anlatıldığı şekilde tüm noktalardaki akım fonksiyonu ψ değerleri hesaplanmış olmalıdır. **Şekil 3.9**'da gösterildiği şekilde, grid hücresinin kenarlarında ψ 'nin lineer değiştiği kabul edilmektedir.



Şekil 3.9 : Hücre içinde akım çizgisinin çizilmesi.

Belirlenen ψ_{sbt} değerinin konumu AB, BC, CD ve AD noktaları arasında **Şekil 3.10**'da gösterildiği şekilde doğrusal enterpolasyon yapılarak bulunur.



Şekil 3.10 : İki nokta arasında enterpolasyon.

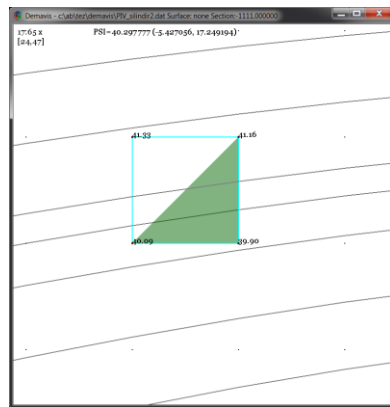
Herhangi iki nokta arasındaki enterpolasyon

$$\begin{aligned} x &= P1_x + \frac{P2_x - P1_x}{P2_\psi - P1_\psi} (\psi - P1_\psi) \\ y &= P1_y + \frac{P2_y - P1_y}{P2_\psi - P1_\psi} (\psi - P1_\psi) \end{aligned} \quad (3.23)$$

şeklinde hesaplanır. Bulunan noktalar arasında doğrular çizilerek hücre içindeki akım çizgisi çizilir. ψ_{sbt} 'in kenarlar üzerinde bulunduğu konumlara göre hücre içindeki akım çizgisi **Şekil 3.9'**da gösterilen akım çizgilerinin çeşitli şekillerde birleşmesinden oluşabilir. Hatta kapalı bir eğri çıkması da algoritma ve uygulama açısından mümkündür.

İşlem grid boyunca tüm hücrelerde tekrarlanarak tüm yüzeydeki ψ_{sbt} değerine karşılık gelen akım çizgileri çizilmiş olur.

Şekil 3.11'de bir hücrede farklı ψ_{sbt} değerleri için çizilmiş akım çizgileri görülmektedir. Örnek olarak da $\psi = 40,29777$ için çizilen akım çizgisi belirtilmiş ve ölçüm noktalarındaki hesaplanan ψ değerleri gösterilmiş durumdadır.

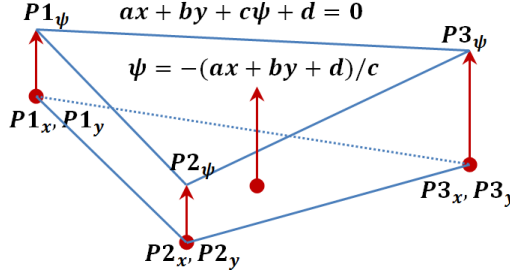


Şekil 3.11 : Bir hücredeki akım çizgisi.

3.1.2.2 Seçilen bir noktadaki akım çizgisinin çizilmesi

Seçilen noktadan geçen akım çizgisi aşağıda belirtilen şekilde oluşturulmaktadır.

Seçilen noktanın içinde bulunduğu grid hücresi, **Bölüm 3.3.7**'de belirtildiği şekilde bulunur. Grid hücresinde seçilen noktayı içine alan üçgen **Bölüm 3.3.8**'de belirtildiği şekilde tespit edilir. **Şekil 3.12**'de gösterildiği gibi bu üçgen eleman kullanılarak yüzey enterpolasyon değerleri belirlenir.



Şekil 3.12 : Üçgen eleman üzerinde yüzey enterpolasyonu.

Üçgen elemanı oluşturan noktalar ve üçüncü boyut olarak ψ akım fonksiyonu değerleri kullanılarak

$$ax + by + cz + d = 0$$

$$z = -\frac{ax + by + d}{c}$$

$$a = P1_y(P2_\psi - P3_\psi) + P2_y(P3_\psi - P1_\psi) + P3_y(P1_\psi - P2_\psi)$$

$$b = P1_\psi(P2_x - P3_x) + P2_\psi(P3_x - P1_x) + P3_\psi(P1_x - P2_x) \quad (3.24)$$

$$c = P1_x(P2_y - P3_y) + P2_x(P3_y - P1_y) + P3_x(P1_y - P2_y)$$

$$d = -(P1_x(P2_y \times P3_\psi - P3_y \times P2_\psi) + P2_x(P3_y \times P1_\psi - P1_y \times P3_\psi) + P3_x(P1_y \times P2_\psi - P2_y \times P1_\psi))$$

üç boyutlu yüzey denklemi elde edilir. Seçilen nokta P_x, P_y olmak üzere buradaki akım fonksiyonu değeri

$$\psi = -\frac{d + aP_x + bP_y}{c} \quad (3.25)$$

bağıntısı ile hesaplanır.

3.1.2.3 Tüm akım çizgilerinin çizilmesi

Tüm akım çizgilerinin çizilmesi için **Şekil 3.13**'te görüldüğü gibi yüzey üzerinde bir referans doğrusu alınmakta ve bu doğru istenen sayıda eşit adımlara bölünmektedir.

Tüm referans değerlerine karşılık gelen akım çizgileri **Bölüm 3.1.2.1**'de belirtilen yöntemle çizilerek istenen tüm akım çizgileri oluşturulmaktadır.

3.2.1 Girdaplılık fonksiyonu

The diagram illustrates a fluid element undergoing shear deformation. A parallelogram element is shown with vertices A, B, and C. The vertical height is dy . The horizontal displacement of the top edge is $(\frac{\partial u}{\partial y}) dy \Delta t$. The angle of deformation is $\Delta\theta$. The shear strain is κ . The horizontal displacement of the bottom edge is $(\frac{\partial v}{\partial x}) dx \Delta t$.

AB ve AC kenarları sırasıyla $-\Delta\theta_1$ ve $\Delta\theta_2$ kadar dönmüştür (saatin tersi yönü pozitif kabul edilmiştir). C noktasının $\Delta\theta_2$ derecelik dönmesiyle oluşan açının tanjantı

28

şeklinde ifade edilir. $\Delta\theta_2$ küçük bir açı olduğundan $\tan\Delta\theta_2 \cong \Delta\theta_2$ kabul edilebilir. Bu durumda (3.26) eşitliği

$$\Delta\theta_2 = \frac{\partial v}{\partial x} \Delta t \quad (3.27)$$

şekline dönüşür.

Benzer işlemler AV kenarının $-\Delta\theta_1$ dönmesi için tekrarlandığında

$$\Delta\theta_1 = -\frac{\partial u}{\partial y} \Delta t \quad (3.28)$$

eşitliği elde edilir. AB ve AC doğrularının açısal hızları $d\theta_1/dt$ ve $d\theta_2/dt$ olarak tanımlandığında, (3.28) ifadesinden

$$\frac{d\theta_1}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\Delta\theta_1}{\Delta t} = -\frac{\partial u}{\partial y} \quad (3.29)$$

ebağıntısı elde edilir, (3.27) ifadesinden de benzer şekilde

$$\frac{d\theta_2}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\Delta\theta_2}{\Delta t} = \frac{\partial v}{\partial x} \quad (3.30)$$

bağıntısı elde edilir. Tanım olarak, xy düzleminde görüldüğü üzere, akışkan elemanın açısal hızı AB ve AC açısal hızlarının ortalamasıdır. Bu büyüklük ω_z olarak isimlendirildiğinde bu tanım

$$\omega_z = \frac{1}{2} \left(\frac{d\theta_1}{dt} + \frac{d\theta_2}{dt} \right) \quad (3.31)$$

şeklinde ifade edilir. Eşitlik (3.27) ve (3.30) bu ifadede yerine konduğunda

$$\omega_z = \frac{1}{2} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \quad (3.32)$$

elde edilir. Yukarıdaki çıkarımda hareketin sadece xy düzleminde olduğu kabul edilmiştir, fakat akışkan elemanı genellikle üç boyutta hareket etmektedir. (3.32) ifadesindeki z indisi açısal hızın sadece z yönündeki büyüklüğünün bulunduğu

ifade etmektedir. ω 'nın x ve ye bileşenleri de benzer şekilde elde edilebilir. Akım elemanın üç boyutlu uzayda elde edilen açısal hızı

$$\omega = \frac{1}{2} [\omega_x \mathbf{i} + \omega_y \mathbf{j} + \omega_z \mathbf{k}] \quad (3.33)$$

olmak üzere

$$\omega = \frac{1}{2} \left[\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right) \mathbf{i} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right) \mathbf{j} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \mathbf{k} \right] \quad (3.34)$$

şeklinde ifade edilmiş olur. 2ω değeri teorik aerodinamik hesabında sıkça kullanılmaktadır. Bu nedenle açısal hızın iki katı, girdaplılık vektörü olarak tanımlanmıştır ve ξ simgesi ile gösterilmektedir:

$$\xi \equiv 2\omega \quad (3.35)$$

Bu durumda (3.34) ifadesi

$$\xi = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right) \mathbf{i} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right) \mathbf{j} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \mathbf{k} \quad (3.36)$$

şekline dönüşür. Buradan herbir eksenindeki girdaplılık değeri

$$\begin{aligned} \xi_x &= \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \\ \xi_y &= \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \xi_z &= \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \end{aligned} \quad (3.37)$$

olarak elde edilir. Bu ifadeler girdaplılık çizgilerinin çizim algoritmasında kullanılacaktır.

Gradyen vektörü tanımı ∇ :

$$\nabla A = \frac{\partial A}{\partial x} \mathbf{i} + \frac{\partial A}{\partial y} \mathbf{j} + \frac{\partial A}{\partial z} \mathbf{k} \quad (3.38)$$

(3.36) ifadesinde vektörel çarpım olarak dikkate alındığında

$$\nabla \times \mathbf{V} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u & v & w \end{vmatrix} \quad (3.39)$$

(3.36)'daki girdaplılık fonksiyonu

$$\xi = \nabla \times \mathbf{V} \quad (3.40)$$

şeklinde elde edilmiş olur. Buradan da bir akış alanında hızın gradyentinin girdaplılığı verdiği görülmektedir. Bundan aşağıdaki iki önemli tanıma ulaşılmaktadır:

1. Eğer akışın her noktasında $\nabla \times \mathbf{V} \neq 0$ şartı sağlanıyorsa akış rotasyoneldir; akışkan elemanlarının açısal hızları vardır.
2. Eğer akışın her noktasında $\nabla \times \mathbf{V} = 0$ şartı sağlanıyorsa akış irrotasyoneldir; akışkan elemanlarının açısal hızları yoktur, sadece taşınma hareketi yapıyorlardır.

İrrotasyonellik koşulu

$$\xi = \nabla \times \mathbf{V} = 0 \quad (3.41)$$

Ve ϕ skaler bir fonksiyon olmak üzere, aşağıdaki eşitlik

$$\nabla \times (\nabla \phi) = 0 \quad (3.42)$$

vektör özdeşliği kullanılarak eşitlik (3.41)'e taşındığında

$$\mathbf{V} = \nabla \phi \quad (3.43)$$

eşitliği elde edilir. Buradan irrotasyonel bir akışta gradyenti hız vektörünü veren bir ϕ fonksiyonu bulunduğu görülür; bu fonksiyon hız potansiyeli olarak adlandırılmaktadır.

Bir potansiyel akım alanında, potansiyel fonksiyonun $\phi(x, y) = \phi_i$ şeklinde sabit değerler aldığı noktaları birleştiren eğrilere “potansiyel eğrisi (çizgisi)” adı verilir. Bu akım alanında hız vektörü potansiyel fonksiyonunun gradyentine eşit olduğu için

potansiyel fonksiyonun herhangi bir doğrultudaki türevi bu doğrultudaki hız bileşenine eşit olur (Yükselen, t.y.).

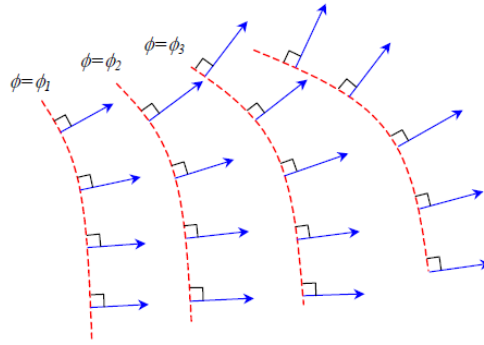
Kartezyen koordinat sisteminde hız bileşenleri u , v , w olmak üzere (3.43) eşitliği

$$u\mathbf{i} + v\mathbf{j} + w\mathbf{k} = \frac{\partial\phi}{\partial x}\mathbf{i} + \frac{\partial\phi}{\partial y}\mathbf{j} + \frac{\partial\phi}{\partial z}\mathbf{k} \quad (3.44)$$

şeklinde yazılır. Buradan hız bileşenleri

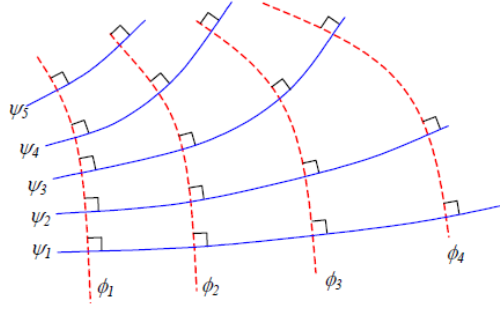
$$\begin{aligned} u &= \frac{\partial\phi}{\partial x} \\ v &= \frac{\partial\phi}{\partial y} \\ w &= \frac{\partial\phi}{\partial z} \end{aligned} \quad (3.45)$$

olarak elde edilir. Bu durumda bir potansiyel eğrisi boyunca $\phi = \text{sabit}$ olduğu için potansiyel fonksiyonunun bu eğrinin herhangi bir noktasında teğeti doğrultusundaki hız bileşeni sıfır olacak, bunun sonucu olarak da bu noktadaki hız bileşeni **Şekil 3.15**'te görüldüğü şekilde potansiyel eğrisine dik olacaktır.



Şekil 3.15 : Hız vektörleri ve bunlara dik potansiyel eğrileri.

Hız vektörleri akım çizgilerine teğet olduğu için akım çizgileri de **Şekil 3.16**'da gösterildiği şekilde potansiyel çizgilerine diktir:



Şekil 3.16 : Akım çizgileri ve bunlara dik potansiyel eğrileri.

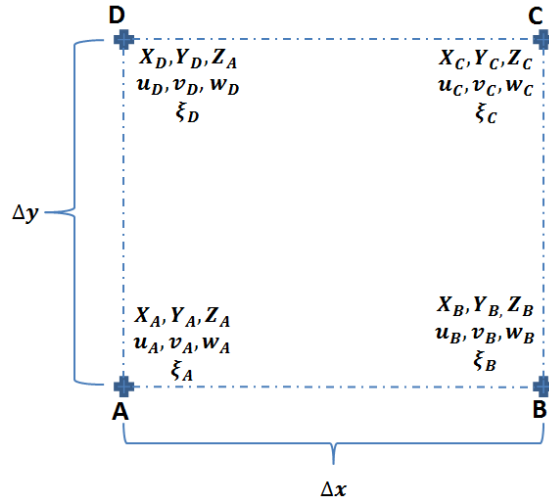
3.2.2 Girdaplılık çizgisi algoritması

Girdaplılık değerleri (3.37)'deki ifadeler kullanılarak

$$\begin{aligned}\xi_x &= \frac{\Delta w}{\Delta y} - \frac{\Delta v}{\Delta z} \\ \xi_y &= \frac{\Delta u}{\Delta z} - \frac{\Delta w}{\Delta x} \\ \xi_z &= \frac{\Delta v}{\Delta x} - \frac{\Delta u}{\Delta y}\end{aligned}\tag{3.46}$$

bağıntıları ile hesaplanabilir.

Şekil 3.17'de bir grid hücresindeki büyüklükler belirtilmiştir.



Şekil 3.17 : Grid hücresinde girdaplılık hesabı için kullanılacak büyüklükler.

Buna göre A noktasındaki girdaplılık değeri eksenlere göre

$$\begin{aligned}
\xi_x &= \frac{D_w - A_w}{D_y - A_y} - \frac{B_v - A_v}{B_z - A_z} \\
\xi_y &= \frac{B_u - A_u}{B_z - A_z} - \frac{D_w - A_w}{D_x - A_x} \\
\xi_z &= \frac{B_v - A_v}{B_x - A_x} - \frac{D_u - A_u}{D_y - A_y}
\end{aligned} \tag{3.47}$$

bağıntıları ile hesaplanır. u, v, w indisleri sırasıyla x, y, z yönündeki hız bileşenlerini göstermektedir.

Program uygulama aşamasında hesaplama ve çizimleri seçilen bir ölçüm düzlemi üzerinde yapmaktadır. Bu düzlem de XY düzlemi olacak şekilde kodlama yapılmıştır. Bu nedenle seçilen düzlemin, uygun şekilde bir eksen dönüşümü ile XY düzlemi haline getirilmesi gerekmektedir. Eşitlik (3.47)'deki eksenel girdaplılık ifadelerinin eksen takımı sol ile gösterilip hesaplama da saatin tersi yönündeki eksenden başlamak suretiyle yapıldığında aynı oldukları görülmektedir. Bu durum dikkate alınarak eksen dönüşümleri yapılabilir. XZ düzleminin XY düzlemine dönüşümü

$$\begin{aligned}
x' &= z \rightarrow u' = w \\
y' &= x \rightarrow v' = u \\
z' &= y \rightarrow w' = z
\end{aligned} \tag{3.48}$$

şeklinde gerçekleştirilir. Benzer şekilde YZ düzleminin XZ düzlemine dönüşümü

$$\begin{aligned}
x' &= y \rightarrow u' = v \\
y' &= z \rightarrow v' = w \\
z' &= x \rightarrow w' = u
\end{aligned} \tag{3.49}$$

şeklinde gerçekleştirilir. Hücre içindeki noktaların dönüşümleri bu şekilde yapıldıktan sonra (3.47)'deki ζ_z ifadesi ile XY düzleminin Z eksenine etrafındaki girdaplılık miktarı hesaplanır.

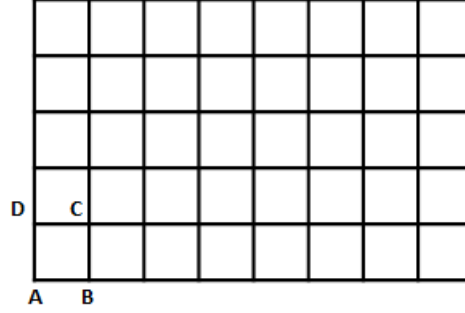
Bu işlem tüm grid hücreleri için tekrarlanarak seçilen düzlemdeki tüm noktalarda girdaplılık değerleri bulunur.

Bundan sonra **Bölüm 3.1.2**'de anlatılan akım çizgisi çizim algoritmaları kullanılarak girdaplılık çizgileri de çizilir.

3.3 Program Algoritmaları

3.3.1 Grid veri yapısı

Ölçüm noktaları, kullanılacak algoritmalara uygunluk sağlaması için bir grid matris şekline dönüştürülmektedir. **Şekil 3.18**'de ölçüm noktalarını düğüm noktaları olarak gösteren tipik bir grid yapısı ve ABCD noktalarının çevrelediği bir grid hücresi görülmektedir.



Şekil 3.18 : Ölçüm noktalarını gösteren tipik bir grid.

Programda bir grid noktası için tanımlanan veri yapısı **Çizelge 3.1**'de gösterilmiştir; bu veri yapısı *file_types.h* dosyasında *GridUnitDataTypeStruct* olarak tanımlanmıştır.

Çizelge 3.1 : Grid noktası veri yapısı.

Tanım	Değişken İsimleri
Koordinatlar	x ve y
Hız bileşenleri	u ve v
Durum bilgisi	status
Katı cisime ait bir nokta olması durumunda ait olduğu cisim numarası	object_number
Hesaplanacak büyüklükler (akım fonksiyonu, vortisite, vb.)	value[VALUE_COUNT]
X yönündeki kontur noktasının y koordinatı	Xconst_y
Bu noktadaki büyüklükler	Xconst_value[VALUE_COUNT]
Y yönündeki kontur noktasının x koordinatı	Yconst_x
Bu noktadaki büyüklükler	Yconst_value[VALUE_COUNT]

Her bir grid noktasında hesaplanabilecek büyüklükler de **Çizelge 3.2**'de gösterilmiştir, bunlar da demavis.h dosyasında *PointsValues* olarak tanımlanmıştır.

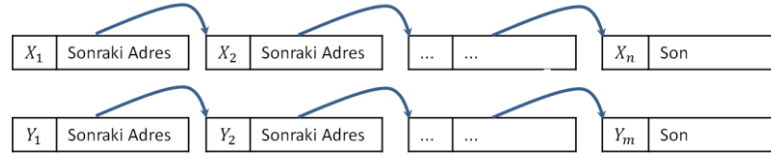
Çizelge 3.2 : Bir grid noktasında hesaplanabilecek büyüklükler.

Tanım	Değişken İsimleri
Akım fonksiyonu	PSI
Bileşke hız	VEL
Statik basınç	PRES_STATIC
Toplam basınç	PRES_TOTAL
Girdaplılık	VORTEX
Hızın U bileşeni	VEL_U
Hızın V bileşeni	VEL_V
Hızın W bileşeni	VEL_W

3.3.2 Ölçüm noktalarının grid matrisine yerleştirilmesi

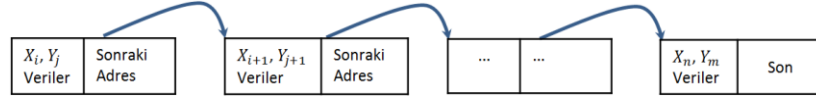
Öncelikle dosyadan okunan her veri satırı dinamik olarak oluşturulan sıralı bir zincir yapısına alınmaktadır. Böylelikle burada sabit bir sınır bulunmamakta, kullanılan bilgisayarın hafızasıyla orantılı bir büyüklük elde edilmektedir. Her satır okunurken X ve Y koordinatları kendilerine ayrılmış ayrı birer yine dinamik zincir yapıya eklenmektedir. Fakat bu işlem yapılırken yeni gelen her bir koordinat zincire eklenmeden önce baştan taranarak daha önce eklenmemiş olduğu kontrol edilmektedir; daha önce eklenen bir koordinat zincire tekrar eklenmemektedir.

Veri dosyasının okunması tamamlandığında **Şekil 3.19**'da görüldüğü gibi bir zincir yapıda tekil X koordinatları, bir zincir yapıda tekil Y koordinatları ve bir zincir yapıda veriler bulunmaktadır. Okunan veri sayısı tekil X ve Y sayılarının çarpımından farklı ise veri sayısındaki farklılık bildirilerek program veri sayısı hatası ile sonlanmaktadır.



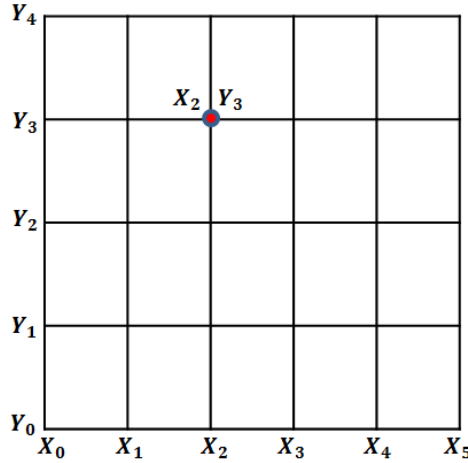
Şekil 3.19 : Tekil X ve Y zincir yapıları.

Bundan sonraki aşama **Şekil 3.20**'de görülen zincir yapısındaki verilerin $X \times Y$ boyutundaki $matrix[i][j]$ matrisine yerleştirilmesidir. Bu işlem şu şekilde yapılmaktadır: Zincir yapıdaki veriden ilk veri alınmakta ve X koordinatının tekil x zincirinde kaçınıcı sırada bulunduğu belirlenmektedir.



Şekil 3.20 : Zincir veri yapısı.

Bulunan değer i , bu verinin yerleştirileceği matrisin satır numarasını göstermektedir. Benzer şekilde Y koordinatının tekil y zincirinde kaçınıcı sırada bulunduğu tespit edilmektedir. bulunan j değeri de matrisin sütun numarasını göstermektedir. i ve j değerleri kullanılarak **Şekil 3.21**'de görüldüğü gibi veri $matrix[i][j]$ hücrelerine atanmaktadır.



Şekil 3.21 : Ölçüm noktalarının grid matrisine yerleştirilmesi.

İşlem aynı şekilde devam ettirilerek tüm veriler matriste yerlerine konmaktadır. Bu yöntemin avantajları aşağıda listelenmiştir:

- X ve Y adımları kendi içlerinde de sabit olmak zorunda değildir. Bu özellik ölçüm noktaları aralıklarındaki farklılıkların sorun oluşturmamasını sağladığı için pratik de oldukça fayda sağlamaktadır.
- Dosyadaki veri satırlarının belirli bir sırada olmasına gerek yoktur.

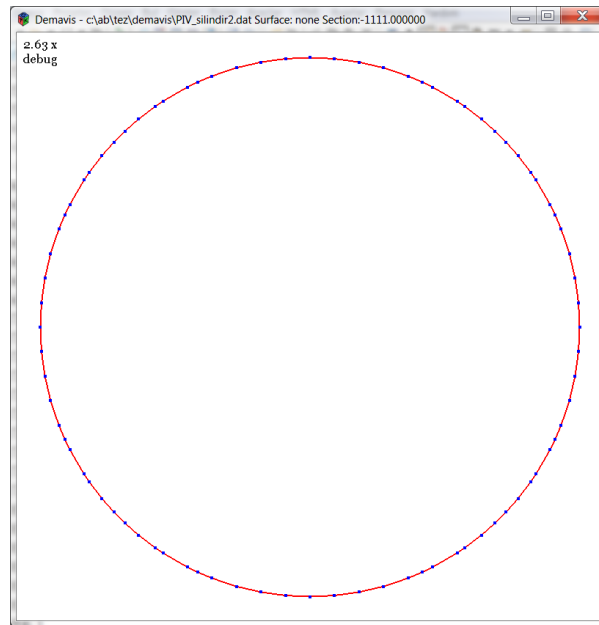
- Veri boyutunda özel bir sınırlama yoktur, pratik sınır kullanılan bilgisayarın hafızası ile orantılıdır. Günümüz bilgisayarlarındaki gigabyte'lar seviyesindeki hafıza pratik bir sınırın çok üzerindedir.

3.3.3 Katı cisim tanımlama

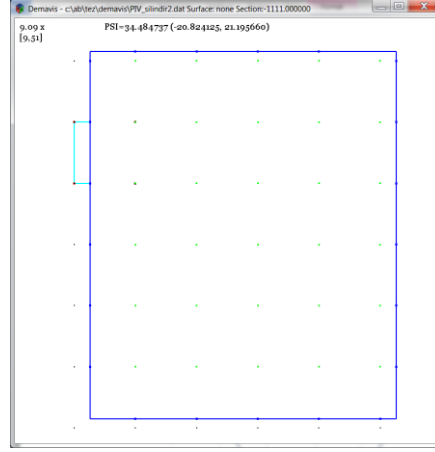
Demavis programında katı cisimler daire, dikdörtgen, üçgen, denklem ve koordinat dosyası ile tanımlanabilecektir. Daire, dikdörtgen ve üçgenin doğrudan ekran üzerinde düzenlenebilmesi de mümkündür, işlemin nasıl yapılacağı program kullanım bölümünde anlatılmaktadır.

3.3.4 Katı cisim yüzey konturunun belirlenmesi

Katı cisim yüzey konturu tanımlaması: Katı cisim yüzey doğrusu/eğrisi ile bu yüzey eğrisinin iki tarafında kalan her bir grid noktası çiftinin oluşturduğu doğrunun kesişim noktaları katı cisim yüzey konturu olarak adlandırılmıştır. Bu noktalar komşu grid noktalarıyla x veya y yönünde aynı hizadadırlar. Programda aslında gerçek katı cisim çevresi olarak bu kontur noktalarının birleştirilmesiyle oluşan poligon kullanılmaktadır. Poligonun çözünürlüğünü grid hücresinin büyüklüğü, yani ölçüm noktaları adımlarının sıklığı belirlemektedir. Veri noktaları arasındaki uzaklık ne kadar küçük olursa, poligon tanımlanan yüzey eğrisine o kadar yakın olacaktır. **Şekil 3.22**'de daire konturu görülmektedir.

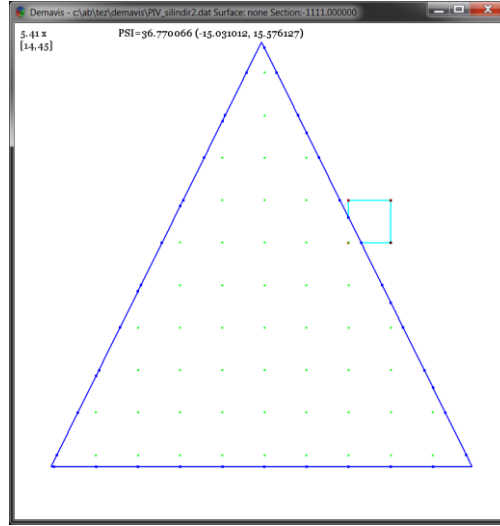


Şekil 3.22 : Daire konturu.



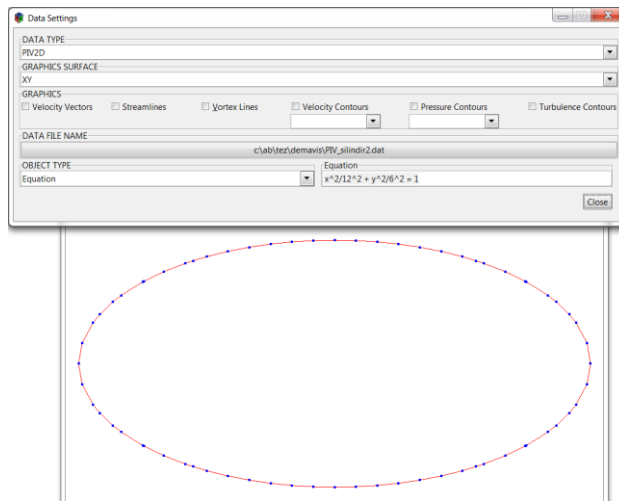
Şekil 3.24 : Dikdörtgen konturu ve küçülmüş bir hücre elemanı.

Şekil 3.25’te üçgen konturunun belirlenmesine ilişkin bir örnek görülmektedir.



Şekil 3.25 : Üçgen konturu ve küçülmüş bir hücre elemanı.

Şekil 3.26’da denklem ile tanımlanmış bir elips konturu görülmektedir.



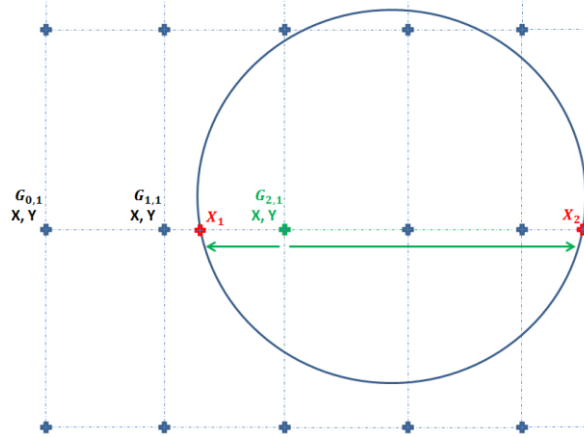
Şekil 3.26 : Elips denklem konturu.

3.3.5 Cisim Konturu içinde Kalan Noktaların Belirlenmesi

Cisim konturu belirlendikten sonra bu kontur içinde kalan grid noktalarının da belirlenmesi ve bunların katı cisim olarak işaretlenmesi gerekmektedir.

Daire, elips vb. denklem ile tanımlanan cisimlerin kontur noktalarının belirlenmesi aşağıdaki yöntemle yapılmaktadır.

Grid $G_{i,j}$ noktasının y koordinatı yüzey eğrisi denklemine konulmakta ve buradan iki adet x değeri X_1 ve X_2 elde edilmektedir. $G_{i,j}$ noktasının x koordinatı elde edilen x değerleri arasında $X_1 < G_{i,j}X$ ve $G_{i,j}X < X_2$ şartı kontrol edilmektedir. Bu şart sağlanıyorsa $G_{i,j}$ noktası için bir kontur içindedir $G_{i,j}$ Status değerine katı cisim kodu atanır. Bu şartın Şekil 3.27’de verilen örnekte $G_{0,1}$ noktası için sağlanmadığı $G_{2,1}$ noktası için sağlandığı görülmektedir.



Şekil 3.27 : Cisim konturu içindeki noktaların belirlenmesi.

Dikdörtgen ve üçgen için de benzer yöntem bu cisimlerin geometrilerine uygun şekilde uygulanmaktadır.

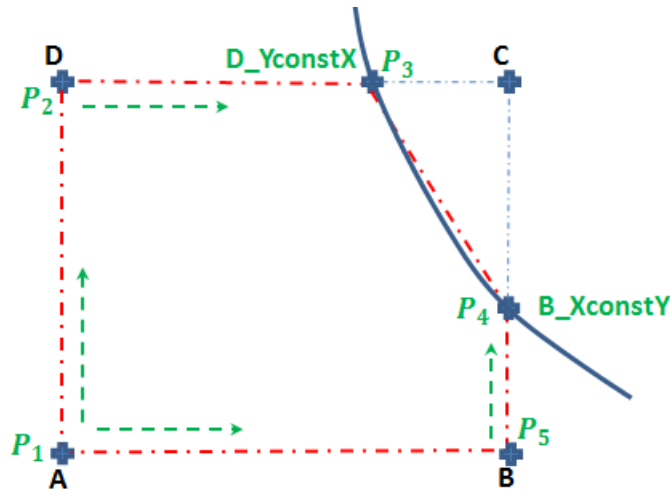
3.3.6 Grid hücresinin köşelerinin belirlenmesi

Cisim yüzeyine en yakın hücrelerin şekilleri, hücrelerin cisim içindeki noktaları yerine kontur noktalarının kullanılmasıyla yeniden şekillenmektedir. Bir hücrenin en genel yapısı Şekil 3.28’te görüldüğü gibi düzgün olmayan bir beşgen formundadır.

Programın akışında aşağıdaki işlemler sırasıyla yapılmaktadır:

- Katı cisim tanımlama
- Cisim kontur noktalarının bulunması

- Kontur içindeki grid noktalarının belirlenmesi



Şekil 3.28 : Genel hücre formu.

Bunlardan sonra büyüklüklere ait çizimler yapılırken grid hücresinin köşeleri belirlenmesi gerekmektedir. Şekil 3.28’te görüldüğü gibi Hücrenin yeni noktaları kontur belirlenmesi sırasında grid veri yapısında tanımlanan XconstY ve YconstX alanlarında tutulmaktadır. Hücrenin referans olan sol alt noktası belirlendikten sonra her bir noktasındaki bu değerler de dikkate alınarak hücrenin köşe noktaları belirlenir.

A, B ve D noktalarına ait kontur noktalarının belirlenmiş olup olmadığı kontrol edilerek bu noktalar grid hücresinin köşe noktalarına ilave edilmektedir.

Bu arada A, B, C ve D noktalarının kontur içinde olup olmadıkları da kontrol edilmekte ve kontur içinde katı nokta olarak işaretlenmişlerse grid hücresinin katı noktası olarak ilave edilmemektedirler.

A noktası referans alınıp saat yönünde noktalar numaralandığında hücre sıralı olarak P_1, P_2, P_3, P_4, P_5 noktalarından oluşmaktadır.

3.3.7 Seçilen bir noktanın bulunduğu hücrenin belirlenmesi

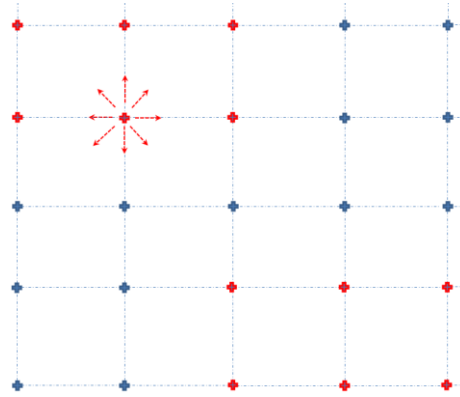
Etkişimli modda ekran üzerinden seçilen bir nokta veya bir referans noktası olarak belirlenmiş bir noktadaki büyüklüklerin hesaplanması için öncelikle bu noktanın hangi hücre içinde olduğu hücrenin sol alt köşesi referans olarak kullanılmak üzere belirlenmelidir.

Aşağıdaki bağıntıda elde edilen her iki şart da sağlanıyorsa noktanın üçgen içinde olduğu belirlenir (McRae, t.y.).

$$\begin{aligned}
 f_{AB} &= (P_Y - T_{0Y})(P_{1X} - P_{0X}) - (P_X - T_{0X})(T_{1Y} - T_{0Y}) \\
 f_{CA} &= (P_Y - T_{2Y})(T_{0X} - T_{2X}) - (P_X - T_{2X})(T_{0Y} - T_{2Y}) \\
 f_{BC} &= (P_Y - T_{1Y})(T_{2X} - T_{1X}) - (P_X - T_{1X})(T_{2Y} - T_{1Y}) \\
 (f_{AB} \times f_{BC}) &\geq 0 \\
 (f_{BC} \times f_{CA}) &\geq 0
 \end{aligned} \tag{3.51}$$

3.3.9 Katı cisim sayma ve numaralandırma

Veri dosyasından grid noktasının ölçüm noktası mı, katı cisim noktası mı olduğuna dair bilgi grid veri yapısı içindeki *Status* değerine okuma sırasında atanmaktadır. Veri okuma işlemi tamamlandıktan sonra Tüm grid noktaları taranarak eğer varsa katı cisimler numaralandırılır. Bunun için ilk katı cisim noktasına gelindiğinde aşağıda gösterilen özyinelemeli bir tarama fonksiyonu çalıştırılır; bu fonksiyon tekrar kendisini **Şekil 3.31**'de görüldüğü gibi her yönde çağırır ve ilerlediği yönde katı cisim bulamazsa geri döner.



Şekil 3.31 : Her yöne tarama yöntemi ile numaralandırma.

Bu yöntemle her bir katı cisime ait grid noktası kendi içinde aynı numarayı alacak şekilde numaralandırılmış olur. Böylelikle akış içindeki katı cisim sayısı ve bunlara ait grid noktaları belirlenmiş olur.

```

static void calculation_obstacle_fill_set_number(int i, int j, int
object_number)
{
    GridInfoType* info = file_get_grid_info();
    GridUnitDataType** matrix = file_get_matrix();

    // Out of grid.
    if (i<0 || j<0 || i>(info->nx-1) || j>(info->ny-1)) {
        return;
    }
}

```

```

    // Already set or not an obstacle
    if (matrix[i][j].object_number || !(matrix[i][j].status & OBSTACLE_MASK) )
return;

    matrix[i][j].object_number = object_number;

    // Fill with the given value
    calculation_obstacle_fill_set_number(i+1, j, object_number);
    calculation_obstacle_fill_set_number(i+1, j+1, object_number);
    calculation_obstacle_fill_set_number(i, j+1, object_number);
    calculation_obstacle_fill_set_number(i-1, j+1, object_number);
    calculation_obstacle_fill_set_number(i-1, j, object_number);
    calculation_obstacle_fill_set_number(i-1, j-1, object_number);
    calculation_obstacle_fill_set_number(i, j-1, object_number);
    calculation_obstacle_fill_set_number(i+1, j-1, object_number);
}

```


4. PROGRAM

Demavis programı hız, performans, çeşitli platformlarda çalışabilirlik, geliştirmeye açıklık, kolay kurulum ve kullanım özellikleri gözetilerek geliştirilmiştir. Bu amaçlar için 64 bit'lik açık kaynak kodlu geliştirme araçları ve standart C dili tercih edilmiştir. Derleyici olarak 64 bit GNU C kullanılmıştır. Çizimler OpenGL ile gerçekleştirilmiştir, Kullanıcı arayüzü olarak GTK kütüphanesi kullanılmıştır. Elde edilen program 64 bit işlemcili Linux ve Windows makinalarda yönetici yetkisi gerektirmeden kopyalama yoluyla çalışmaktadır. Uygulama yaklaşık 9900 satır kod'dan oluşmakta ve binary büyüklüğü 300 KB civarındadır.

Program çalıştırıldığında seçilen yüzey için tüm büyüklüklere ait çizim noktalarını hesaplamakta ve hızlı işleyiş için hafızada tutmaktadır. Sonrasında, yaklaştırma, kaydırma, çizimleri birlikte ve ayrı gösterme vb çeşitli fonksiyonallikleri OpenGL ve diğer kütüphaneler vasıtası ile interaktif olarak hızlı bir şekilde gerçekleştirilmektedir.

Program açık kaynak kodlu olarak konuya ilgi duyan uluslararası araştırmacılar tarafından da geliştirilebilmesi için İngilizce olarak geliştirilmiştir.

4.1 Genel Çalışma Prensipleri ve Çalışma Adımları

Programın çalışma sırasında yaptığı işlemler aşağıdaki listede özetlenmiştir:

- GTK ve OpenGL kütüphanelerinin tanımlarının yapılması ve menünün hazırlanması.
- DataSettings penceresinin hazırlanması.
- OpenGL render ve diğer fonksiyonlarının GTK geriçağırım fonksiyonlarına bağlanması ve programın sürekli çizim moduna alınması.
- Kullanıcı tarafından veri tipinin seçilmesi ve ilgili parametrelerin belirlenmesi.
- Buna uygun şekilde veri dosyasının okunması.

- Varsa cisim geometrilerinin grid üzerinde çizilmesi.
- Hesaplamaların başlatılması.
- X ve Y tekil zincir yapılarının oluşturulması.
- Grid matrisinin oluşturulması.
- Ekranda gridin gösterilmesi.
- Cisim noktalarının belirlenmesi, numaralandırılarak sayılması ve gösterilmesi.
- Cisim konturunun belirlenmesi.
- Kontur içinde kalan grid noktalarının belirlenmesi.
- Büyüklüklerin grid ve kontur noktalarında hesaplanması.
- Büyüklüklerin min. ve max. değerlerinin bulunması.
- Elde edilen verilerin grid bilgisi olarak ekrana basılması.
- Akım çizgileri referans noktalarının belirlenmesi.
- Akım çizgilerini oluşturan noktaların hesaplanması.
- Diğer büyüklükler için referans değerlerin hesaplanması.
- Diğer büyüklüklerin çizimlerine ait noktaların hesaplanması.
- Kullanıcının seçimine göre çizimlerin ayrı ayrı veya birlikte yapılması.
- Analiz modlarında etkileşimli ve dinamik olarak çizimlerin yapılması ve tüm değerlerin detaylı olarak incelenmesi.
- İstenen noktadan geçen yeni çizgilerin dinamik olarak eklenmesi.
- Eş büyüklük (kontur) çizgilerinin referans aralıklarının tekrar tanımlanarak çizilebilmesi.
- Programdan çıkışta kullanılan tüm hafıza alanının serbest bırakılması.

Program başlayıp GTK ve OpenGL kütüphaneleri için gerekli tanımların yapılmasıyla birlikte sürekli çizim moduna geçmektedir. Kullanıcı seçimleriyle

belirtilen çizimler sürekli çalışacak şekilde ana program döngüsü başlar. Pencere ve kullanıcı arabirimleri (fare, klavye) etkileşimiyle istenilen işlemler gerçekleştirilir.

Programın hesaplama kısmında çizimler için kullanılacak nokta koordinatları zincir yapılar şeklinde dinamik olarak hafızaya alınmaktadır. Böylelikle OpenGL kütüphanesi ile çok hızlı çizimler yapabilmek mümkün olmaktadır.

Çizimlerin gösterimlerinde oyun programlarında kullanılan animasyon tekniği kullanılmaktadır. Tüm seçilen çizimler çok hızlı bir şekilde grafik ekranın ilk ekranında yapılmakta, ikinci ekran sayfasında bir sonraki çizim hazırlanmakta ve tamamlandığında ekran değişimi yapılarak kesintisiz bir görünüm elde edilmektedir. Bu döngü program boyunca sürekli çalışmaktadır. Çift ekran sayfası kullanım işlemi, büyütme-küçültme, kaydırma işlemleri OpenGL kullanılarak yapılmaktadır.

4.2 Genelleştirilmiş Parametrik Yapı ve Çizim Fonksiyonları

Hesaplama, çizim ve menü fonksiyonları seçilen büyüklükler (akım fonksiyonu, vortisite, basınç, hız, vb.) için parametrik çalışacak şekilde tasarlanmıştır. Böylelikle tüm işlemler aynı fonksiyonlar kullanılarak yapılabilmektedir. Yöntemde sağlanan bu bütünlük; kod tasarrufu, yeni büyüklüklerin kolayca eklenmesini sağlamak ve yöntemin tekrar tekrar kodlanmasıyla oluşabilecek kodlama hatalarını minimize etmektedir.

Programda yapılacak çizimler için aşağıdaki yapı kullanılmaktadır:

```
struct toggle_struct {
    int value[TOGGLE_COUNT];

    // Function Pointer Array
    void (*RenderFunction[TOGGLE_COUNT]) ();

    GtkWidget *RenderMenu_item[TOGGLE_COUNT];
    char text[TOGGLE_COUNT][100];

    // define colors
};
```

Bu yapıda tanımlanan değişkenlerin kullanım amaçları **Çizelge 4.1**'de verilmiştir.

Çizelge 4.1 : Çizim veri yapısı.

Tanım	Değişken İsimleri
value[TOGGLE_COUNT]	Çizimin yapılıp yapılmayacağını belirleyen değer.
*(RenderFunction[TOGGLE_COUNT])	Büyükliğin çizimini yapan fonksiyon.
RenderMenu_item[TOGGLE_COUNT]	Menü alanı.
Text[TOGGLE_COUNT]	Menüdeki ismi.
//define colors	Çizim rengi henüz tanımlanmadı.

Bu yapı kullanılarak 30 seçenekli çizim menüsü aşağıdaki kod ile oluşturulmaktadır.

```
// Render Menu
GtkWidget *RenderMenu_container=gtk_menu_new();
GtkWidget *RenderMenu_main_item=gtk_menu_item_new_with_mnemonic("_Render");

int i; static char id[TOGGLE_COUNT][10];
for(i=0; i<TOGGLE_COUNT; i++) {

    winfo.toggle.RenderMenu_item[i]=gtk_check_menu_item_new_with_label(winfo.togg
le.text[i]);
    itoa(i, id[i], 10);
    g_signal_connect( winfo.toggle.RenderMenu_item[i], "activate",
G_CALLBACK(gtkwin_toggle_checkmenu_function), (gpointer)id[i] );
    gtk_menu_shell_append( GTK_MENU_SHELL(RenderMenu_container),
winfo.toggle.RenderMenu_item[i] );
}

gtk_widget_show(RenderMenu_main_item);
gtk_widget_show_all(RenderMenu_container);
```

Çizim işlemleri de yine bu yapı kullanılarak ortak fonksiyonlar ile aşağıdaki kod ve benzeri ile yapılabilmektedir:

```
static void display_render_ANALYSE_PSI()
{
    // Assign current mouse position's real coordinates
    display_analyze_selected_point(dispatch.realPoint, PSI);
}

static void display_render_ANALYSE_VELOCITY()
{
    // Assign current mouse position's real coordinates
    display_analyze_selected_point(dispatch.realPoint, VEL);
}

static void display_render_ANALYSE_VELOCITY_U_CONTOURS()
{
}
```

```
// Assign current mouse position's real coordinates
display_analyze_selected_point(disp.realPoint, VEL_U);
}
```

4.3 Programın Doğrulaması

Programın doğrulaması teorik potansiyel akım fonksiyonu ile karşılaştırma ve yaygın kullanılan TechPlot programının çizimiyle karşılaştırılarak yapılmıştır.

4.3.1 Potansiyel akım örneği ile kıyaslama

Matematikte, dairesel bir silindir etrafındaki potansiyel akış, silindir etrafındaki sürtünmesiz, sıkıştırılamaz akımın klasik bir çözümüdür. Akım, silindirden uzakta tekyönlü ve uniform yapıdadır. Akışta girdaplılık yoktur buna bağlı olarak hız alanı irrotasyonel ve potansiyel akış olarak modellendirilebilir. Bu akışta, gerçek bir akışkandan farklı olarak yüzeyde sıfır net sürükleme vardır bu da d'Alembert paradoksu olarak bilinir (Dairesel silindir etrafında potansiyel akış, t.y.).

R yarıçaplı bir silindirin iki boyutlu, sıkıştırılamaz, sürtünmesiz bir akışın içine konduğu durum ele alınacaktır. Amaç hız vektörü $\mathbf{V}$ ve P basınç alanını bulmaktır. Silindirden yeterince uzakta hız vektörü

$$\mathbf{V} = U\mathbf{i} + 0\mathbf{j} \quad (4.1)$$

olarak ifade edilebilir. U sabit akımın hızını gösteren bir sabittir. Silindir yüzeyinde

$$\mathbf{V} \cdot \mathbf{n} = 0 \quad (4.2)$$

şartı geçerlidir, $\mathbf{n}$ silindir yüzeyine dik bir vektördür. Sıkıştırılamazlık ve irrotasyonel akım şartı

$$\nabla \times \mathbf{V} = 0 \quad (4.3)$$

şeklinde gösterilir. Akış irrotasyonel olduğu için hız potansiyeli mevcuttur, bu ilişki

$$\mathbf{V} = \nabla \phi \quad (4.4)$$

Şeklinde gösterilir. Sıkıştırılamazlık şartından dolayı

$$\nabla \cdot \mathbf{V} = 0 \quad (4.5)$$

yazılabilir. Bu nedenle $\emptyset$ potansiyel fonksiyonu Laplace denklemini sağlamak zorundadır:

$$\nabla^2 \phi = 0 \quad (4.6)$$

Bu denklemin kutupsal koordinatlarda çözümü daha kolay olacağı için kartezyen koordinatlardan kutupsal koordinatlara dönüşüm

$$\begin{aligned} x &= r \cos \theta \\ y &= r \sin \theta \end{aligned} \quad (4.7)$$

bağıntılarıyla sağlanır. Bu durumda (4.7) dönüşümü Laplace (4.6) denkleminde kullanıldığında

$$\frac{\partial^2 \phi}{r^2} + \frac{1}{r} \frac{\partial \phi}{\partial r} + \frac{1}{r^2} \frac{\partial^2 \phi}{\partial \theta^2} = 0 \quad (4.8)$$

elde edilir. Sınır şartları da kullanıldığında $\emptyset$ potansiyel fonksiyonu

$$\phi(r, \theta) = U \left(r + \frac{R^2}{r} \right) \cos \theta \quad (4.9)$$

şeklinde elde edilir (William, 1993). Hız bileşenleri kutupsal koordinatlarda (4.7) bağıntısı kullanılarak

$$V_r = \frac{\partial \phi}{\partial r} = U \left(1 - \frac{R^2}{r^2} \right) \cos \theta \quad (4.10)$$

ve

$$V_\theta = \frac{1}{r} \frac{\partial \phi}{\partial \theta} = -U \left(1 + \frac{R^2}{r^2} \right) \sin \theta \quad (4.11)$$

şeklinde elde edilir.

Akım fonksiyonu sıkıştırılmaz akımda

$$\mathbf{V} = \nabla \psi \times \mathbf{k} \quad (4.12)$$

şeklinde gösterilip vektör denkliği kullanılırsa

$$\mathbf{V} \cdot \nabla \psi = 0 \quad (4.13)$$

eşitliği elde edilir. ψ 'nin sabit bir değeri için elde edilen kontur hız vektörü $\mathbf{V}$ 'ye teğet bir akım çizgisi oluşturur. Silindirin üzerinden geçen akım için

$$\psi = U \left(r - \frac{R^2}{r} \right) \sin \theta \quad (4.14)$$

bağıntısı elde edilir.

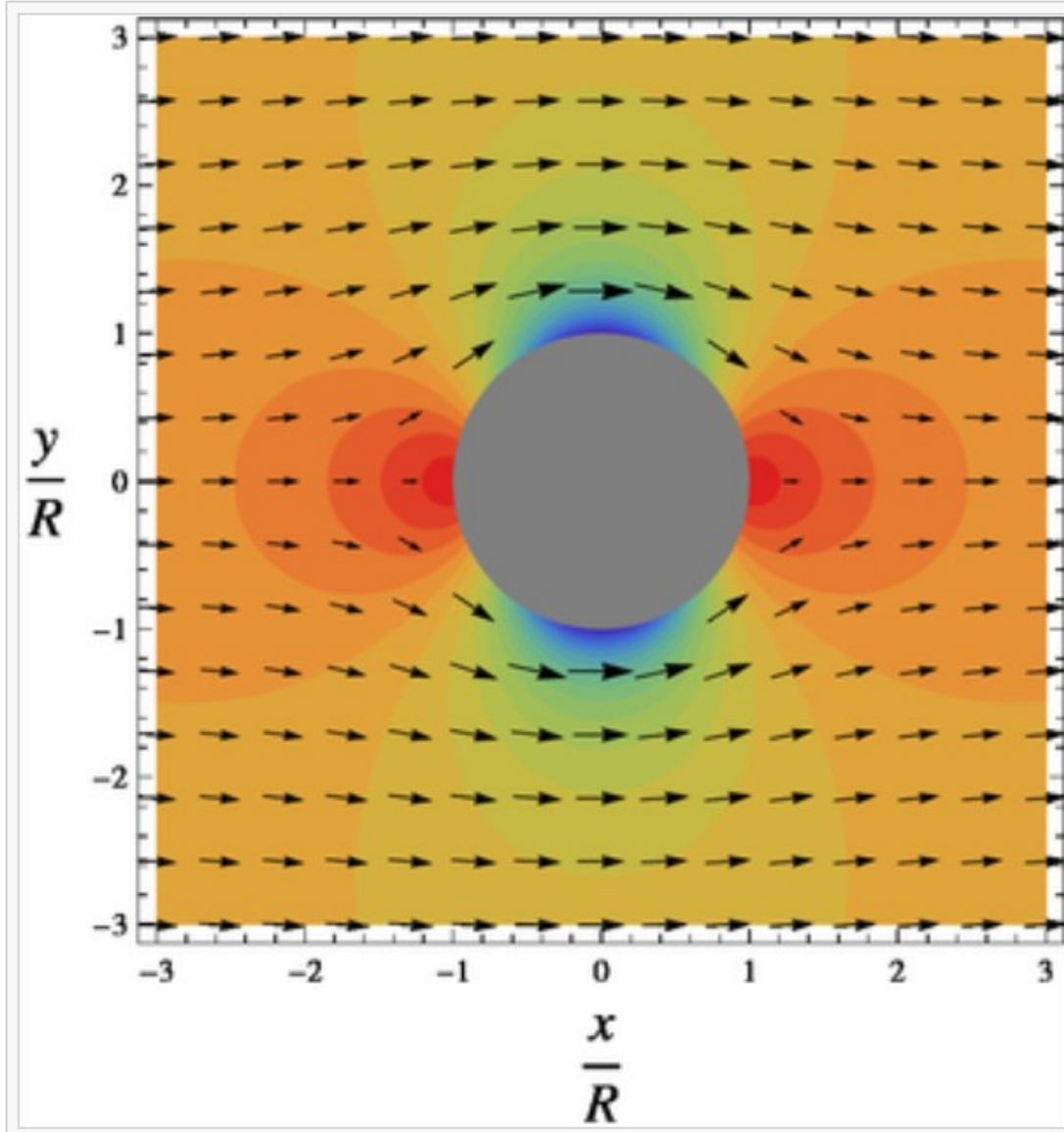
4.3.2 Potansiyel akım hız vektörleri ve akım çizgileri ile kıyaslama

Geliştirilen Demavis programında hız vektörleri ve akım çizgilerini karşılaştırmak için potansiyel akıma ait koordinat ve hız verileri oluşturularak akım çizgileri çizdirilmiştir. Uygulamada x-y ekseninin merkezinde 10 cm. yarıçapında bir silindir olduğu varsayılmıştır. $x=[-30, 30]$ ve $y=[-30, 30]$ düzlemi x ve y'nin 1'er mm.'lik adımlarıyla taranmış ve noktadaki hız bileşenleri sayısal olarak hesaplanarak PIV veri formatından üretilmiştir. **Çizelge 4.2**'de bu tablonun başlangıç kısmı gösterilmiştir.

Çizelge 4.2 : Potansiyel akım veri örneği.

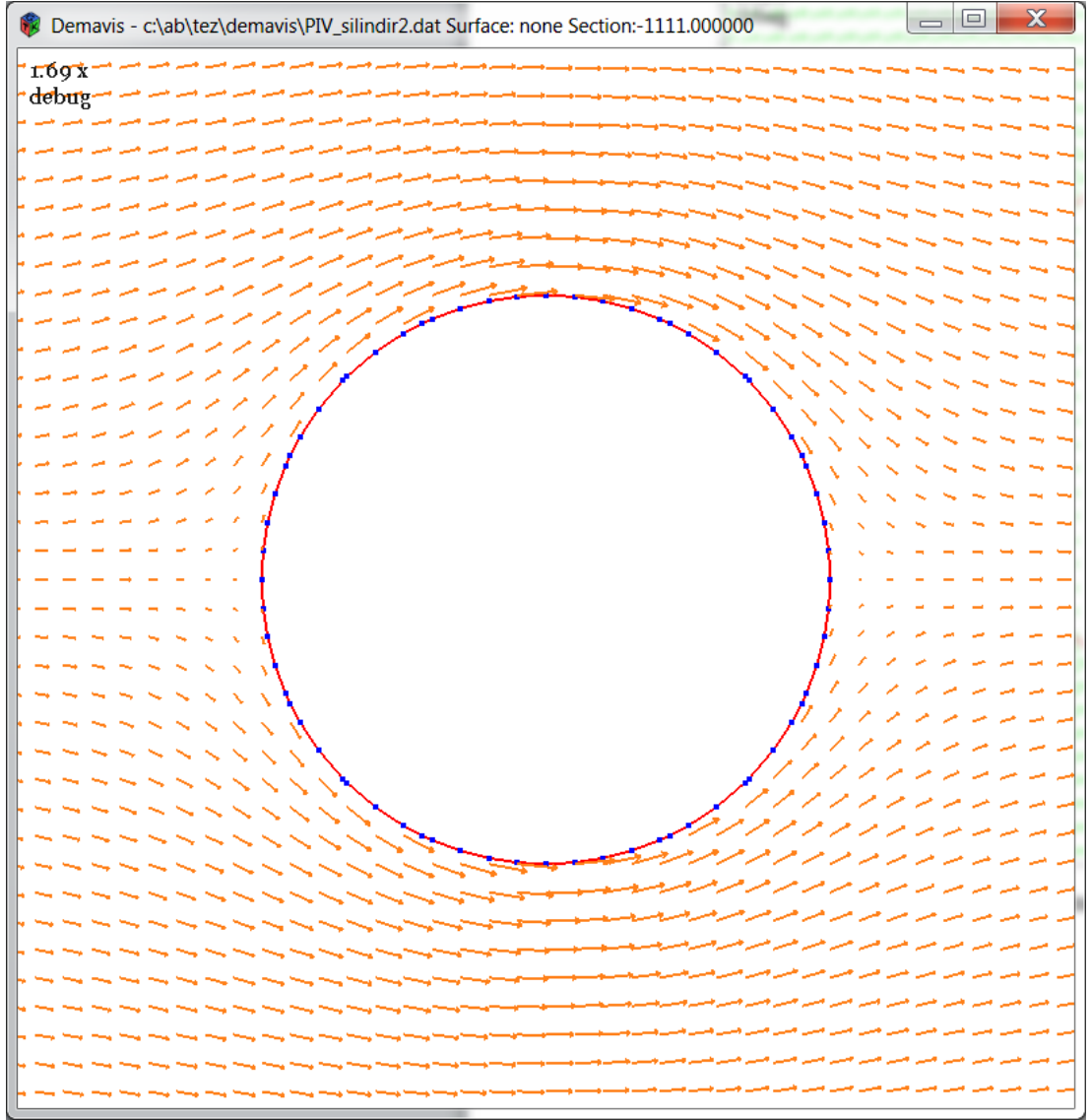
x	y	u	v	status
-30	-30	1	-0.055555556	16
-29	-30	1.0019465	-0.057405262	16
-28	-30	1.004090476	-0.059241372	16
-27	-30	1.006443976	-0.061048198	16
-26	-30	1.009018527	-0.062807596	16

Şekil 4.1'de silindir etrafındaki potansiyel akışa ait hız vektörleri görülmektedir.



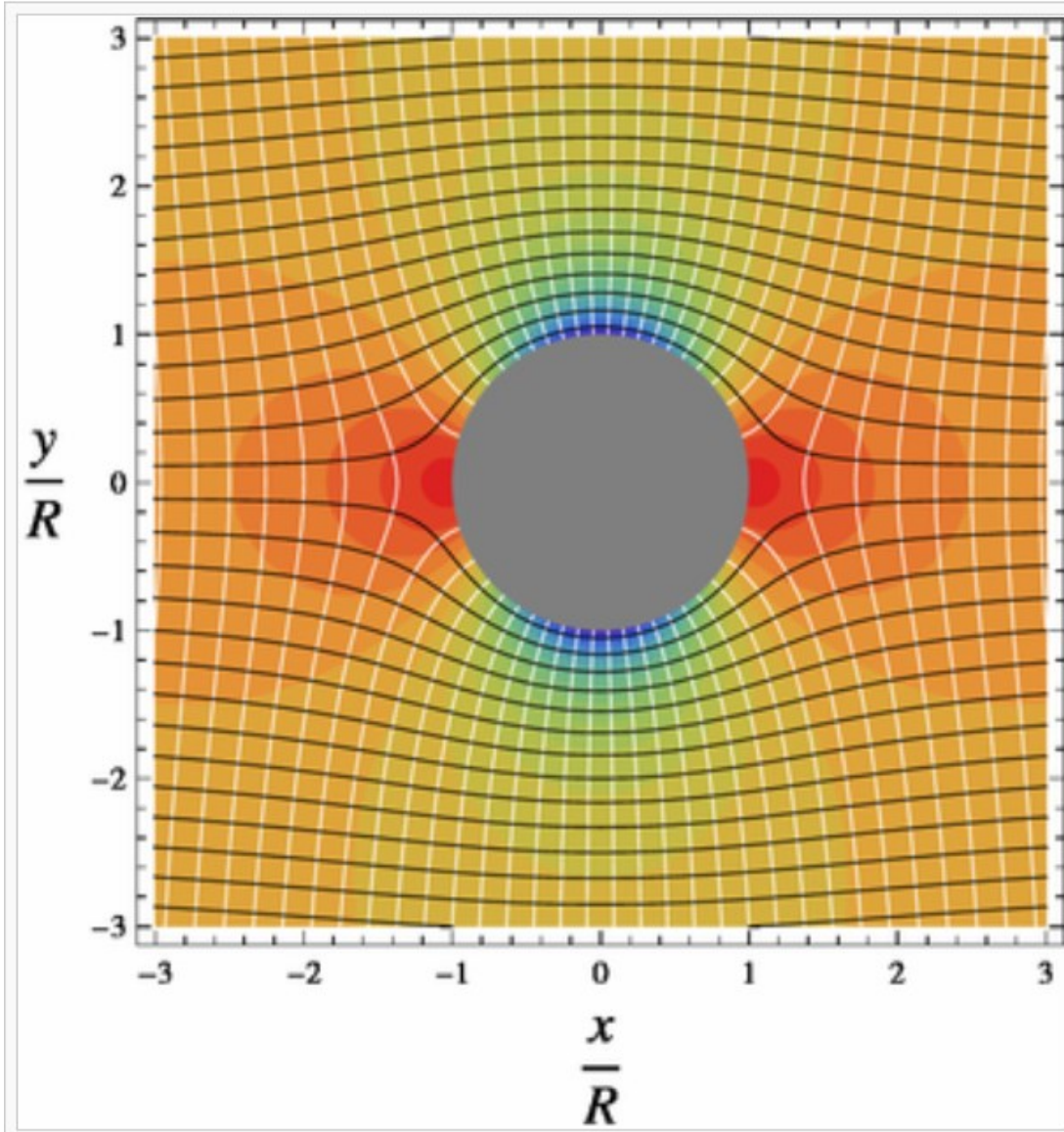
Şekil 4.1 : Silindir etrafındaki potansiyel akışa ait hız vektörleri.

Şekil 4.2’de Demavis tarafından çizilmiş silindir etrafındaki potansiyel akışa ait hız vektörleri görülmektedir.



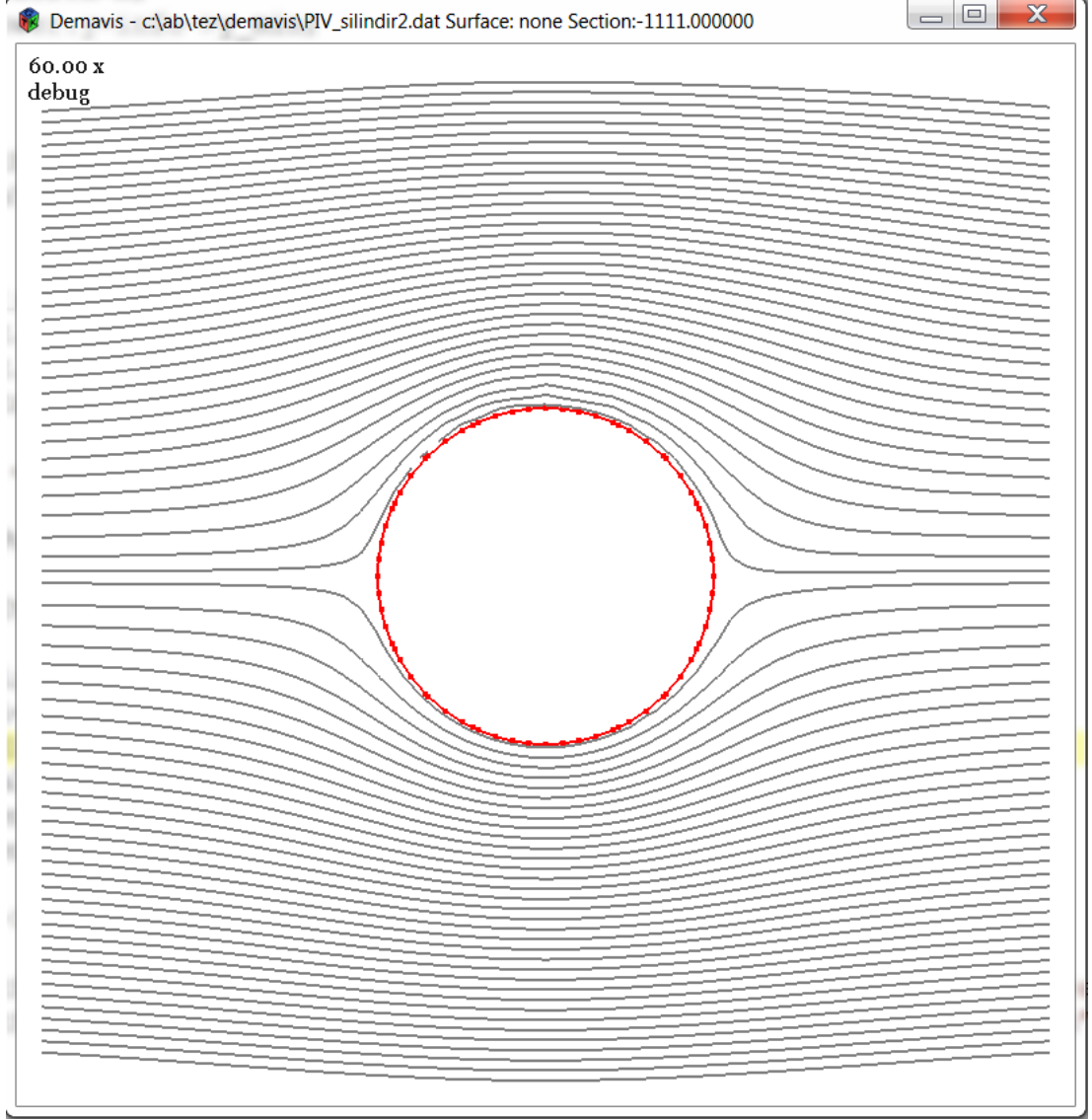
Şekil 4.2 : Demavis tarafından çizilmiş potansiyel akışa ait hız vektörleri.

Şekil 4.3’de silindir etrafındaki potansiyel akışa ait akım çizgileri görülmektedir.



Şekil 4.3 : Silindir etrafındaki potansiyel akımda akım çizgileri.

Şekil 4.4’te potansiyel akım verilerini kullanarak Demavis programının oluşturduğu akım çizgileri görülmektedir.



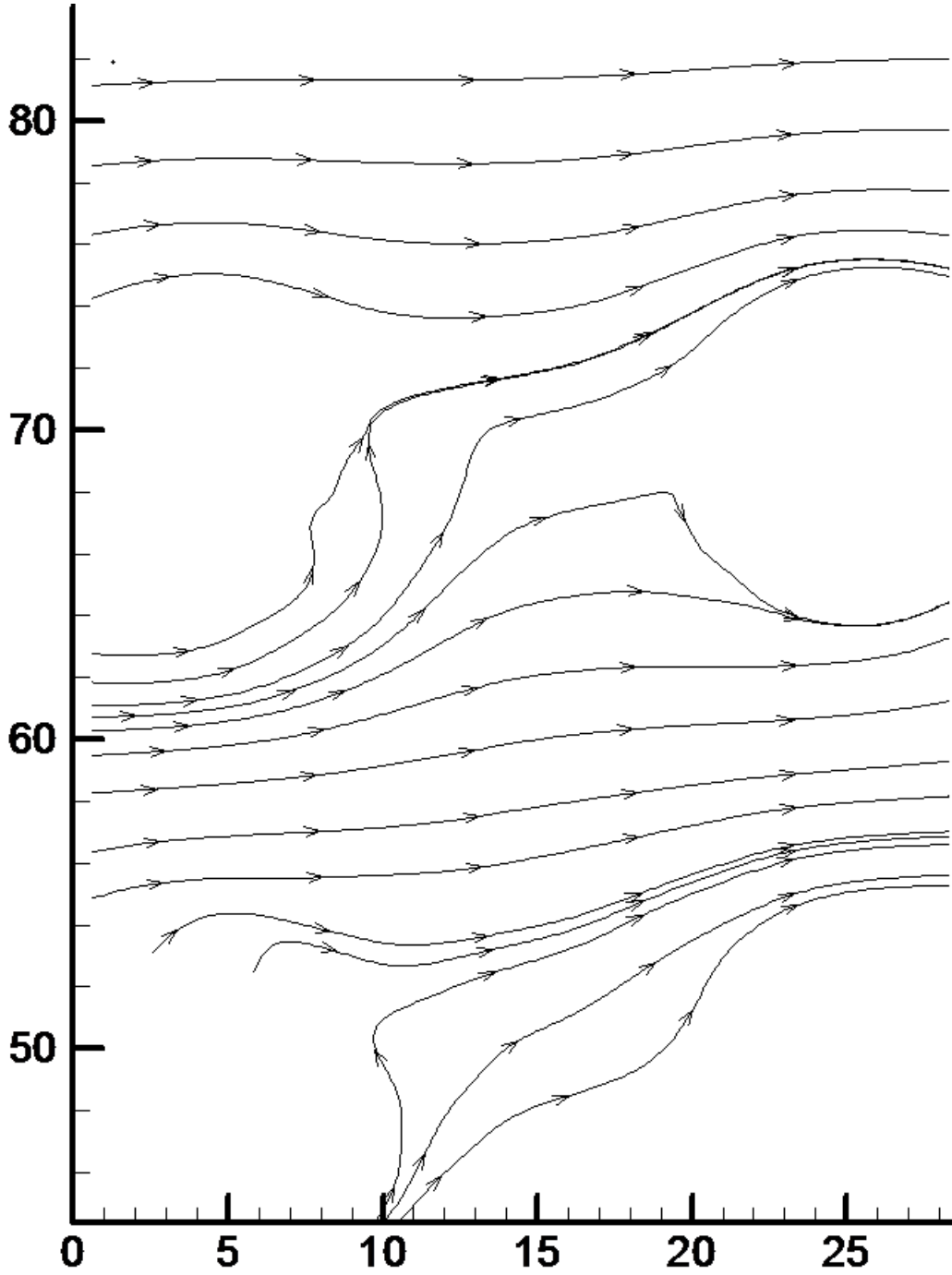
Şekil 4.4 : Demavis programının oluşturduğu akım çizgileri.

Bu karşılaştırmada Demavis tarafından yapılan çizimlerin potansiyel akım ile uyum içinde olduğu gözlenmektedir.

Ayrıca potansiyel fonksiyon değerleri ile Demavis tarafından hesaplanan değerler kıyaslandığında maksimum %0,01 mertebesinde hata görülmüştür.

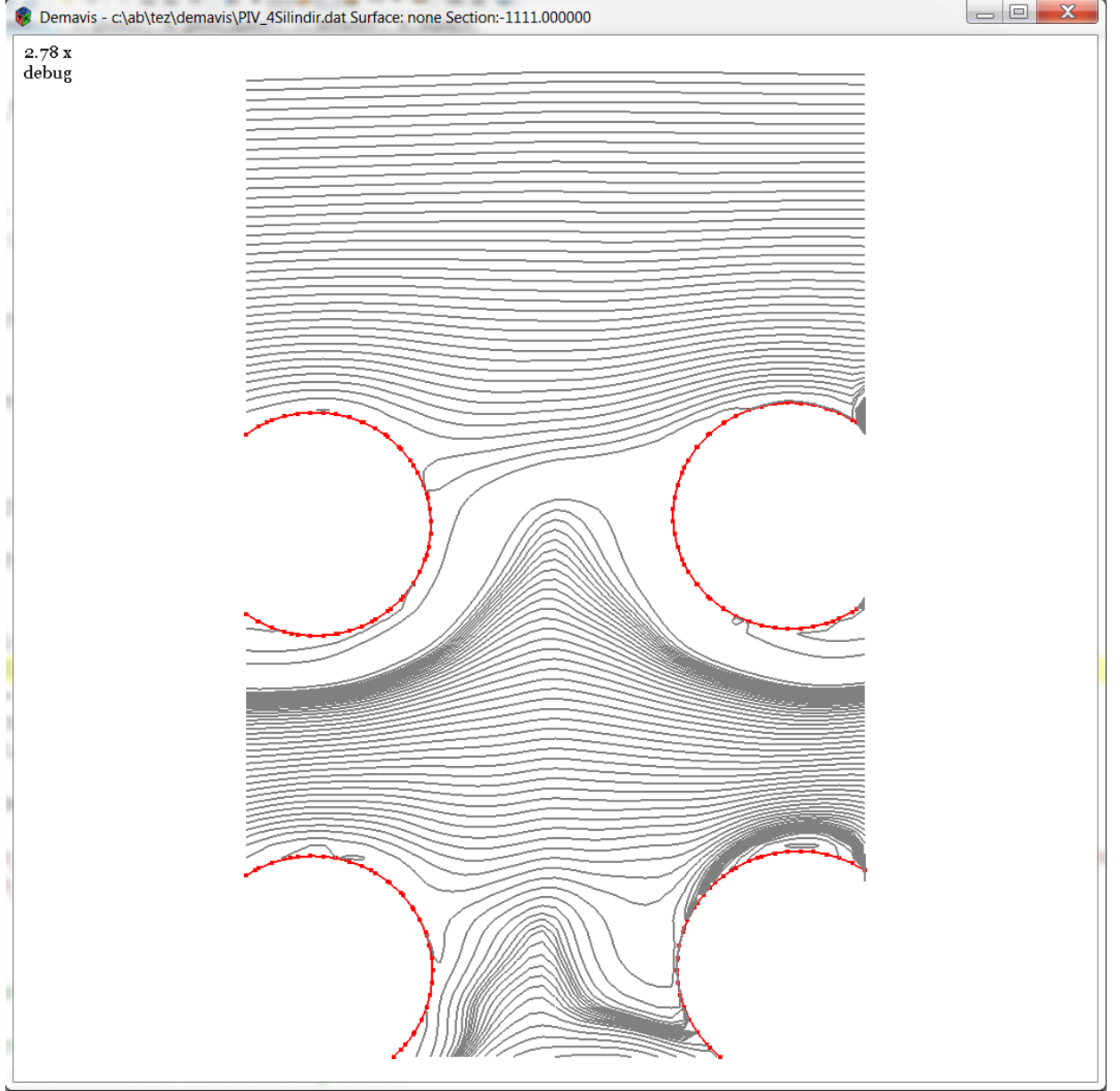
4.3.3 Mevcut uygulamalarla kıyaslama

4 silindir etrafındaki akışa ait TechPlot programı tarafından oluşturulmuş akım çizgileri **Şekil 4.5**'te görülmektedir.



Şekil 4.5 : TechPlot programı tarafından oluşturulmuş akım çizgileri.

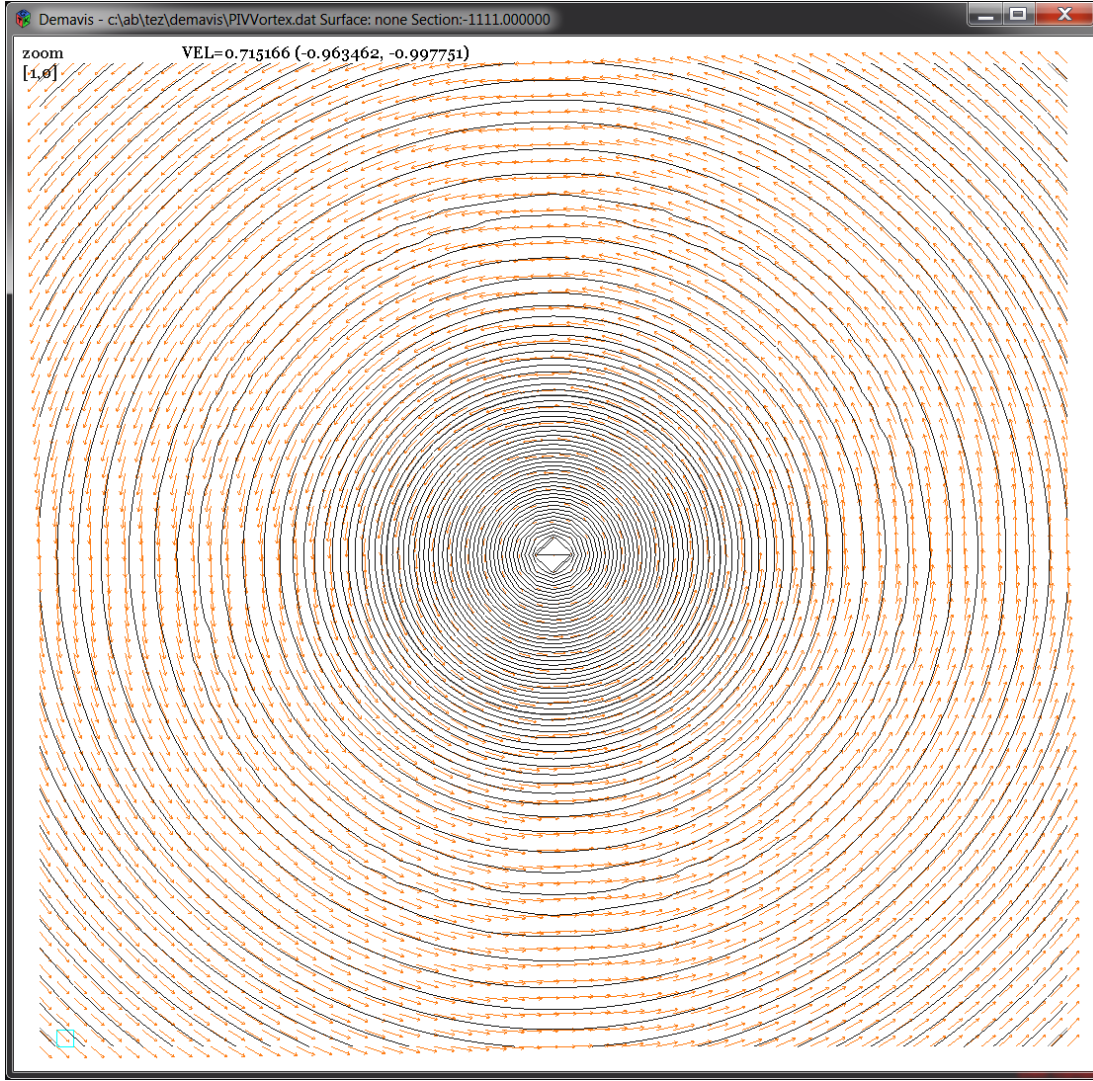
Aynı veri için Demavis programının oluşturduğu akım çizgileri Şekil 4.6'da görülmektedir.



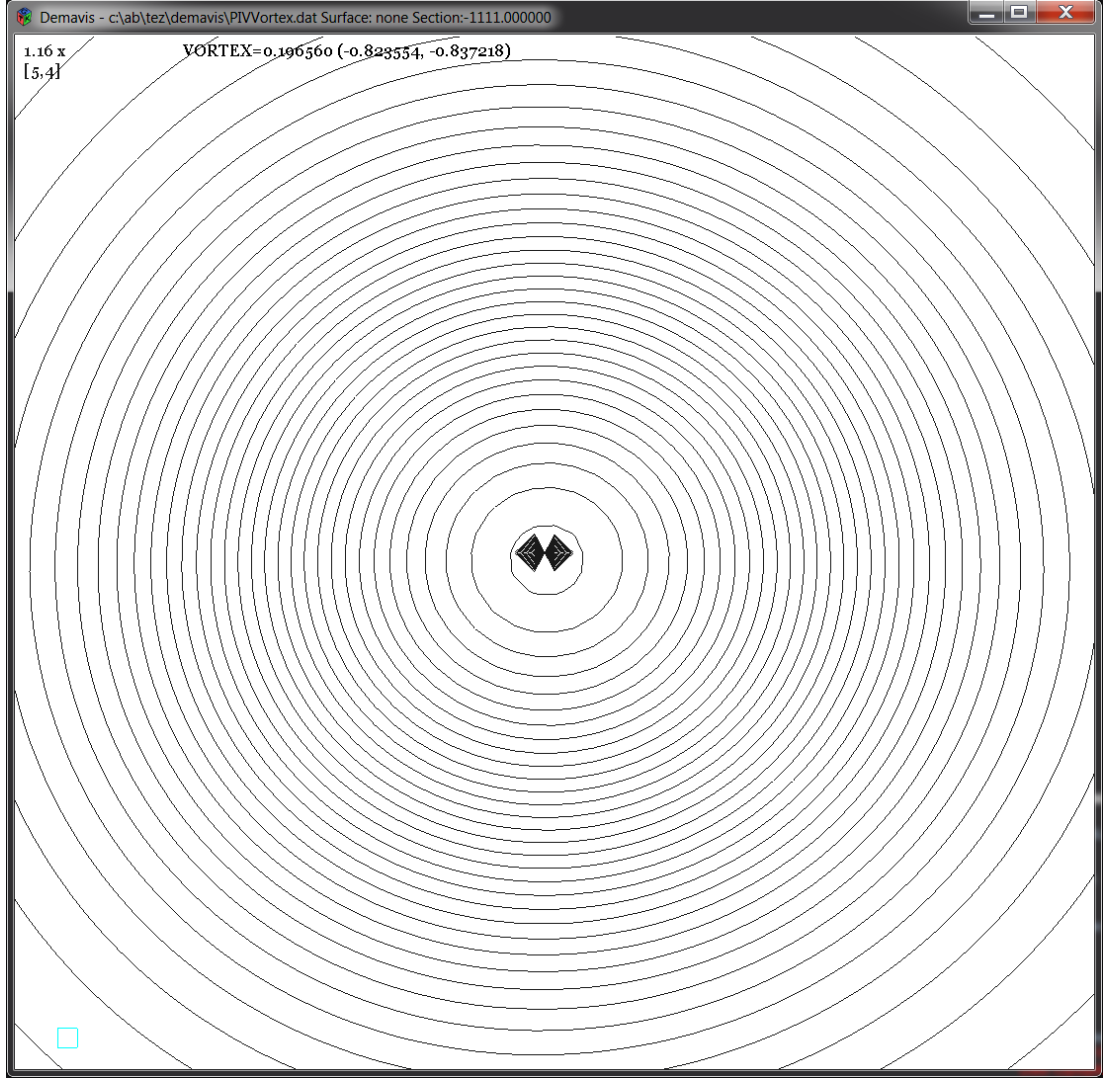
Şekil 4.6 : Demavis programı tarafından oluşturulmuş akım çizgileri.

4.3.4 Girdaplılık Çizimi Doğrulaması

Burgers-Rott girdaplılık çözümünden üretilen teorik nümerik veriler kullanılarak bu girdap akışına ait çizimler Demavis ile yapılmıştır (Drazin, 2006). **Şekil 4.7'**de bu akışa ait hız vektörleri ve eş hız eğrileri **Şekil 4.8'**de de girdaplılık çizimleri görülmektedir.



Şekil 4.7 : Girdap akışına ait hız vektörleri ve eş hız eğrileri.



Şekil 4.8 : Girdap konturları.

Yapılan incelemede Demavis'te tanımlanan algoritmaya uygun şekilde (3.47) bağıntısıyla vortisite değerlerinin hesaplandığı görülmüştür. Bununla beraber kullanılan analitik yönteme ait girdaplılık değerlerinin, algoritma değerlerine göre, akışın özellikle kenar kısımlarında farklılıklar gösterdiği görülmüştür. Bu farkın akımdaki ani değişimlerden kaynaklandığı düşünülmektedir.

4.4 Derleme ve Kurulum

4.4.1 Kullanılan kütüphaneler

Demavis programı GNU C (GNU “GNU’s Not Unix” kelimelerinin ilk harflerinden oluşmuş öz yinelemeli bir kısaltmadır) ve temel olarak standart C dili kütüphaneleri kullanılarak geliştirilmiştir. 64 bitlik işlemci performansının doğrudan kullanılabilmesi için derleyici ve kullanılan tüm kütüphaneler 64 bit olarak veya

destekleyecek şekilde seçilmiştir. Program aşağıdaki kütüphanelerin çalıştığı 64 bit işletim sistemlerinde çalışabilir. Geliştirme 64 bit Linux Gentoo dağıtımı üzerinde yapılmış ve 64 bit Windows 7 üzerinde de test edilmiştir.

Programda kullanılan kütüphaneler ve kullanım amaçları **Çizelge 4.3**'te verilmiştir. gösterilmiştir.

Çizelge 4.3 : Programda kullanılan kütüphaneler.

Kütüphane	Kullanım Amacı
GNU C	Programın derlenerek çalışır hale getirilmesi (Free Software Foundation, t.y.).
MinGW	Microsoft Windows işletim sisteminde derleme (MinGW.org, t.y.)
GTK+	Menü vb. kullanıcı ara yüzünün oluşturulması (GTK+ Team, t.y.).
OpenGL	Çizimlerin gerçekleştirilmesi (Khronos Group, t.y.).
GtkGLExt	GTK+ arayüzü ile OpenGL çizimlerinin gerçekleştirilebilmesi (Gnome, t.y.).
Mathomatic	Cisim yüzeyini tanımlayan cebirsel eşitliklerin kullanılması (Gesslein, t.y.).

4.4.2 Programın derlenmesi

Demavis kaynak kodlarının derlenmesi için Linux ve Windows işletim sistemleri için makefile dosyaları MakefileLinux ve MakefileWindows dosyaları oluşturulmuştur, bu dosyalarda her iki işletim sisteminde GNU C derleyicinin derleme ve link aşamasında kullanacağı kütüphaneler ve C kaynak kodlarının derleme parametre ve ilişkileri tanımlanmıştır. Derlemenin yapılabilmesi için kütüphanelerin bu dosyalar içinde belirtilen dizinlere kurulması gerekmektedir. Dosya içerikleri **Ek B**'de verilmiştir.

Windows işletim sisteminde derleme aşamaları aşağıda verilmiştir.

- **Bölüm 4.4.1**'de verilen kütüphaneler c:\ab dizinine kurulur.

- Demavis programının kaynak kodlarının bulunduğu demavis dizini c:\ab altına kopyalanır.
- Windows çevre değişkeni PATH'e MakefileWindows dosyası içinde belirtildiği gibi kullanılan kütüphanelerin çalışma dizinleri eklenir.
- MinGW kurulumunun içinde gelen msys kullanıcı komut satırı arayüz programı çalıştırılır ve cd c:/ab/demavis komutu ile kaynak kod dizinine geçilir.
- Derleme işlemini başlatmak “make -f MakefileWindows” işleminin yapılmasını sağlayan “win” dosyası çalıştırılır.

Şekil 4.9’de derleme işleminin çıktısı ve oluşan ikili dosya boyutu görülmektedir.

```

MINGW32/c:/ab/demavis
abasel@abasel /c/ab/demavis
$ cd c:/ab/demavis

abasel@abasel /c/ab/demavis
$ win
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
main.c
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
display.c
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
file.c
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
calculation.c
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
gtkwin.c
gcc -Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL `pkg-config --cfl
ags gtkglext-1.0` -I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-15.8.2\lib" -c
equation.c
gcc -o demavis main.o display.o file.o calculation.o gtkwin.o equation.o `pkg-co
nfig --libs gtkglext-1.0` libmathomatic.so

abasel@abasel /c/ab/demavis
$ ls -lh demavis.exe
-rwxr-xr-x 1 abasel Administrators 315K Mar 18 12:58 demavis.exe

abasel@abasel /c/ab/demavis
$

```

Şekil 4.9 : Demavis programının derlenmesi.

Temiz bir derleme olması için görüldüğü gibi tüm derleme adımlarında “-Wall” parametresi kullanılmış ve uyarı seviyesindeki bildirimlerin de hata olarak kabul edilmesi sağlanmıştır. Bu nedenle program derlenme sırasında uyarı dahil gelmeyecek şekilde C kodlaması yapılmıştır.

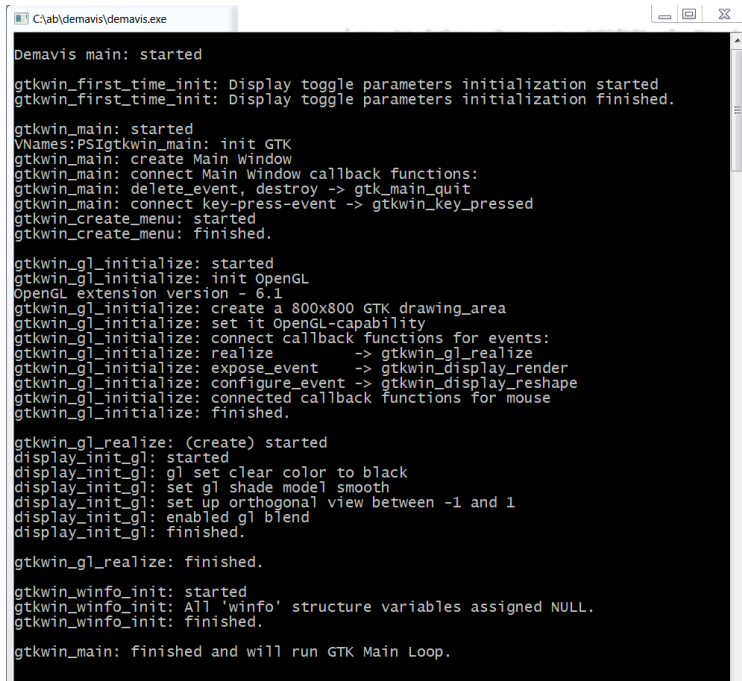
Derleme işleminin çıktısından da görüldüğü gibi demavis programı main.c, display.c, file.c, calculation.c, gtkwin.c, equation.c kaynak kodlarının birlikte derlenmesinden oluşmaktadır ve büyüklüğü 315 KB’dır.

4.4.3 Programın kurulması ve çalıştırılması

Windows işletim sisteminde kurulum ve çalıştırma aşağıdaki şekilde yapılır, yönetici yetkisi gerekmemektedir.

Program çalışma zamanında sadece Gtk64 kütüphane dizinine ihtiyaç duymaktadır, bu kütüphanenin c:\ab\Gtk64 dizinine kopyalanması yeterlidir. Bu dizine programın ulaşabilmesi için çevre değişkeni PATH'e aynı şekilde ilave edilmesi gerekmektedir. Kullanılan diğer kütüphanelerin (gtkglext) dll ve (mathomatic) so dosyaları demavis dizininde bulunmaktadır.

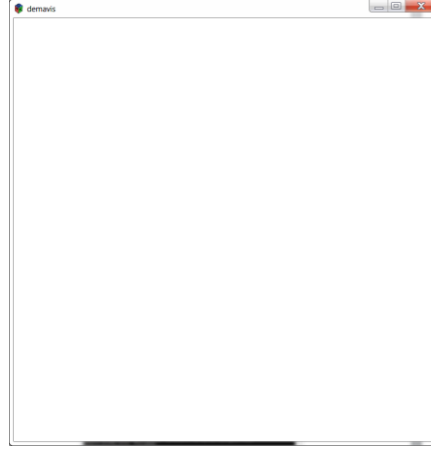
Programı çalıştırmak için programın bulunduğu demavis dizini içindeki demavis.exe programına çift tıklamak yeterli olmaktadır, programla birlikte yapılan işlemlerin kayıtlarını gösteren bir log penceresi de açılmaktadır. Eğer bu loglar daha sonra detaylı incelenmek istenirse bir DOS ekranından demavis dizinine geçilip “demavis > log” şeklinde başlatılabilir. **Şekil 4.10**'da programın açılışı görülmektedir.



Şekil 4.10 : Demavis programının çalıştırılması.

Demavis **Şekil 4.11**'de görüldüğü gibi boş bir ekranla açılmaktadır.

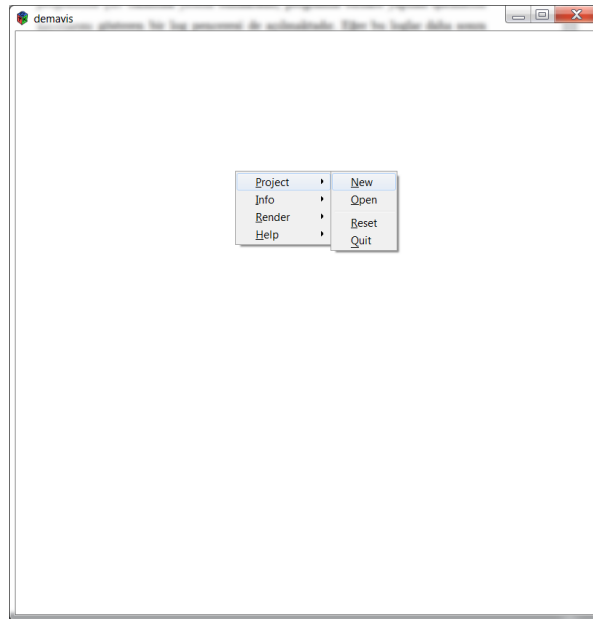
Deney sistemlerinde üretilen veriler genellikle İngilizce formatında ondalık basamakları noktalı olarak alınmaktadır. Buna uyum için Demavis'de de bölgesel ayarlar aynı şekilde yapılarak kullanılmalıdır. Veriler arasında ayraç olarak boşluk kullanılmaktadır.



Şekil 4.11 : Demavis boş açılış ekranı.

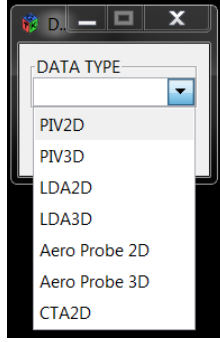
4.5 Programın Kullanımı

Demavis programı etkileşimli olarak çalışmaktadır. Program çalıştırıldığında ana ekran boş olarak açılır. Ekran tüm alanını en verimli şekilde kullanmak için sabit menü kullanılmamıştır. Menü farenin sağ tuşuna basıldığında çıkmaktadır. Program ana ekranın sağ üst köşesindeki X'e basılarak veya menüden Project->Quit seçeneği ile kapatılabilir. Programın ana menüsü **Şekil 4.12**'de görülmektedir.



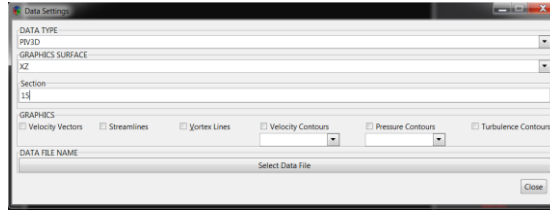
Şekil 4.12 : Demavis'in ana menüsü.

Project->New menüsü ile işlenecek verinin tanımlanacağı “Data Settings” penceresi açılır. Öncelikle **Şekil 4.13**'te görülen listeden kullanılan veri tipi “DATA TYPE” alanında seçilir.



Şekil 4.13 : Kullanılacak veri tipinin seçilmesi.

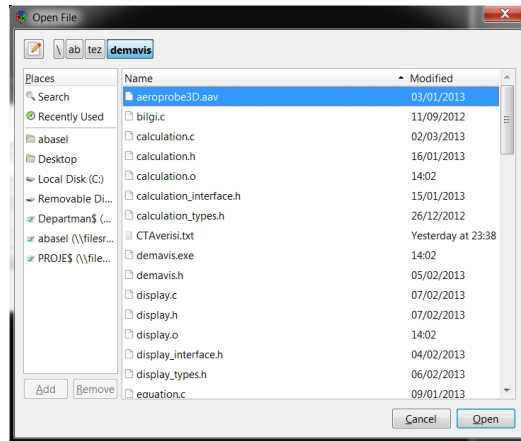
3 Boyutlu veriye sahip dosyalar için hangi düzlemin kullanılacağı **Şekil 4.14**'de görülen “GRAPHICS SURFACE” alanında seçilir. Eğer seçilen veri tipi 2 boyutlu ise bu alan doğrudan XY seçilmiş olarak gelir; buradaki XY'nin özel bir anlamı yoktur verideki düzlem kullanılacaktır.



Şekil 4.14 : Kullanılacak verinin tanımlandığı “Data Settings” penceresi.

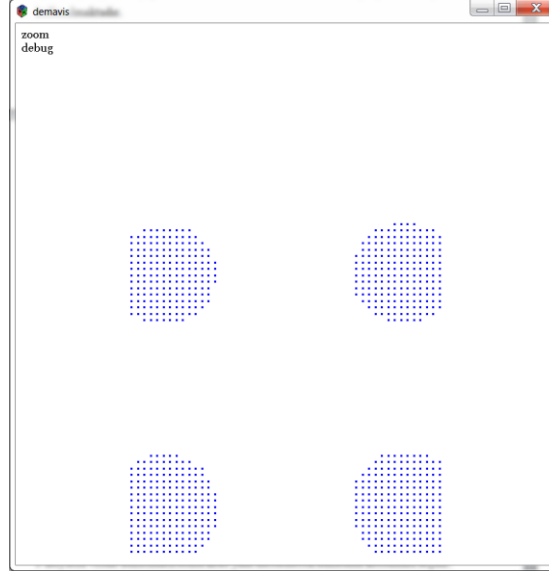
3 Boyutlu veride düzlemden sonra kesit yani incelenecek düzlemin koordinatı seçilir. Program sadece bu düzlemdeki verilerle işlem yapacaktır. 2 Boyutlu veride bu alanın gözükmemektedir.

Daha sonra kullanılacak veri dosyası “Select Data File” ile seçilmek üzere **Şekil 4.15**'te görülen dosya tarayıcı penceresi açılır. Bu pencere kullanılarak istenen veri dosyası seçimi yapılır.



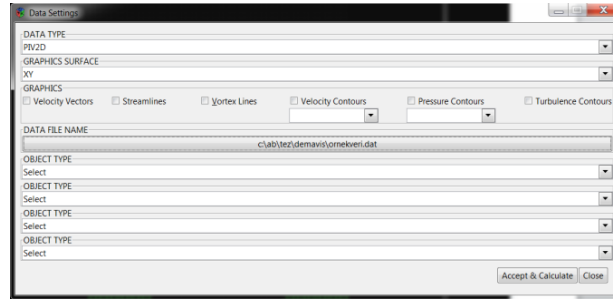
Şekil 4.15 : Veri dosyası seçimi için tarayıcı pencere.

Seilen veri okunarak lm noktalarını gsteren ekrana izilir. Veri tipi PIV ise veri iindeki “Status” kolonu kullanılarak akış iinde ka farklı katı cisim bulunduėu da belirlenir ve ekranda katı cisim noktaları **Şekil 4.16**’da grldėu gibi ekrana izilir.



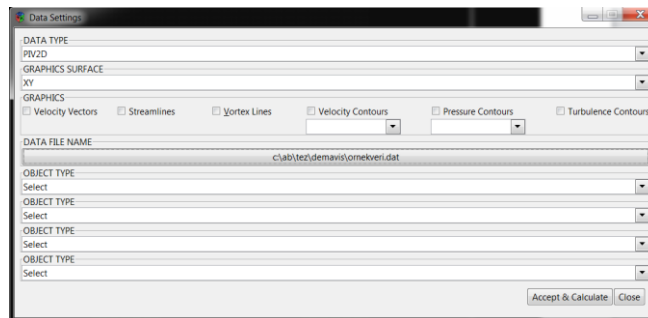
Şekil 4.16 : lm ve katı cisim noktalarının izimi.

“Data Settings” pencersinde aynı sayıda katı cisim alanı **Şekil 4.17**’de grldėu şekilde oluřturulur.



Şekil 4.17 : Katı cisimler iin tanımlama satırları.

Diėer veri tiplerinde akım iinde ka katı cisim bulunduėu kullanııcıdan alınmakta ve bunlara iliřkin veri giriř alanları **Şekil 4.18**’de grldėu şekilde hazırlanmaktadır.



Şekil 4.18 : Cisim sayısının kullanıcı tarafından girilmesi.

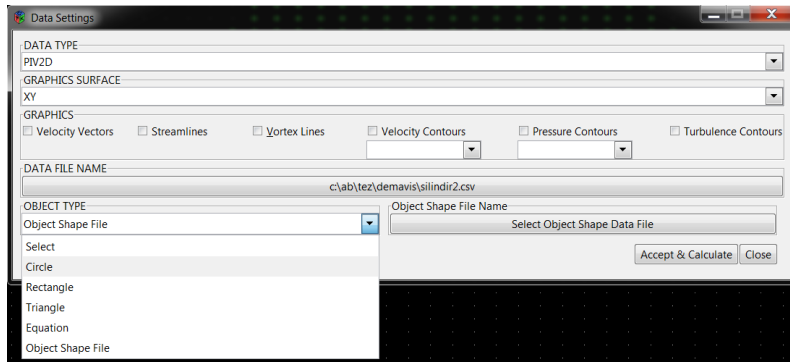
4.5.1 Katı cisim tanımlama

Demavis programında daire, dikdörtgen, üçgen, denklem ve obje dosyası ile katı cisim tanımlanmaktadır.

4.5.1.1 Daire Tanımlama

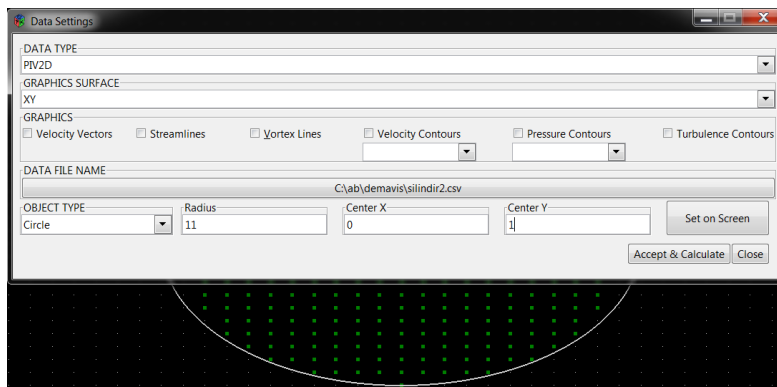
Daire tanımlaması aşağıda anlatıldığı şekilde yapılır.

Cisim tipi olarak daire “Circle” seçildiğinde **Şekil 4.19**’da görüldüğü gibi yarıçap ve merkezin koordinatları kullanıcıdan alınır.



Şekil 4.19 : Katı cisim tipleri.

Veri girişiyle eş zamanlı olarak çizim de gerçekleştirilir. Kullanıcı “Set on Screen” düğmesine basarak, ekran üzerinde dairenin çapını ok tuşlarıyla büyütüp küçültebilir ve fare ile de yerini değiştirebilir. Nihai konumlama yapıldığında shift ve z tuşlarına basılarak işlem tamamlanır ve “Data Settings” penceresine geri dönlür. Kullanıcı burada isterse yine etkileşimli olarak **Şekil 4.20**’deki ekran ile daha hassas ayarlama da yapabilir.

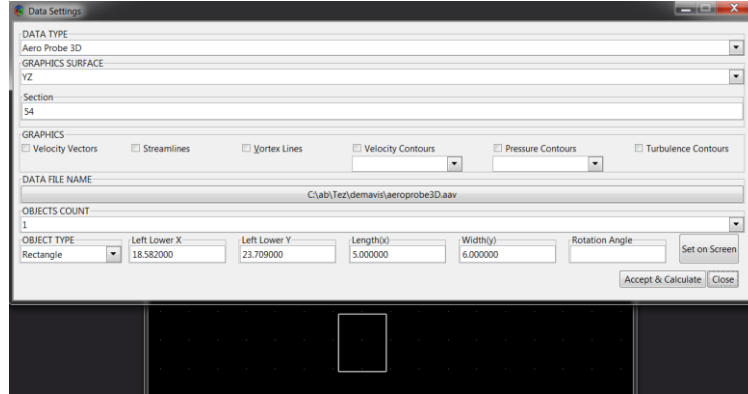


Şekil 4.20 : Demavis daire tanımlama ekranı.

4.5.1.2 Dikdörtgen Tanımlama

Dikdörtgen tanımlaması aşağıda anlatıldığı şekilde yapılır.

Cisim tipi olarak “Rectangle” seçildiğinde **Şekil 4.21**'de görüldüğü gibi dikdörtgenin sol alt köşesinin koordinatları, uzunluğu ve genişliği alınır. Bu büyüklükler girildikten sonra istenirse “Set on Screen” düğmesine tıklanarak grid üzerinde yerleştirme ve ok tuşlarıyla büyütme, küçültme yapılabilir.



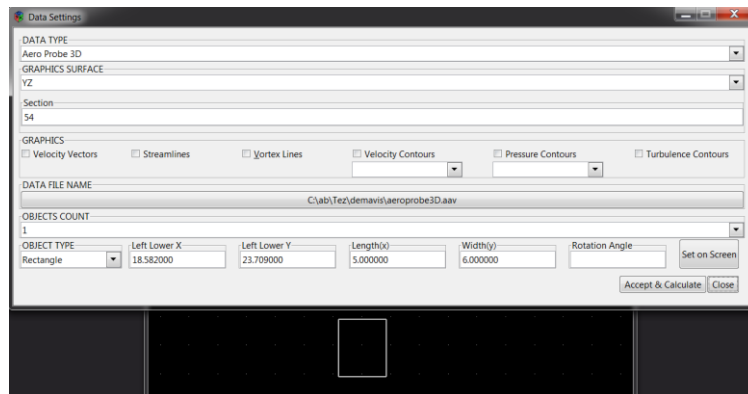
Şekil 4.21 : Demavis dikdörtgen tanımlama ekranı.

Programa daha sonradan bir de dönme açısı “Rotation Angle” eklenmiştir, verilen açı kadar dikdörtgen döndürülmektedir, fakat bu durum için cisim dikdörtgen içinde kalan grid noktalarını belirleme işleme tamamlanmadığı için bu seçenek tam fonksiyonel değildir.

4.5.1.3 Üçgen tanımlama

Üçgen tanımlaması aşağıda anlatıldığı şekilde yapılır.

Cisim tipi olarak “Triangle” seçildiğinde **Şekil 4.22**'de görüldüğü gibi üçgenin üç noktasının da koordinatları alınır.



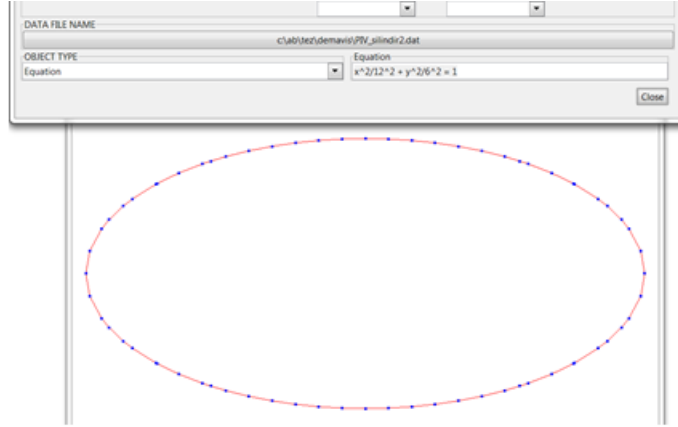
Şekil 4.22 : Demavis üçgen tanımlama ekranı.

Bu büyüklükler girildikten sonra istenirse “Set on Screen” düğmesine tıklanarak grid üzerinde yerleştirme yapılabilir.

4.5.1.4 Denklem ile tanımlama

İstenirse cisim konturu bir denklemle de tanımlanabilir.

Cisim tipi olarak “Equation” seçildiğinde **Şekil 4.23**’de görüldüğü gibi $f(x, y)=0$ şeklinde bir kapalı eğri denklemi girilir.

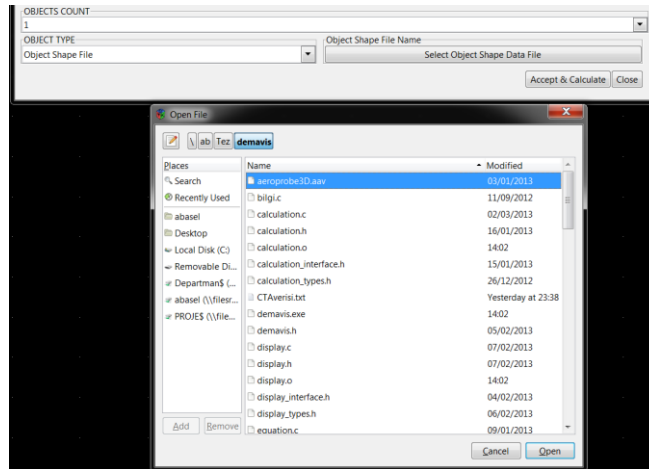


Şekil 4.23 : Demavis denklem ile cisim tanımlama ekranı.

4.5.1.5 Şekil dosyası ile tanımlama

İstenirse cisim yüzeyi kullanıcı tarafından oluşturulmuş bir koordinat dosyasından da okunabilecektir.

Cisim tipi olarak “Object Shape File” seçildiğinde **Şekil 4.24**’de görüldüğü gibi dosyayı yüklemek için bir tarayıcı açılır. Bu fonksiyonalitye henüz tamamlanmamıştır.



Şekil 4.24 : Demavis denklem dosyası okuma ekranı.

4.5.2 Cisimi yakınlaştırma, uzaklaştırma ve kaydırma işlemleri

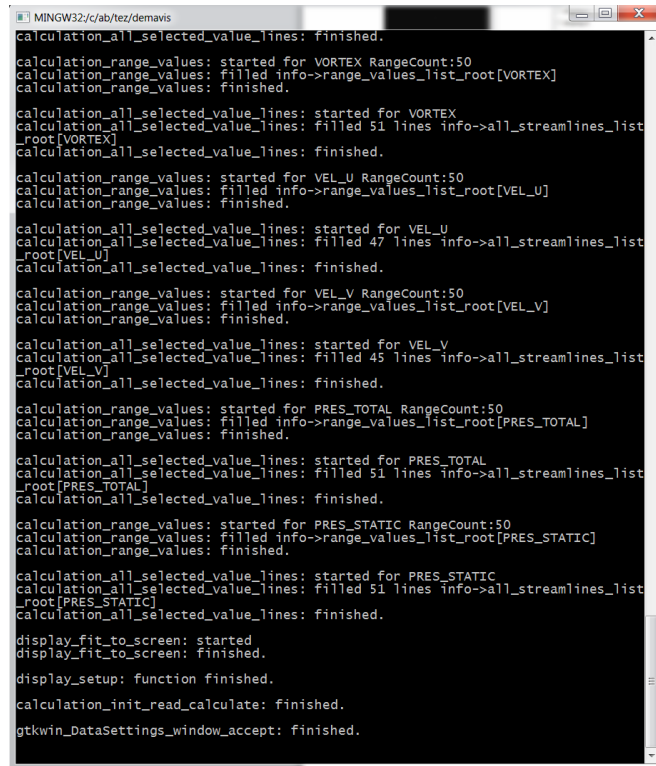
Çizimi Büyütme ve küçültme işlemleri farenin orta tuşuna basılarak ileri ve geri hareket ettirilmesiyle yapılır.

Çizimi kaydırma işlemi de farenin sol tuşuna basılarak hareket ettirilmesiyle gerçekleştirilir.

4.5.3 Hesaplamaların yaptırılması

Data Properties penceresindeki *Accept & Calculate* düğmesine basılarak çizimlerin yapılması için gerekli işlemler başlatılır. Veriye ilişkin çizilebilecek bütün çizimler için hesaplamalar yapılır ve çizimleri oluşturan noktalar belirlenerek hızlı kullanım için bunlar direk hafızada sıralı listelere yerleştirilir.

Program konsol ekranında yaptığı işlemleri C fonksiyonları bazında belirtmektedir. Hesaplama gerçekleştikten sonra bu düğme pencereden kaldırılır. Bundan sonra “Render” menüsü kullanılarak istenen çizimler birlikte veya ayrı ayrı hızlı bir şekilde çizdirilebilir. Hesaplamaların yapılması genelde bir dakikadan kısa bir süre almaktadır. **Şekil 4.25**'de hesaplamalar tamamlandığında oluşan kayıtlar görülmektedir. tamamlanmamıştır.

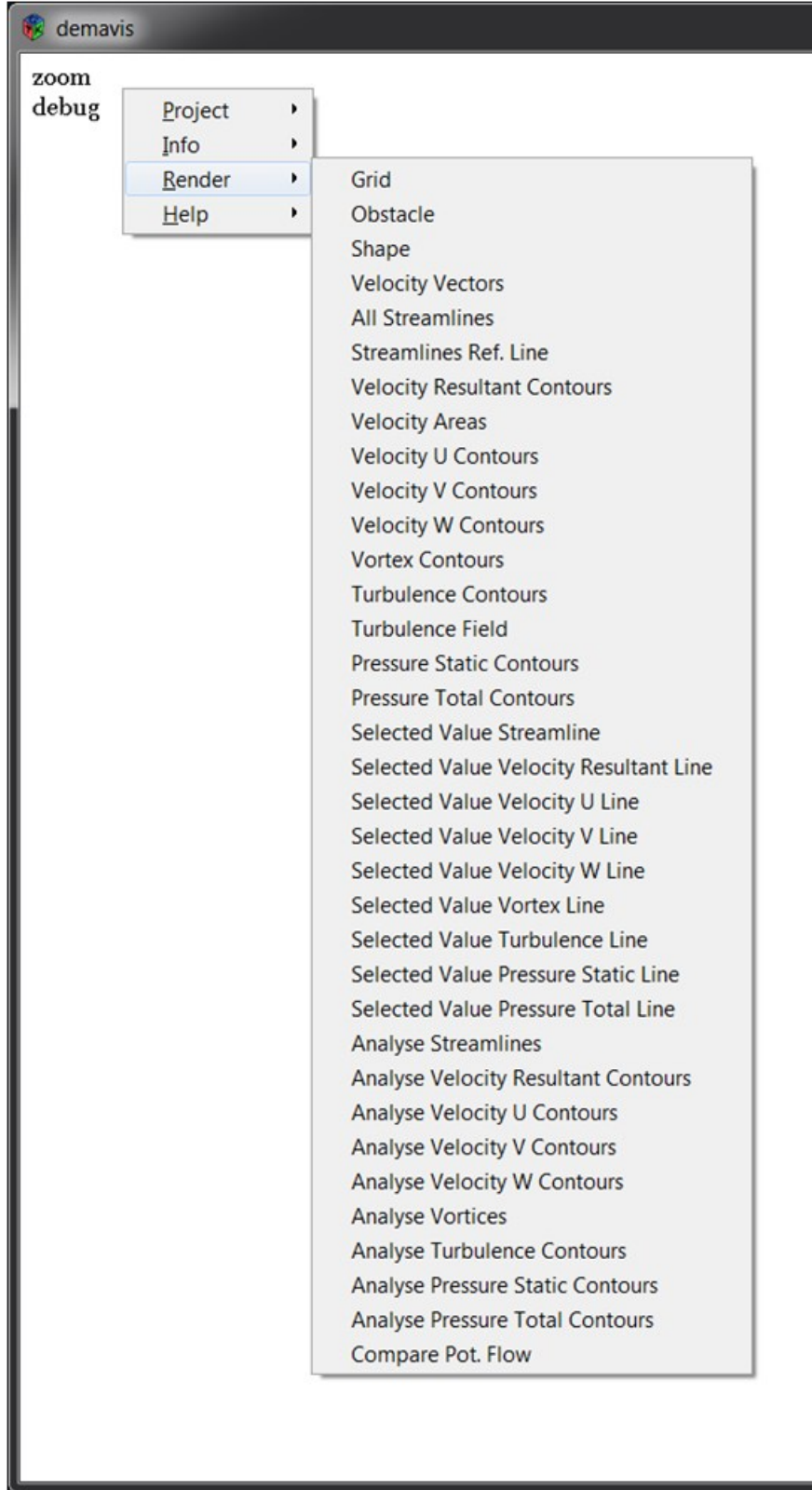


```
MINGW32/c/ab/tez/demavis
calculation_all_selected_value_lines: finished.
calculation_range_values: started for VORTEX RangeCount:50
calculation_range_values: filled info->range_values_list_root[VORTEX]
calculation_range_values: finished.
calculation_all_selected_value_lines: started for VORTEX
calculation_all_selected_value_lines: filled 51 lines info->all_streamlines_list_root[VORTEX]
calculation_all_selected_value_lines: finished.
calculation_range_values: started for VEL_U RangeCount:50
calculation_range_values: filled info->range_values_list_root[VEL_U]
calculation_range_values: finished.
calculation_all_selected_value_lines: started for VEL_U
calculation_all_selected_value_lines: filled 47 lines info->all_streamlines_list_root[VEL_U]
calculation_all_selected_value_lines: finished.
calculation_range_values: started for VEL_V RangeCount:50
calculation_range_values: filled info->range_values_list_root[VEL_V]
calculation_range_values: finished.
calculation_all_selected_value_lines: started for VEL_V
calculation_all_selected_value_lines: filled 45 lines info->all_streamlines_list_root[VEL_V]
calculation_all_selected_value_lines: finished.
calculation_range_values: started for PRES_TOTAL RangeCount:50
calculation_range_values: filled info->range_values_list_root[PRES_TOTAL]
calculation_range_values: finished.
calculation_all_selected_value_lines: started for PRES_TOTAL
calculation_all_selected_value_lines: filled 51 lines info->all_streamlines_list_root[PRES_TOTAL]
calculation_all_selected_value_lines: finished.
calculation_range_values: started for PRES_STATIC RangeCount:50
calculation_range_values: filled info->range_values_list_root[PRES_STATIC]
calculation_range_values: finished.
calculation_all_selected_value_lines: started for PRES_STATIC
calculation_all_selected_value_lines: filled 51 lines info->all_streamlines_list_root[PRES_STATIC]
calculation_all_selected_value_lines: finished.
display_fit_to_screen: started
display_fit_to_screen: finished.
display_setup: function finished.
calculation_init_read_calculate: finished.
gtkwin_DataSettings_window_accept: finished.
```

Şekil 4.25 : Hesaplama işlemlerinin sonu.

4.5.4 Grafiklerin çizdirilmesi

Hesaplamalar yapıldıktan sonra ana menüdeki “Render” seçeneği ile Şekil 4.26’da görülen menü ile istenen grafikler ayrı ayrı veya birlikte çizdirilebilirler.

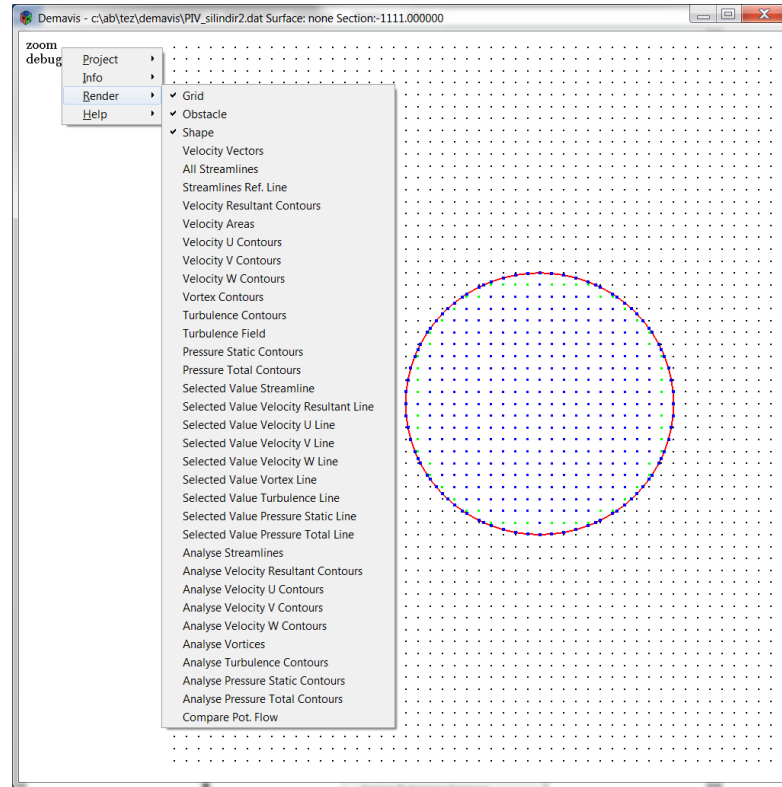


Şekil 4.26 : Demavis çizilebilecek grafikler menüsü.

Bu menünün 3 kısımdan oluştuğu düşünülebilir. İlk kısımda grid noktaları, katı cisim noktaları ve cisim konturu ile deney ortamının çizimi yapılmaktadır. İkinci kısım hız vektörleri ile başlamakta toplam basıncı içine almaktadır, burada bu büyüklüklerin çizimleri yapılmaktadır. Devam eden üçüncü kısım ise sıralı olarak bunların ekran üzerinde farenin bulunduğu yerden geçen çizgilerinin çizildiği seçeneklerdir. Dördüncü ve son bölüm de yine bu büyüklüklerin incelemelerinin yapılabildiği analiz seçeneklerinden oluşmaktadır.

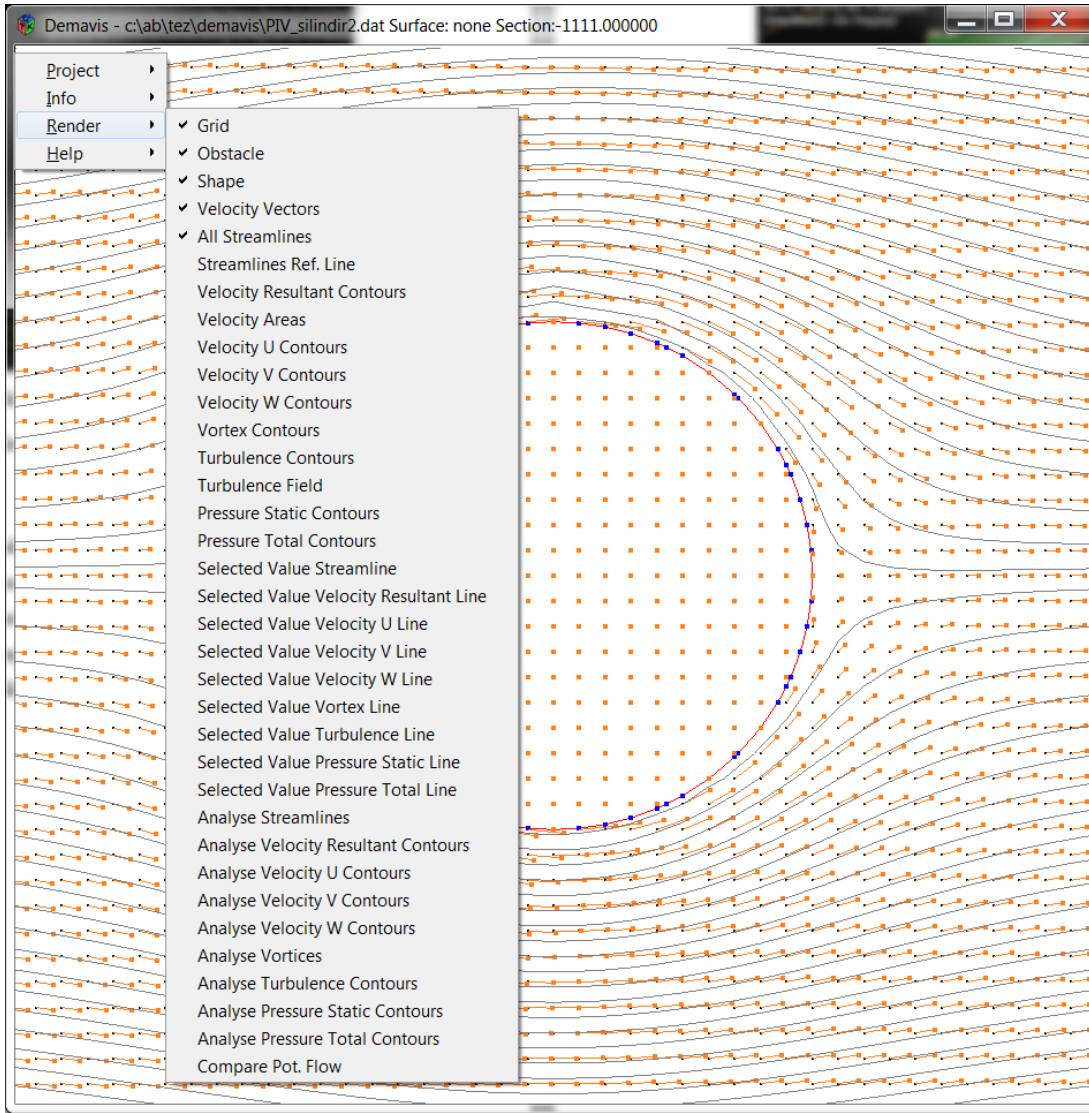
Bu bölümde bu menü kullanılarak oluşturulabilecek çizimlerin bazılarına yer verilecektir, detaylı inceleme program üzerinden yapılabilir.

Bir veri dosyası okutulup hesaplama başlatıldığında program **Şekil 4.27**'de görüldüğü gibi varsayılan olarak ilk üç seçeneği seçerek grid ve katı cisim noktalarını göstermekte ve cisim konturunu çizmektedir.



Şekil 4.27 : Programda varsayılan olarak çizilen büyüklükler.

Bundan sonra **Şekil 4.28**'de görüldüğü gibi sırasıyla hız vektörleri ve akım çizgileri çizimlere eklenerek incelemeye devam edilebilir.

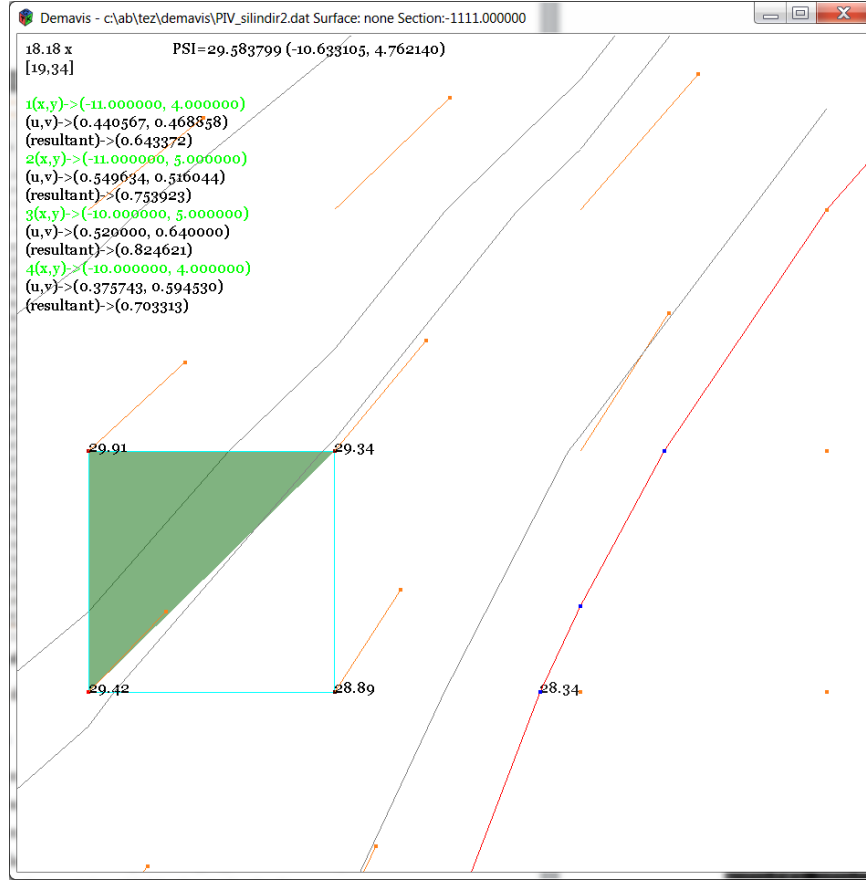


Şekil 4.28 : Hız vektörleri ve akım çizgileri bir arada.

Diğer büyüklükler de isteğe göre eklenip çıkartılarak çizimleri yapılabilir.

4.5.5 Analiz modu

Şekil 4.29’da yukarıdaki seçimlere akım çizgisi analiz fonksiyonu eklenmiş durum görülmektedir. Burada seçili noktanın bulunduğu grid hücresi, bu hücredeki üçgen alan, grid noktalarında akım fonksiyonu değerleri, ekranın üstünde bulunan noktanın koordinatları ve burada hesaplanan değer görülmektedir. Ayrıca sol tarafta grid hücresinin noktalarına ait büyüklükler de görülmektedir.



Şekil 4.29 : Seçilen noktadaki akım çizgisi ve hız alanı eklenmiş görünüm.

Yine bu modda klavyeden “P” tuşuna basıldığında Şekil 4.30’da görüldüğü gibi hücre referans noktası olan sol alt noktaya ait tüm büyüklüklerin değerleri görülebilmektedir.

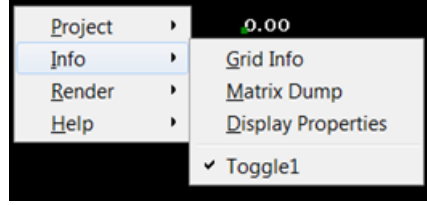
```
file_dump_matrix_cell: started for (19, 34)
nx=61, ny=61
matrix[19, 34] (x:-11.000000, y:4.000000) , status=16, obj_nbr=0
Velocities(u: 0.440567, v: 0.468858), Resultant Velocity=0.643372
PSI= 29.419318
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[PSI]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[PSI]=-1.000000,
VEL= 0.643372
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[VEL]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[VEL]=-1.000000,
PRES_STATIC= 0.000000
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[PRES_STATIC]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[PRES_STATIC]=-1.000000,
PRES_TOTAL= 0.000000
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[PRES_TOTAL]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[PRES_TOTAL]=-1.000000,
VORTEX= 0.016605
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[VORTEX]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[VORTEX]=-1.000000,
TURB= 0.000000
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[TURB]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[TURB]=-1.000000,
VEL_U= 0.440567
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[VEL_U]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[VEL_U]=-1.000000,
VEL_V= 0.468858
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[VEL_V]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[VEL_V]=-1.000000,
VEL_W= 0.000000
Yconst_avg=0, Yconst_x=0.000000, Yconst_value[VEL_W]=-1.000000,
Xconst_avg=0, Xconst_y=0.000000, Xconst_value[VEL_W]=-1.000000,
file_dump_matrix_cell: finished.
```

Şekil 4.30 : Hücre referans noktasına ait tüm büyüklükler.

Analiz modu menüde görüldüğü gibi tüm büyüklükler için çalışmaktadır.

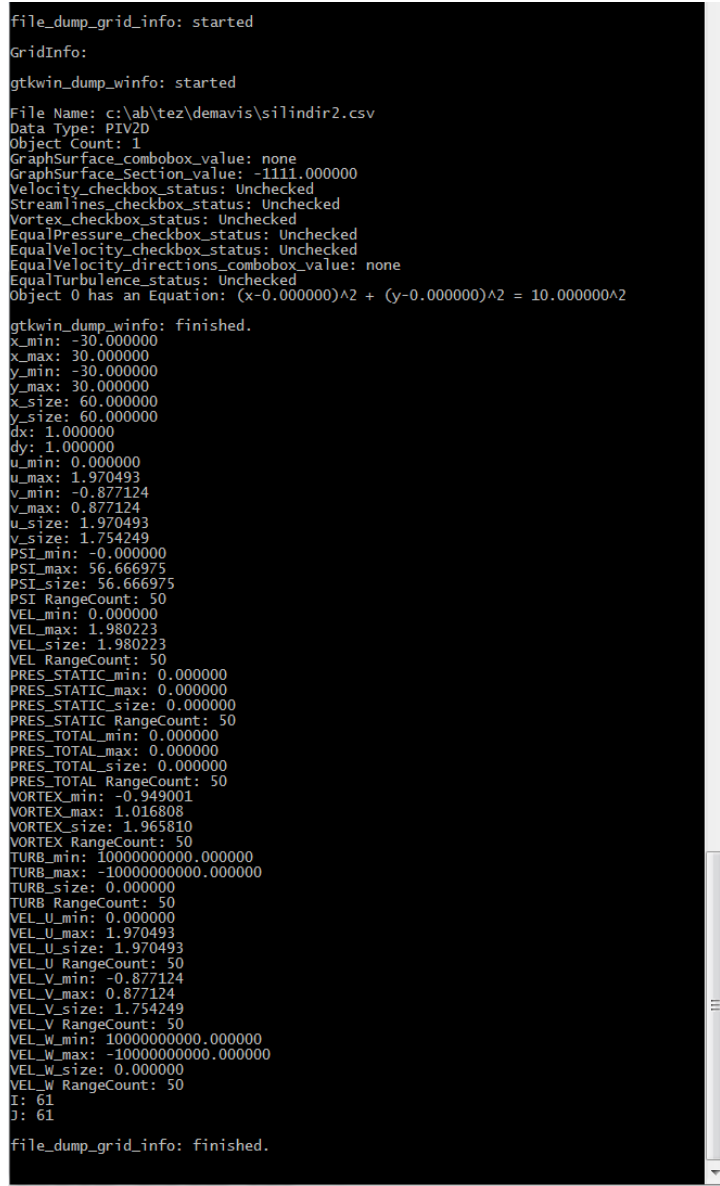
Ayrıca Şekil 4.31’de görülen “Info” bilgi menüsü kullanılarak gride ait genel bilgiler

Şekil 4.32’de görüldüğü şekilde elde edilebilir.



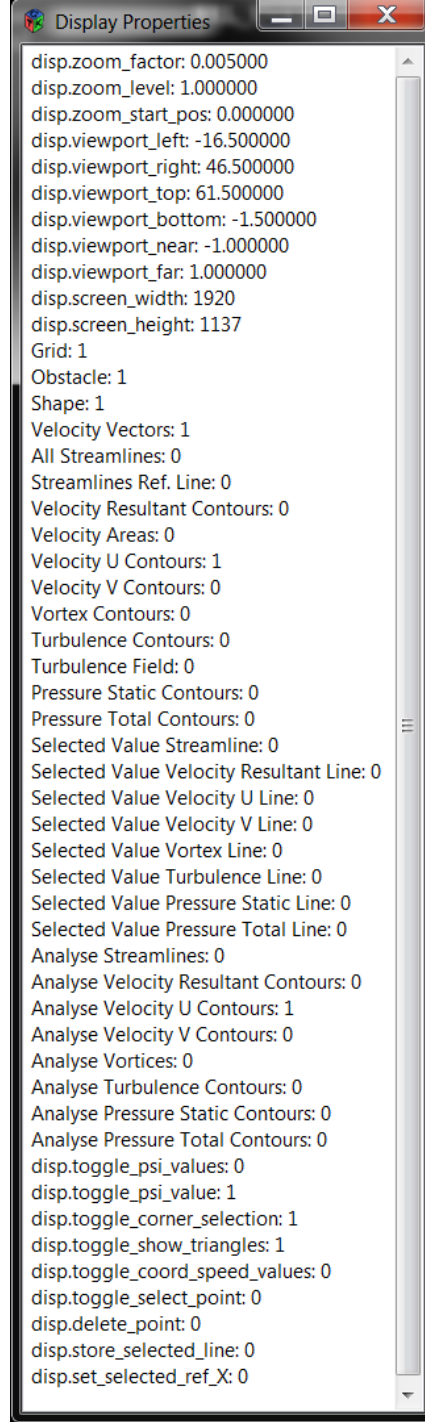
Şekil 4.31 : Bilgi menüsü.

Benzer şekilde “Matrix Dump” seçeneği ile tüm grid noktalarındaki veriler de alınabilir.



Şekil 4.32 : Grid genel bilgileri.

Ayrıca Info menüsündeki “Display Properties” seçeneği **Şekil 4.33**’de görülen yeni bir pencere yardımıyla programdaki görüntüleme anahtarları ve hesaplanan bazı önemli değerler program boyunca izlenebilir, pencere bir tuşa basıldığında verileri güncellemektedir.



Şekil 4.33 : Program içindeki değişkenlerin izlenmesi.

4.5.6 Çizgi ekleme

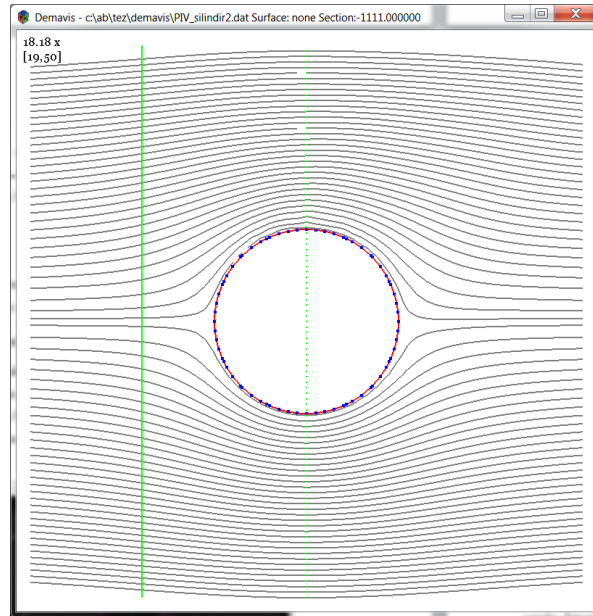
Seçilen noktadaki çizimlerin yapılması sırasında oluşturulan çizimlerin mevcut çizimlere eklenmesi sağlanabilir. Bunun için klavyeden z tuşuna basılır. Bu işlem sırasında tüm ilgili büyüklüğe ait tüm çizgilerin çizilmesi seçeneği de işaretlenmiş olmalıdır.

4.5.7 Çizgi silme

Benzer şekilde istenmeyen çizgilerin silinmesi de referans noktalar baz alınarak yapılabilir, fakat bu işlem henüz programa eklenmemiştir.

4.5.8 Akım çizgileri referans doğrusu yerinin değiştirilmesi

Varsayılan olarak akım çizgilerinin referans doğrusu **Şekil 4.34**'de görüldüğü gibi gridin tam ortasında oluşturulmaktadır. Tüm akım çizgilerinin çizdirilmesi sırasında klavyeden f tuşuna basılarak bu çizginin yeri istendiği şekilde değiştirilir, Z tuşuna basıldığında yeni konuma göre referans değerleri tekrar hesaplanarak tüm çizgiler tekrar çizilir.



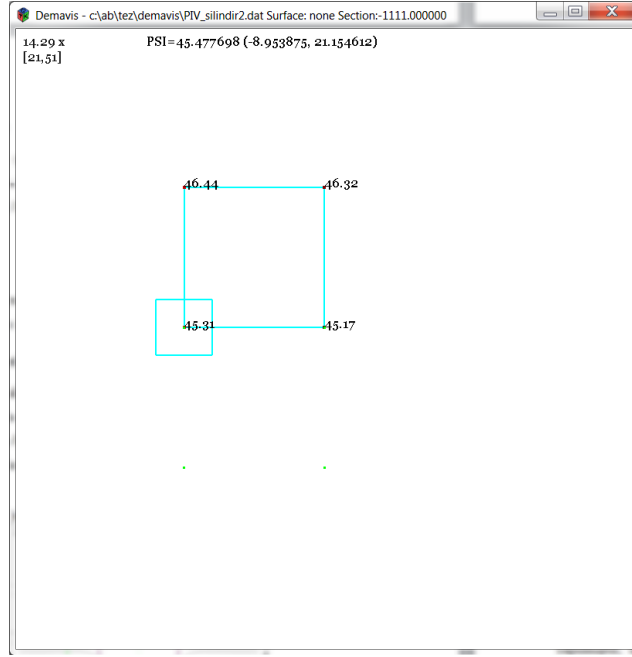
Şekil 4.34 : Akım çizgileri referans noktaları doğrusu.

Ortada yeşil ile gösterilen referans çizgisinin sol kenara kaydırıldığı görülmektedir.

4.5.9 Seçilen grid noktalarının katı cism yapılması

Cisime çok yakın bazı grid noktalarındaki ölçümlerin hatalı olma ihtimali nedeniyle bunların katı cisimmiş gibi hesaplamaya katılması gerekebilmektedir. Bu nedenle bu amaç için bir grid noktası silme fonksiyonallitesi geliştirilmiştir.

Analiz modundayken d tuşuna basılarak bu moda geçilir, grid seçimi **Şekil 4.35**'de görülen kutucuk ile yapılır.



Şekil 4.35 : Grid noktası silme modu.

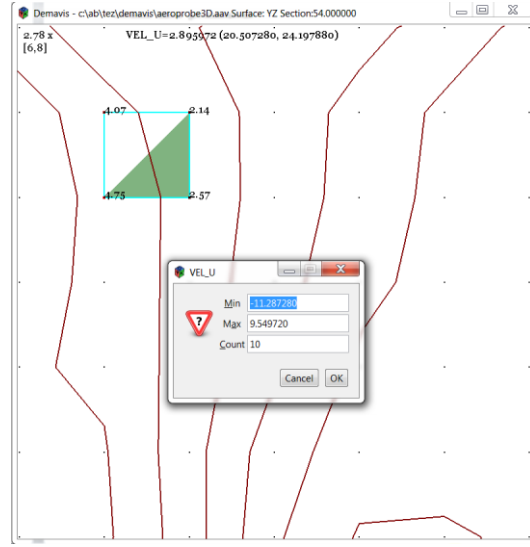
Seçilen boşluk tuşuna basılarak silinir ve katı cisim noktaları gibi kırmızı ile işaretlenir. Bu aşamada çalışılan büyüklük için tüm hesaplama ve çizim yeniden yapılır. İstenirse katı cisim yapılan nokta aynı şekilde geri alınabilir.

Bu fonksiyonallite programın ilk geliştirme aşamasında sadece akım çizgisi için yapılmıştır, daha sonraki programın genelleştirme aşamasında bu fonksiyonun genelleştirilmesi tam olarak bitirilmemiştir, bu nedenle tekrar gözden geçirilmesi gerekmektedir.

4.5.10 Büyüklük kontur çizgilerinin tekrar çizdirilmesi

Analiz modundayken incelenen büyüklük kontur çizgilerinin sayıları, başlangıç ve bitiş değerleri yeniden belirlenerek çizdirilebilir. Bunun için klavyeden X tuşuna basılır. Açılan pencerede **Şekil 4.36**'da görüldüğü gibi büyüklüğün min, max ve adet

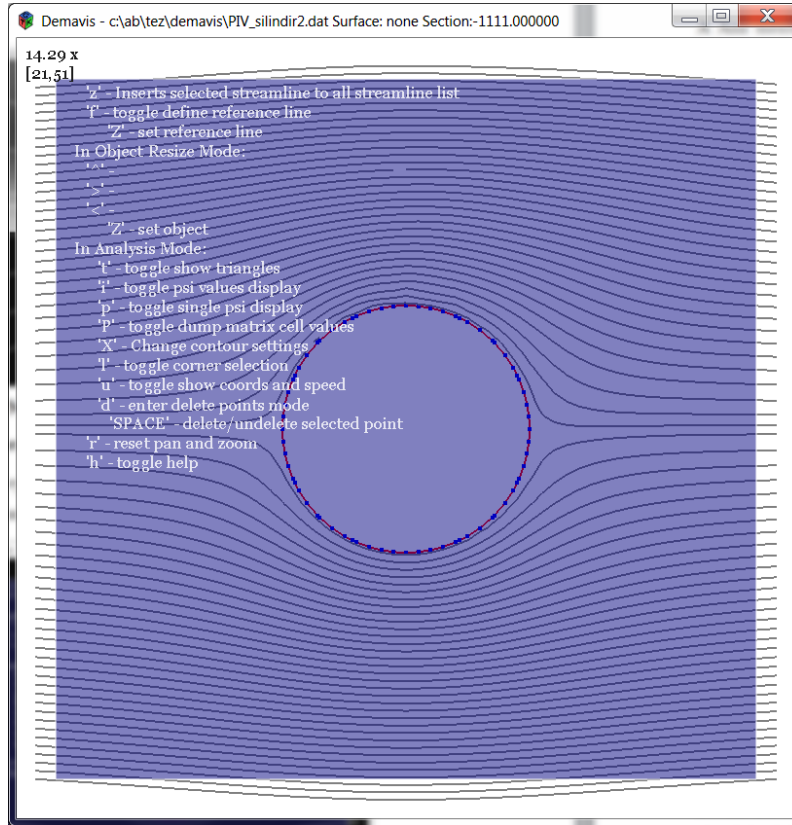
bilgileri görülmektedir. Bunlar istendiği şekilde değiştirilerek yeni çizimlerin yapılması sağlanır.



Şekil 4.36 : Büyüklük kontur çizim parametreleri.

4.5.11 Ekran tuşları yardım sayfası

İstendiği anda klavyeden h tuşuna basılarak Şekil 4.37’de görüldüğü gibi klavye tuşlarının kullanımına ilişkin açıklamalar görülebilir.



Şekil 4.37 : Tuş kısayol ekranı.

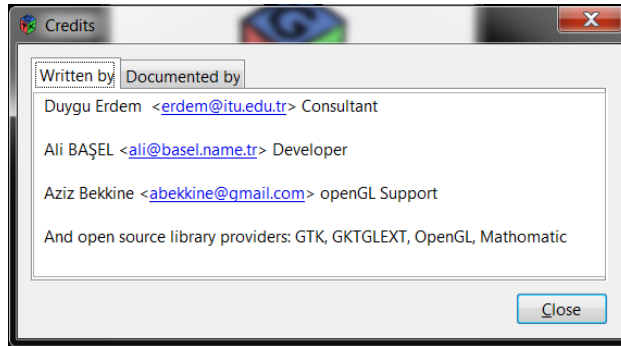
4.5.12 Hakkında penceresi

A Ana menüden “About” seçildiğinde **Şekil 4.38**'de görülen hakkında penceresi açılır, bu pencerede programın adı, versiyonu, İTÜ Trisonik Laboratuvarı internet sitesinin adresi görülmektedir.



Şekil 4.38 : Demavis hakkında penceresi.

“Credits” penceresinde **Şekil 4.39**'da görüldüğü gibi programı geliştiren ve katkıda bulunanlar görülmektedir.



Şekil 4.39 : Demavis geliştirici ekranı.

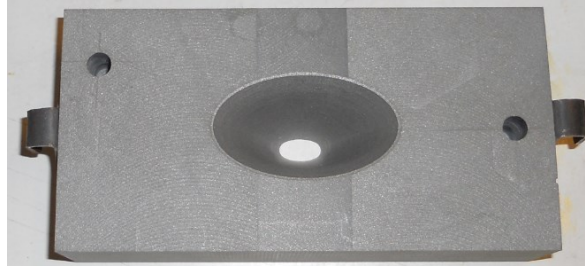
4.6 Deneysel Çalışma

Bölüm 2.1'de tanıtılan sıcak tel anemometresi kullanılarak **Şekil 4.41**'de görülen eliptik bir lüleden yapılan hava jetinin hız ve türbülans ölçüm deneyi yapılmıştır. Dantec StreamLine deney donanımları StreamWare yazılımı tarafından yönetilmektedir. **Şekil 4.40**'da ölçüm ve travers kontrol üniteleri görülmektedir.



Şekil 4.40 : Sıcak tel anemometre ölçüm ve traverse kontrol ünitesi.

Program donanım konfigürasyonunun yapılması, modüllerin kurulması (ısıtma oranının ayarlanması, kare dalga testi, filtreleme işlemi, prob sinyallerinin amplifikasyonu), prob kalibrasyonunun yapılması, travers kontrolü, verilerin elde edilmesi ve bunların işlenmesi işlemlerinin yapılmasını sağlamaktadır (Dantec Measurement, 2001).



Şekil 4.41 : Jet deneyinde kullanılan eliptik lüle.

Deneye öncelikle sıcak tel probunun kalibrasyonu yapılarak başlanmıştır. Kalibrasyon işlemi tamamlandıktan sonra deney düzeneği hazırlanmıştır. **Şekil 4.42'**de deneye başlamadan önce sıcak tel probunun kalibrasyonunun yapıldığı kalibrasyon ünitesi görülmektedir.



Şekil 4.42 : Sıcak tel probu kalibrasyon ünitesi.

Deney için bu lüle jet akış masasına **Şekil 4.43**'de görüldüğü şekilde bağlanmış ve hava jeti için hazırlanmıştır.



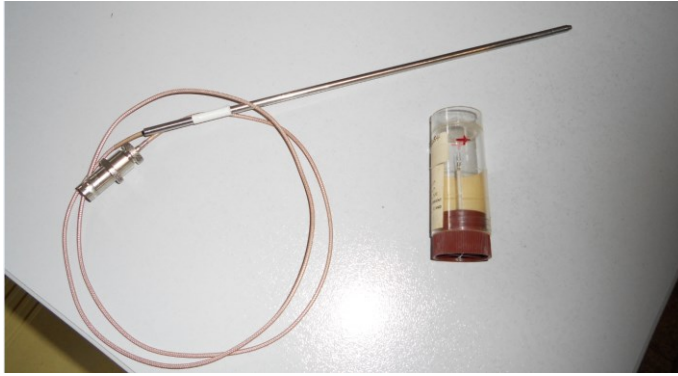
Şekil 4.43 : Deney düzeneğine alttan bakış.

Şekil 4.44'de deneyde kullanılan hava jet masası görülmektedir.



Şekil 4.44 : Deneyde kullanılan hava jet masası.

Şekil 4.45’de görülen sıcak tel probu Şekil 4.46’da görülen traverse bağlı desteğe bağlanarak deney düzeneği hazırlanmıştır.



Şekil 4.45 : Deneyde kullanılan sıcak tel sensörü.

Travers eksen tanımları yapılarak, travers prob başlangıç noktasına, taranacak ilk yüzeyin sol köşesine hareket ettirilmiştir. Ardından deney planı oluşturulmuş ve deney başlatılmıştır. Deney tamamlandıktan sonra veri işlenerek voltaj verisinin hız ve türbülans değerlerine dönüşümü sağlanmıştır.



Şekil 4.46 : Deneyde kullanılan travers sistemi.

Deney hacmi jetten 3 cm. uzakta başlayacak şekilde z ekseninde 0 ile 60 arasında 2 mm.'lik adımlarla, y ekseninde 0 ile 30mm. arasında 2 mm.'lik adımlarla, x ekseninde 0 ile 60 mm. arasında 6 mm'lik adımlarla taranacak şekilde travers programlanmış ve ölçümler belirtilen adımlarda yapılmıştır. Her bir noktada iki saniye boyunca veri alınmış, ölçümden sonra titreşim etkilerini azaltmak için de 1 sn'lik bekleme süresi konmuştur.

4.6.1 Deney verileri

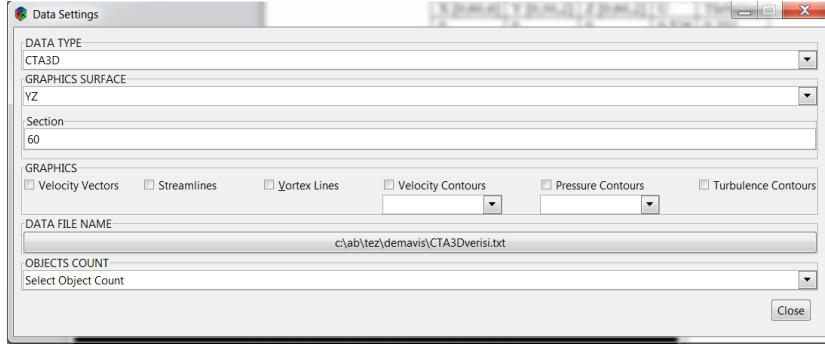
Çizelge 4.4'de deney verilerinin baş kısmından bir kesit görülmektedir. gösterilmiştir.

Çizelge 4.4 : Deney verilerinden bir kesit.

X [0;60,6]	Y [0;30,2]	Z [0;60,2]	U	Türbülans
0	0	0	0.526	0.202
0	0	2	0.556	0.205
0	0	4	0.629	0.251
0	0	6	0.620	0.220
0	0	8	0.589	0.233

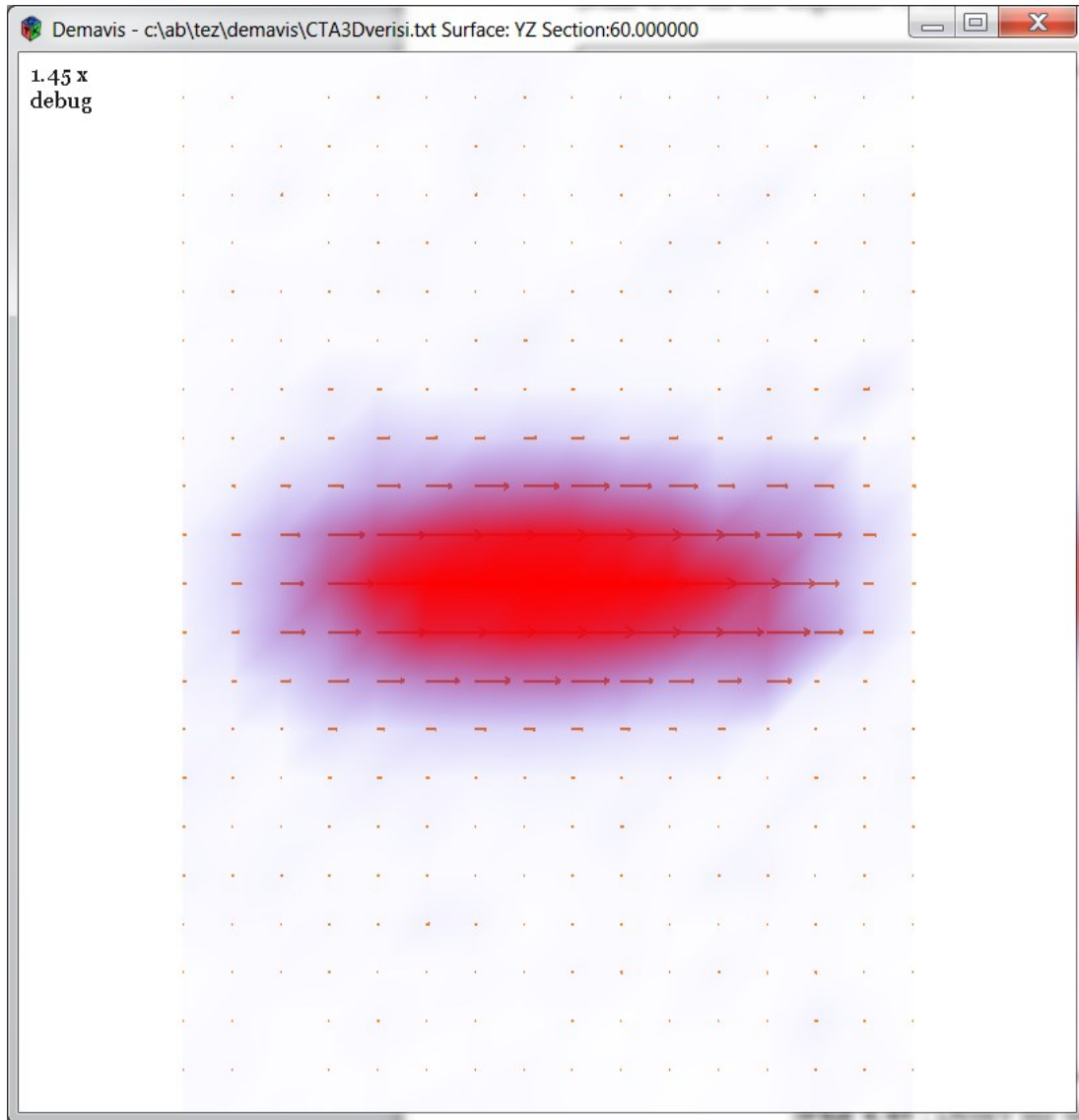
4.6.2 Deney verilerinin demavis ile çizilmesi

Deney verileri Demavis'e **Şekil 4.47'**te görüldüğü gibi hava jetinden en uzak düzlem olan X=60 düzlemindeki verileri işleyecek şekilde tanıtılmıştır.



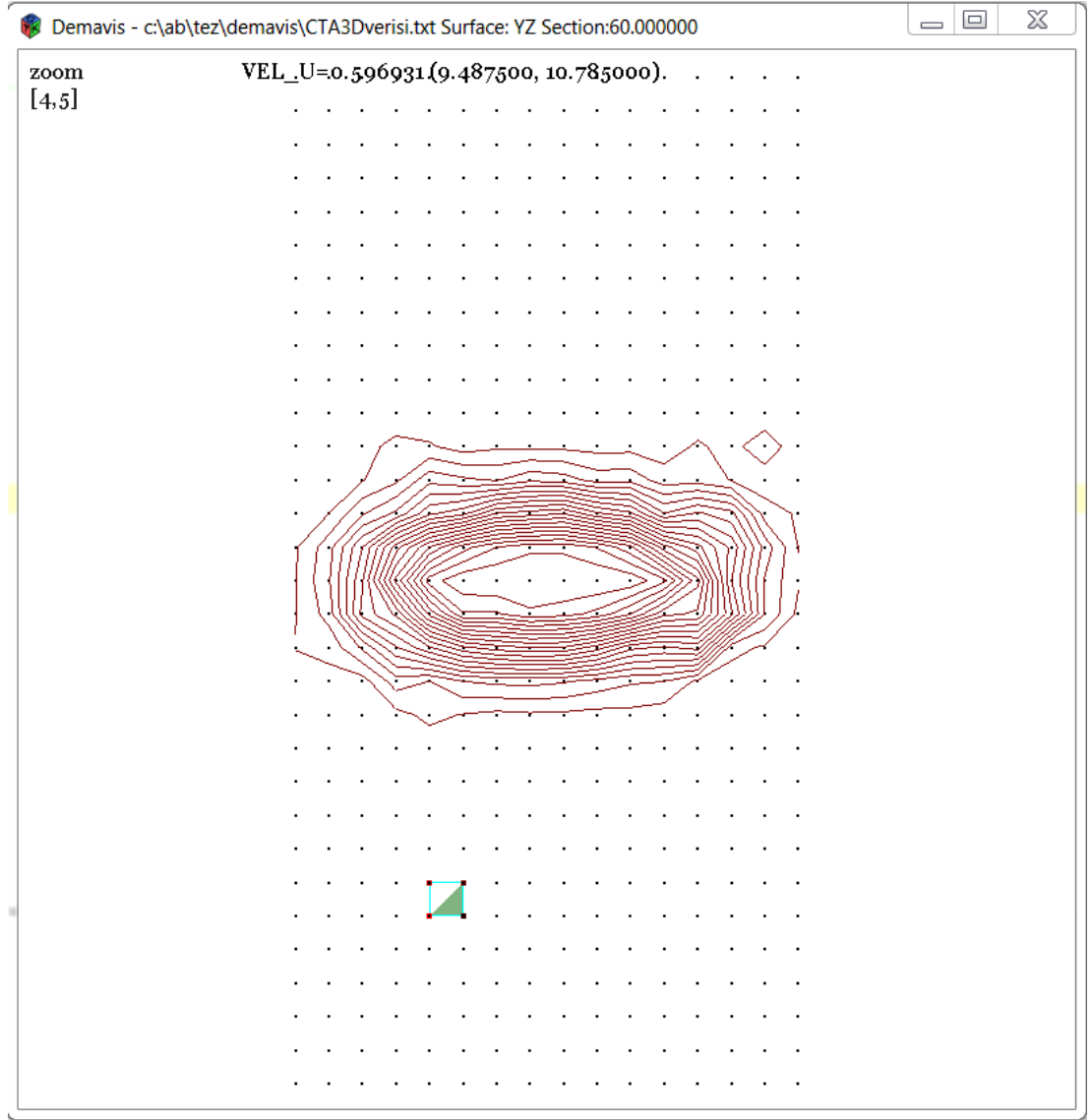
Şekil 4.47 : Deney verilerinin Demavis'e tanıtılması.

Şekil 4.48'de hız dağılımı ve hız alanları görülmektedir.



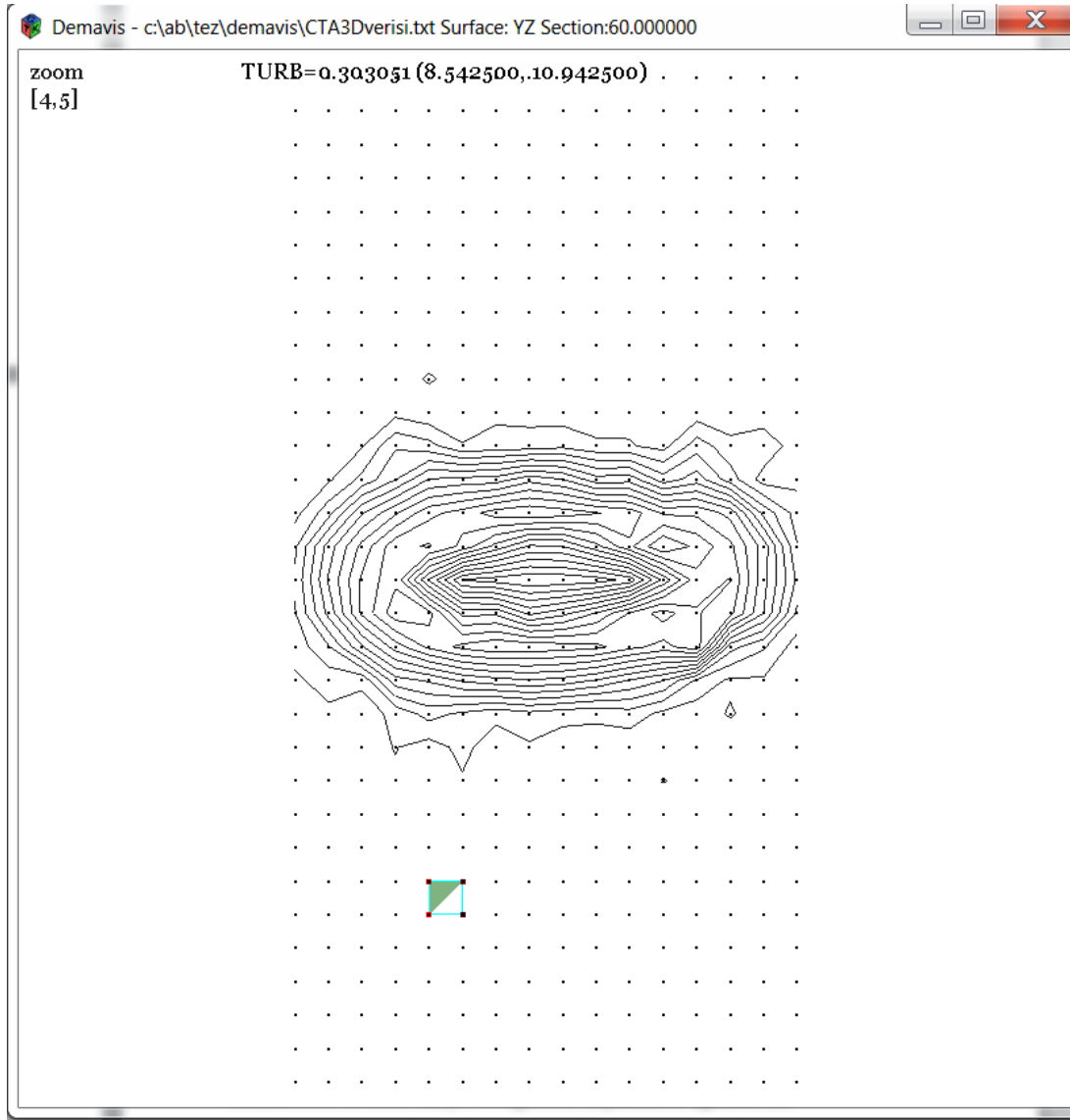
Şekil 4.48 : Deney hız dağılımı ve hız alanı renkleri.

Yine aynı düzlemdeki 20 adet hız konturu Şekil 4.49'da görülmektedir.



Şekil 4.49 : Deney hız konturları.

Türbülans dağılımı ve türbülans konturlarının birlikte görünümü **Şekil 4.50'**de görülmektedir.



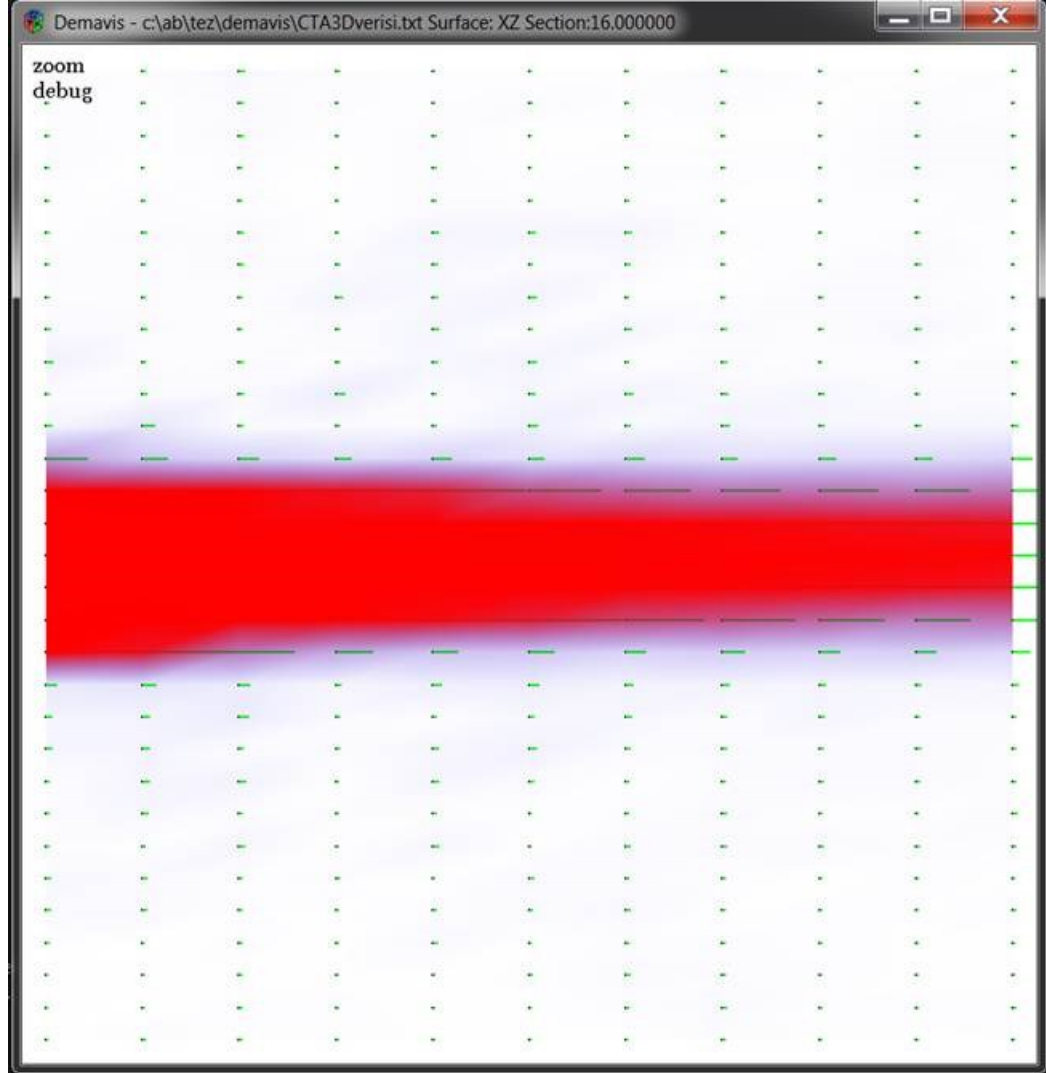
Şekil 4.50 : Deney türbülans konturları.

Türbülans değerleri Şekil 4.51’de Demavis info çıktısının kesintisinde görüldüğü gibi hız değerlerinden oldukça küçüktür, fakat programda hücre boyuna göre ölçekleme yapıldığı için hız dağılımına yakınlık göstermiştir. Bu ölçekleme özellikle büyük hızlar olduğunda görselliği kolaylaştırmaktadır.

```
TURB_min: 0.102000
TURB_max: 5.969000
TURB_size: 5.867000
TURB RangeCount: 10
TURB RangeStart: 0.400000
TURB RangeEnd: 5.969000
VEL_U_min: 0.317000
VEL_U_max: 30.899000
VEL_U_size: 30.582000
VEL_U RangeCount: 50
VEL_V_min: 0.000000
VEL_V_max: 0.000000
```

Şekil 4.51 : Deney info çıktısından kesit.

Bu düzleme dik ve akışa paralel $Y=16$ XZ düzlemindeki hız alanı ve hız büyüklükleri Şekil 4.52’de görülmektedir.



Şekil 4.52 : XZ düzleminde hız alanı ve hız büyüklükleri.

Demavis’in deney başından sona kadar yaptığı işlevleri gösteren program işlev bildirim kayıtları Ek A’da verilmiştir.

5. SONUÇ VE ÖNERİLER

Bu çalışma ile deney verilerinde incelenen akım ve girdaplılık çizgilerinin bilgisayar ortamında oluşturulması için bu büyüklüklerin analitik fonksiyonlarına uygun bir şekilde hesaplama algoritmaları geliştirilmiştir. Bu algoritmaların çizimlerde kullanılabilmesi için de çeşitli programlama algoritmaları geliştirilmiştir. Bunların yanısıra diğer büyüklüklerin de detaylı incelenebileceği işlevsel bir uygulama elde edilmiştir.

Demavis programının yapılan deneyleri görsel ve sayısal olarak incelemekte faydalı olacağı düşünülmektedir. İhtiyaç duyulan diğer fonksiyonaltelerin eklenmesi için de bir altyapı oluşturulmuştur.

Programın standart C ve açık kaynak kodlu kütüphanelerle geliştirilmiş olması hem programın çeşitli kitleler tarafından desteklenmesi ve geliştirilmesine olanak sunmakta hem de ekstra maliyet ve bağımlılık oluşmamasını sağlamaktadır.

5.1 Çalışmanın Uygulama Alanı

Demavis tanıtılan deney sistemleri verilerinin detaylı incelenmesinde konu üzerinde çalışan öğrenci ve araştırmacılar tarafından kullanılabilir. Benzer diğer sistemler için de uyarlamalar yapılabilir.

KAYNAKLAR

- Anderson, J. D. Jr.** (2007). Fundamentals of aerodynamics, 4th edition. Mc Graw-Hill Higher Education.
- Aeroprobe corporation.** (2003). Five and seven hole probe manual, version 2.0. Aeroprobe corporation, Virginia.
- Aeroprobe corporation.** (2010). Fast-response probe user's guide, edition 1.2. Aeroprobe corporation, Virginia.
- Bruun, H. H.** (1995). Hot-wire anemometry principles and signal analysis. Oxford university press, New York.
- Drazin, P. G., Riley, N.** (2006). The Navier-Stokes equations: a classification of flows and exact solutions, London Mathematical Society Lecture Note Series. 334 Cambridge University Press.
- Goldstein, Richard J.** (1996). Fluid Mechanics Measurements, Second Edition. Taylor & Francis, USA.
- Gümüş, Sibel** (2002). Sıcak tel probunun kalibrasyonu. Yüksek Lisans, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Makina Mühendisliği Ana Bilim Dalı, Kayseri.
- Perry, A. E.** (1982). Hot-wire anemometry. Oxford University Press, New York.
- Raffel, M., Willert, C. ve Kompenhans, L.** (1998). Particle image velocimetry: a practical guide. Springer-Verlag, Germany.
- William, S. Janna.** (1993). Introduction to fluid mechanics. PWS publishing company, Boston.
- Dairesel silindir etrafında potansiyel akış.** (t.y.). *Wikipedia*. Alındığı tarih: 15.02.2013, adres: http://en.wikipedia.org/wiki/Potential_flow_around_a_circular_cylinder
- Free Software Foundation** (t.y.). *GCC the GNU compiler collection*. Alındığı tarih: 01.03.2012, adres: <http://gcc.gnu.org>
- Gesslein, George II.** (t.y.). *Mathomatic*. Alındığı tarih: 01.03.2012, adres: <http://www.mathomatic.org/math>
- Gnome** (t.y.). *GTK+ Open GL Extension*. Alındığı tarih: 01.03.2012, adres: <http://www.mathomatic.org/math>
- GTK+ Team** (t.y.). *The GTK+ Project*. Alındığı tarih: 01.03.2012, adres: <http://www.gtk.org>
- Khronos Group** (t.y.). *OpenGL the industry's foundation for high performance graphics*. Alındığı tarih: 01.03.2012, adres: <http://www.opengl.org>
- Kurtuluş, Dilek Funda** (2010). Hot wire measurements, lecture note. Alındığı tarih: 15.02.2013, adres: <http://ocw.metu.edu.tr/course/view.php?id=66>.

MinGW.org (t.y.). *MinGW minimalist GNU for windows*. Alındığı tarih: 01.03.2012, adres: <http://www.mingw.org>

McRae, Graeme (t.y.). *Is point inside triangle*. Alındığı tarih: 01.03.2012, adres: <http://2000clicks.com/mathhelp/GeometryPointAndTriangle2.aspx>

Yükselen, M. Adil (t.y.). *Aerodinamik ders notları*. Alındığı tarih: 11.02.2013, adres: http://web.itu.edu.tr/~yukselen/Uck351/UCK351_Index.htm.

EKLER

EK A: Deney verisinin işlenmesi sırasında oluşan Demavis işlev bildirim kayıtları.

EK B: Demavis derleme dosyası MakefileWindows çıktısı.

EK C: Demavis program DVD'si.

EKA

Demavis main: started

gtkwin_first_time_init: Display toggle parameters initialization started
gtkwin_first_time_init: Display toggle parameters initialization finished.

gtkwin_main: started
gtkwin_main: init GTK
gtkwin_main: create Main Window
gtkwin_main: connect Main Window callback functions:
gtkwin_main: delete_event, destroy -> gtk_main_quit
gtkwin_main: connect key-press-event -> gtkwin_key_pressed
gtkwin_create_menu: started
gtkwin_create_menu: finished.

gtkwin_gl_initialize: started
gtkwin_gl_initialize: init OpenGL
OpenGL extension version - 6.1
gtkwin_gl_initialize: create a 800x800 GTK drawing_area
gtkwin_gl_initialize: set it OpenGL-capability
gtkwin_gl_initialize: connect callback functions for events:
gtkwin_gl_initialize: realize -> gtkwin_gl_realize
gtkwin_gl_initialize: expose_event -> gtkwin_display_render
gtkwin_gl_initialize: configure_event -> gtkwin_display_reshape
gtkwin_gl_initialize: connected callback functions for mouse
gtkwin_gl_initialize: finished.

gtkwin_gl_realize: (create) started
display_init_gl: started
display_init_gl: gl set clear color to black
display_init_gl: set gl shade model smooth
display_init_gl: set up orthogonal view between -1 and 1
display_init_gl: enabled gl blend
display_init_gl: finished.

gtkwin_gl_realize: finished.

gtkwin_wininfo_init: started
gtkwin_wininfo_init: All 'wininfo' structure variables assigned NULL.
gtkwin_wininfo_init: finished.

gtkwin_main: finished and will run GTK Main Loop.

gtkwin_DataSettingsWindow: started
gtkwin_DataSettingsWindow: Data Settings Window created
gtkwin_DataSettingsWindow: cancel button created
gtkwin_DataType_combobox: started
gtkwin_DataType_combobox: finished.

gtkwin_DataSettingsWindow: finished.

gtkwin_DataType_combobox_callback: started
gtkwin_DataType_combobox_callback: Data Type selected: CTA3D
gtkwin_GraphSurface: started
gtkwin_GraphSurface_destroy: started
gtkwin_GraphSurface_destroy: finished.

gtkwin_GraphSurface: finished.

gtkwin_GraphTypes: started
gtkwin_GraphTypes_destroy: started
gtkwin_GraphTypes_destroy: finished.

gtkwin_GraphTypes: finished.

gtkwin_DataFile_button: started
gtkwin_ObjectsTypes_combobox_destroy: started
gtkwin_ObjectsTypes_combobox_destroy: finished.

gtkwin_DataFile_button_destroy: started

```

gtkwin_DataFile_button_destroy: finished.

gtkwin_DataFile_button: finished.

gtkwin_DataType_combobox_callback: finished.

gtkwin_DataFile_button_callback: started
gtkwin_DataFile_button_callback: file select button clicked
gtkwin_DataFile_button_callback: File selected:c:\ab\tez\demavis\CTA3Dverisi.txt
file_read_data_compute_grid_show: started
file_init_grid_info: started
file_init_grid_info: all values initialized
file_init_grid_info: all lists roots assigned NULL
file_init_grid_info: finished.

file_read_data: started
Section=60.000000
file_read_CTA3D_data: started reading data
file:c:\ab\tez\demavis\CTA3Dverisi.txt
file_count_linkedlist_x: started
file_count_linkedlist_x: unique x number: 16
file_count_linkedlist_x: finished.

file_count_linkedlist_y: started
file_count_linkedlist_y: unique y number: 31
file_count_linkedlist_y: finished.

file_read_CTA3D_data: the_grid_linked_list_root filled with nodes
file_read_CTA3D_data: c:\ab\tez\demavis\CTA3Dverisi.txt, unique x count:16, y
count:31
file_read_CTA3D_data: finished.

file_read_data: finished.

file_compute_grid_info: Grid info computation started
Point count:496, nx=16, ny=31
file_compute_grid_info: Grid info computation finished.

file_initialize_matrix: Grid Matrix creation started
file_initialize_matrix: Grid Matrix creation finished.

display_init: Display parameters initialization started
gtkwin_init_toggles: Started
gtkwin_init_toggles: finished.

display_init: Display parameters initialization finished

display_fit_to_screen: started
display_fit_to_screen: finished.

file_read_data_compute_grid_show: finished.

gtkwin_ObjectsCount_combobox: started
gtkwin_ObjectsTypes_combobox_destroy: started
gtkwin_ObjectsTypes_combobox_destroy: finished.

gtkwin_ObjectsCount_combobox_callback: Started
Object count selected: 0
gtkwin_ObjectsTypes_combobox: started
gtkwin_ObjectsTypes_combobox_destroy: started
gtkwin_ObjectsTypes_combobox_destroy: finished.

gtkwin_ObjectsTypes_combobox: finished.

gtkwin_ObjectsCount_combobox_callback: finished.

gtkwin_ObjectsCount_combobox: finished.

gtkwin_ObjectsTypes_combobox: started
gtkwin_ObjectsTypes_combobox_destroy: started
gtkwin_ObjectsTypes_combobox_destroy: finished.

```

```

gtkwin_ObjectsTypes_combobox: finished.

gtkwin_DataFile_button_callback: finished.

gtkwin_DataSettings_window_accept: started
gtkwin_set_window_title: started for: c:\ab\tez\demavis\CTA3Dverisi.txt Surface:
YZ Section:60.000000
gtkwin_set_window_title: finished.

calculation_init_read_calculate: started
calculate_shape_points: started
calculate_shape_points: finished.

calculate_determine_points_inside_shape: started
calculate_determine_points_inside_shape: finished.

calculation_compute_grid_resultant_velocity_values: started
calculation_compute_grid_resultant_velocity_values: finished.

calculation_set_surface_points_value_zero: started for VEL
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of VEL started
file_compute_maxmin_value: Calculation of max min values finished
calculation_compute_grid_values: started for: PSI
calculation_compute_grid_values finished.

file_compute_maxmin_value: Calculation of max min values of PSI started
file_compute_maxmin_value: Calculation of max min values finished
calculation_compute_grid_vortex_values started
calculation_compute_grid_vortex_values finished.

calculation_set_surface_points_value_zero: started for VORTEX
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of VORTEX started
file_compute_maxmin_value: Calculation of max min values finished
calculation_set_surface_points_value_zero: started for VEL_U
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of VEL_U started
file_compute_maxmin_value: Calculation of max min values finished
calculation_set_surface_points_value_zero: started for VEL_V
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of VEL_V started
file_compute_maxmin_value: Calculation of max min values finished
calculation_set_surface_points_value_zero: started for TURB
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of TURB started
file_compute_maxmin_value: Calculation of max min values finished
calculation_set_surface_points_value_zero: started for PRES_TOTAL
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of PRES_TOTAL started
file_compute_maxmin_value: Calculation of max min values finished
calculation_set_surface_points_value_zero: started for PRES_STATIC
calculation_set_surface_points_value_zero: finished.

file_compute_maxmin_value: Calculation of max min values of PRES_STATIC started
file_compute_maxmin_value: Calculation of max min values finished

file_dump_grid_info: started

GridInfo:

gtkwin_dump_winfo: started

File Name: c:\ab\tez\demavis\CTA3Dverisi.txt
Data Type: CTA3D
Object Count: 0

```

GraphSurface_combobox_value: YZ
GraphSurface_Section_value: 60.000000
Velocity_checkbox_status: Unchecked
Streamlines_checkbox_status: Unchecked
Vortex_checkbox_status: Unchecked
EqualPressure_checkbox_status: Unchecked
EqualVelocity_checkbox_status: Unchecked
EqualVelocity_directions_combobox_value: none
EqualTurbulence_status: Unchecked

gtkwin_dump_wininfo: finished.

x_min: 0.000000
x_max: 30.000000
y_min: 0.000000
y_max: 60.000000
x_size: 30.000000
y_size: 60.000000
dx: 2.000000
dy: 2.000000
u_min: 0.317000
u_max: 30.899000
v_min: 0.000000
v_max: 0.000000
u_size: 30.582000
v_size: 0.000000
PSI_min: -0.000000
PSI_max: 237.043653
PSI_size: 237.043653
PSI_RangeCount: 50
VEL_min: 0.317000
VEL_max: 30.899000
VEL_size: 30.582000
VEL_RangeCount: 50
PRES_STATIC_min: 0.000000
PRES_STATIC_max: 0.000000
PRES_STATIC_size: 0.000000
PRES_STATIC_RangeCount: 50
PRES_TOTAL_min: 0.000000
PRES_TOTAL_max: 0.000000
PRES_TOTAL_size: 0.000000
PRES_TOTAL_RangeCount: 50
VORTEX_min: -6.668500
VORTEX_max: 6.777500
VORTEX_size: 13.446000
VORTEX_RangeCount: 50
TURB_min: 0.102000
TURB_max: 5.969000
TURB_size: 5.867000
TURB_RangeCount: 50
VEL_U_min: 0.317000
VEL_U_max: 30.899000
VEL_U_size: 30.582000
VEL_U_RangeCount: 50
VEL_V_min: 0.000000
VEL_V_max: 0.000000
VEL_V_size: 0.000000
VEL_V_RangeCount: 50
VEL_W_min: 10000000000.000000
VEL_W_max: -10000000000.000000
VEL_W_size: 0.000000
VEL_W_RangeCount: 50
_min: 10000000000.000000
_max: -10000000000.000000
_size: 0.000000
_RangeCount: 50
I: 16
J: 31

file_dump_grid_info: finished.

display_setup: function started
VNames:PSI

```

calculation_all_streamlines_reference_points: started for x=-1.000000 count:100
calculation_all_streamlines_reference_points: filled info-
>all_streamlines_ref_points_list_root
calculation_all_streamlines_reference_points: finished.

calculation_all_streamlines: started
calculation_all_streamlines: calculated 100 streamlines => info-
>all_streamlines_list_root[PSI]
calculation_all_streamlines: finished
calculation_appended_lines: started for PSI
calculation_appended_lines: appended 0 lines into info-
>all_streamlines_list_root[PSI]
calculation_appended_lines: finished.

calculation_range_values: started for VEL RangeCount:50
calculation_range_values: filled info->range_values_list_root[VEL]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for VEL
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[VEL]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for VORTEX RangeCount:50
calculation_range_values: filled info->range_values_list_root[VORTEX]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for VORTEX
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[VORTEX]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for TURB RangeCount:50
calculation_range_values: filled info->range_values_list_root[TURB]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for TURB
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[TURB]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for VEL_U RangeCount:50
calculation_range_values: filled info->range_values_list_root[VEL_U]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for VEL_U
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[VEL_U]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for VEL_V RangeCount:50
calculation_range_values: filled info->range_values_list_root[VEL_V]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for VEL_V
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[VEL_V]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for PRES_TOTAL RangeCount:50
calculation_range_values: filled info->range_values_list_root[PRES_TOTAL]
calculation_range_values: finished.

calculation_all_selected_value_lines: started for PRES_TOTAL
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[PRES_TOTAL]
calculation_all_selected_value_lines: finished.

calculation_range_values: started for PRES_STATIC RangeCount:50
calculation_range_values: filled info->range_values_list_root[PRES_STATIC]
calculation_range_values: finished.

```

```

calculation_all_selected_value_lines: started for PRES_STATIC
calculation_all_selected_value_lines: filled 51 lines info-
>all_streamlines_list_root[PRES_STATIC]
calculation_all_selected_value_lines: finished.

display_fit_to_screen: started
display_fit_to_screen: finished.

display_setup: function finished.

calculation_init_read_calculate: finished.

gtkwin_DataSettings_window_accept: finished.

main_quit: started
main_free_lists: started
file_cleanup_grid_linked_list_and_matrix: started
file_free_linkedlist_x: started
file_free_linkedlist_x: made free -info.x_list_root-
file_free_linkedlist_x: finished.

file_free_linkedlist_y: started
file_free_linkedlist_y: made free -info.y_list_root-
file_free_linkedlist_y: finished.

file_cleanup_matrix: Grid Matrix cleanup started
file_cleanup_matrix: Grid Matrix cleanup finished.

file_cleanup_grid_linked_list_and_matrix: made free --
info.the_grid_linked_list_root and matrix--
file_cleanup_grid_linked_list_and_matrix: finished.

calculation_cleanup: free all lists started
calculation_free_all_value_lines: started for PSI
calculation_free_all_value_lines: made free info->all_streamlines_list_root[PSI]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for PSI
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[PSI]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for VEL
calculation_free_all_value_lines: made free info->all_streamlines_list_root[VEL]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for VEL
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[VEL]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for PRES_STATIC
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[PRES_STATIC]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for PRES_STATIC
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[PRES_STATIC]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for PRES_TOTAL
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[PRES_TOTAL]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for PRES_TOTAL
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[PRES_TOTAL]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for VORTEX

```

```

calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[VORTEX]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for VORTEX
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[VORTEX]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for TURB
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[TURB]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for TURB
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[TURB]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for VEL_U
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[VEL_U]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for VEL_U
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[VEL_U]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for VEL_V
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[VEL_V]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for VEL_V
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[VEL_V]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for VEL_W
calculation_free_all_value_lines: made free info-
>all_streamlines_list_root[VEL_W]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for VEL_W
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[VEL_W]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_all_value_lines: started for
calculation_free_all_value_lines: made free info->all_streamlines_list_root[]
calculation_free_all_value_lines: finished.

calculation_free_inserted_streamlines_ref_point_list: started for
calculation_free_inserted_streamlines_ref_point_list: made free info-
>inserted_streamline_ref_points_list_root[]
calculation_free_inserted_streamlines_ref_point_list: finished.

calculation_free_streamlines_reference_points_list: started
calculation_free_streamlines_reference_points_list: made free info-
>all_streamlines_ref_points_list_root
calculation_free_streamlines_reference_points_list: finished.

calculation_cleanup: finished.

file_init_grid_info: started
file_init_grid_info: all values initialized
file_init_grid_info: all lists roots assigned NULL
file_init_grid_info: finished.

main_free_lists: finished.

```

main_quit: finished.

Demavis main: finished.

EK B

```
# 2012.02.29 Ali BAŞEL
#
# ITU Trisonik Lab demavis application
# Copyright 2013 TriSonik Lab., ITU
#
#
# ITU Trisonik Lab demavis make Makefile for MinGW 64
#
# http://ftp.gnome.org/pub/gnome/binaries/win64/gtk+/2.22/gtk+-bundle_2.22.1-
20101229_win64.zip
#
# PKG_CONFIG_PATH must be defined for compiling
# export PKG_CONFIG_PATH="c:/ab/gtkglext/lib/pkgconfig"

# windows'un PATH değişkenine aşağıdaki path'lerin eklenmesi gerekiyor:
# ...;C:\ab\mingw64\bin;c:\ab\Gtk64\bin

MATHOMATIC_WIN_CCFLAGS=-I"C:\ab\mathomatic-15.8.2" -I"C:\ab\mathomatic-
15.8.2\lib"
MATHOMATIC_WIN_LIB=libmathomatic.so

CFLAGS=-Wall -Wextra -Werror -Wfatal-errors -I. -I/usr/include/GL

GTK_CCFLAGS=$(CFLAGS) `pkg-config --cflags gtkglext-1.0`
GTK_LIB=`pkg-config --libs gtkglext-1.0`

all: demavis

CCFLAGS= $(GTK_CCFLAGS) $(MATHOMATIC_WIN_CCFLAGS)
LIBFLAGS= $(GTK_LIB) $(MATHOMATIC_WIN_LIB)

demavis: main.o display.o file.o calculation.o gtkwin.o equation.o
    gcc -o demavis main.o display.o file.o calculation.o gtkwin.o equation.o
$(LIBFLAGS)

main.o: main.c main.h main_interface.h demavis.h
    gcc $(CCFLAGS) -c main.c

display.o: display.c *.h
    gcc $(CCFLAGS) -c display.c

file.o: file.c file.h *.h
    gcc $(CCFLAGS) -c file.c

calculation.o: calculation.c calculation.h *.h
    gcc $(CCFLAGS) -c calculation.c

gtkwin.o: gtkwin.c gtkwin.h *.h
    gcc $(CCFLAGS) -c gtkwin.c

equation.o: equation.c equation.h *.h
    gcc $(CCFLAGS) -c equation.c

clean:
    rm demavis.exe *.o
```


ÖZGEÇMİŞ



Ad Soyad: Ali BAŞEL

Doğum Yeri ve Tarihi: Sivas 1969

E-Posta: ali@basel.name.tr; abasel@ford.com.tr; basela@itu.edu.tr

Lisans: İTÜ Uçak Mühendisliği