Kartezyen Hesaplama Ağları için Üç Boyutlu Euler Çözücüsü Geliştirilmesi

Proje No: 106M471

Prof. Dr. M. Halûk AKSEL
Y. Doç. Dr. Cüneyt SERT
Makina Yük. Müh. Mehtap ÇAKMAK
Makina Yük. Müh. Bercan SİYAHHAN

EYLÜL 2009
ANKARA

ÖNSÖZ

Bu proje çerçevesinde Kartezyen hesaplama ağları için zamana bağlı olmayan üç boyutlu bir Euler çözücüsü geliştirilmiştir. Bu çalışmadaki sonuçlar Orta Doğu Teknik Üniversitesi, Makina Mühendisliği Bölümünde, Prof. Dr. M. Haluk Aksel'in tez yürütücülüğünde ve Y. Doç. Dr. Cüneyt SERT'in yardımcı tez yürütücülüğünde Makina Yüksek Mühendisi Mehtap Çakmak tarafından gerçekleştirilen "Development of a Multıgrıd Accelerated Euler Solver on Adaptıvely Refıned Two- and Three-Dımensıonal Cartesıan Grıds" ve Makina Yüksek Mühendisi Bercan Siyahhan tarafından gerçekleştirilen "A Two-Dimensional Euler Flow Solver on Adaptıve Cartesıan Grıds" isimli Yüksek Lisans tezleri çerçevesinde elde edilmiştir. Bu proje Türkiye Bilimsel ve Teknik Araştırma Kurumu, Mühendislik Araştırma Grubu tarafından desteklenmiştir.

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

ÖZET

Bu proje çerçevesinde Kartezyen hesaplama ağları için zamana bağlı olmayan üç boyutlu bir Euler çözücüsü geliştirilmiştir. Bu yöntemle geometrik karmaşıklıklarla ilgili zorluklar ile yapısal ve yapısal olmayan hesaplama ağlarında karşılaşılan akışa ve geometriye yönelik adaptasyon problemleri ortadan kaldırılmıştır. Buna ek olarak, geliştirilmiş olan yazılım tam otomatik olup, ağ üretimi ile çözüm aşamaları arasında kullanıcı müdahalesi gerektirmemektedir.

Diziler gibi klasik veri yapıları yerine dörtlü ağaç (quadtree) ve sekizli ağaç (octree) ve bağlanmış liste (linked list) gibi dinamik veri yapıları kullanılmıştır. Bağlantı bilgileri dörtlü ağaç ve sekizli ağaç veri yapısının hücreler arası ebebeyin-çocuk ilişkisi ile elde edilmiştir. Bu tip bir veri yapısıyla, yapısal ve yapısal olmayan hesaplama ağlarına göre daha karmaşık geometrilerin ele alınması mümkün olmuştur.

Euler denklemlerinin sonlu hacim formülasyonu hücre merkezli yaklaşımla kullanılmıştır. Hücre yüzlerindeki akılar akı fark ayrıştırması ve akı vektör ayrıştırması yöntemleriyle hesaplanmıştır. Uzayda ikinci dereceden hassasiyet elde edilebilmesi için basit değişkenlerin yeniden oluşturulmasında yol tümleme (path integration) ve asgari kareler (least squares) yöntemleri kullanılmıştır.

Yakınsamanın hızlandırılabilmesi için yerel zaman adımlarıyla birlikte çok kademeli (multistage) zaman adımlaması kullanılmıştır. Kartezyen hesaplama ağlarının adaptasyona çok uygun olmasından dolayı, yapılan adaptasyon sonucunda hücre boyları arasında önemli farklılıklar oluştuğu için yerel zaman adımı uygulaması yakınsama hızını arttırmıştır.

Çözüme bağlı hesaplama ağı adaptasyonu çözüm ile ağ arasındaki uyumun oluşmasını sağlayarak akıştaki kritik bölgelerin daha iyi çözümlenmesine neden olmuştur. Böylelikle, çözüm zamanında önemli bir artış olmadan yüksek seviyede hassasiyet elde edilmesi mümkün olmuştur. Çözüm adaptasyonu kayma tabakalarında hız dönümü, normal ve oblik şoklarda ise hız gradyanı kullanılarak gerçekleştirilmiştir. Bu iki kriterin bir arada kullanılması bir tanesinin kullanılmasına göre daha iyi sonuçlar vermiştir.

Geliştirilen yazılım literatürde mevcut deneysel sonuçlarla karşılaştırılarak doğrulanmıştır.

Anahtar Sözcükler Kartezyen Hesaplama, Euler Akış Çözücüsü, Geometrik Adaptasyon, Çözüm Adaptasyonu, Çoklu Ağ Yöntemi

ABSTRACT

A Cartesian method for the solution of the steady-state three-dimensional Euler equations is developed for Cartesian grids. This method is used to overcome the difficulties associated with geometric complexities and adaptation problems encountered in structured and ordinary unstructured methods. In addition, the developed code is fully automatic to eliminate the user interference between the mesh generation and solution steps.

Instead of simple conventional data structures like two-dimensional arrays, dynamic data structures like quadtree, octree and linked list are used. Connectivity information is obtained from the quadtree or octree data structure via parent-children relationships between the cells. This kind of data structure enables to handle much more complicated input geometries compared to structured and ordinary unstructured methods.

The finite volume formulation of the three-dimensional Euler equations is used with cell-centered approach. Flux difference splitting and flux vector splitting methods is employed for formulation of the flux at cell faces. Primitive variables are reconstructed using the path integral and least squares methods to achieve second order accuracy in space. In order to ensure accurate and bounded values, limiters are employed in the reconstruction process.

Multistage time stepping is used with local time steps to increase the convergence rate. Since Cartesian meshes are highly adaptive, there are significant differences between the length scales of the cells. Hence, the use of local time steps improves the convergence rate.

Multigrid convergence acceleration technique, specifically nested iteration, is used in order to increase the convergence rate to the steady-state even more. The problem under consideration is first solved on a coarse mesh. This solution is then used for successively finer meshes as the improved initial guess.

Solution adaptation is used for resolving more critical regions in the solution domain. As a result, higher levels of accuracy are obtained without a significant increase in the computational time. Curl of velocity is used for resolving shear layers and divergence of velocity is used for resolving oblique and normal shock waves. The combination of these two criteria gives better results than a single one.

The developed code is then verified with the experimental data available in the literature.

Keywords: Cartesian Grids, Euler Flow Solver, Geometric Adaptation, Solution Adaptation, Multigrid Method

BÖLÜM 1
GİRİŞ

1.1 GENEL

Hesaplamalı akışkanlar dinamiğinde kullanılan yapısal ve yapısal olmayan hesaplama ağları gerçek mühendislik problemlerin çözümünde başarıyla uygulanmıştır. Her iki yöntem de karmaşık geometriler etrafındaki hesaplama bölgesinin ayrıştırılmasında tam otomasyon sağlayamamıştır. Bunun en önemli nedenlerinden biri, her iki yöntemde de katı cisimlerin yakınındaki hücrelerin yüzeye uyumunun gerekmesidir. Bu durumda hesaplama ağı katı cismin geometrisine ve topolojisine bağlanmakta ve yüzeydeki hesaplama ağı yerel geometri ile akışın çelişkili gereksinimine maruz kalmaktadır. Yapısal olmayan yüzey ağlarındaki üçgenlemeler bu gereksinimi rahatlatmasına karşılık, yapısal yüzey ağlarında önceden belirlenmiş bağlantılar ek kısıtlamalar ortaya çıkartmaktadır. Yapısal olmayan ağ teknolojisi, geometrik esnekliği nedeniyle popülerlik kazanarak özellikle viskoz olmayan akışların modellenmesinde kullanılmaktadır. Bu yöntemin (i) belirli bir görünüş oranına (aspect ratio) sahip yüksek kaliteli dört yüzlü hesaplama ağlarının yaratılamaması ve (ii) yüksek derecede eğilmiş dört yüzlü hesaplama ağlarında akış çözücüsünün doğruluk ve güvenilirliğinin azalması olmak üzere iki önemli dezavantajı bulunmaktadır. Bu nedenle, karmaşık geometrileri ele alabilmek için alternatif teknolojiler geliştirilmiştir. Bu yöntemlerin en göze çarpanlardan biri Kartezyen ağ yöntemdir. Bu yöntemin (i) otomatik ağ üretimi, (ii) otomatik ağ uyarlaması, (iii) basit akı hesaplama yöntemi ve basitleştirilmiş veri yapısı olmak üzere çok önemli üç avantajı bulunmaktadır.

1.2. LİTERATÜR ARAŞTIRMASI

Kartezyen hesaplama ağları ise yüzeye uyum gerektirmediğinden diğer hesaplama ağlarına göre farklıdır. Bu tip hesaplama ağları dörtgen prizma hücrelerden oluşmakta ve ağ çizgileri birbirlerine dik olarak katı cisim içerisinde doğru uzanmaktadır. Daha sonra bu yöntemde, tümüyle katı cisim içerisinde olan hücreler ile katı cisimle kesişen hücreler belirlenmektedir. Bunların dışında kalan hücreler ise akışın olduğu hücrelerdir. Kartezyen yaklaşımında yüzey geometrisi ile dörtgen prizmaların kesişmelerinin hesaplanmasından kaynaklanan problemler ile yüzeye uygun hesaplama ağı oluşturulması arasında bir

tercih yapılmaktadır. Bu kesiştirme yöntemleri De Zeeuw ve Powell (1991), Karman (1995), Melton, Berger, Afdosmis ve Wong (1995), Melton (1996), Quirk (1992) ile Werterlen ve Karman (1995) gibi araştırmacılar tarafından karmaşık geometrilere uygulanmıştır.

Kartezyen yöntemi ile yüzey modellemede bu yaklaşım çok önemli bir sonuç doğurmaktadır. Hesaplama ağındaki hücrelerin geometriyi gelişi güzel kesmesinden dolayı, yüzeydeki kesilmiş hücreler yüzey tanımından bağımsız hale gelmektedir. Böylece, yüzeyin tanımlanması yüzeye uyumlu yaklaşımlarda olduğu gibi hem akışı hem de yerel geometriyi çözümlemeyi gerektirmemektedir.

Kartezyen yaklaşımlar iki genel kategoriye girmektedirler. İlk kategoride h-rafine dörtgen prizmalardan oluşan yapısal olmayan veya sekizli ağaç sistemine dayanan yapısal hesaplama ağları yer almaktadır. Diğer kategori ise De Zeeuw ve Powell (1991) ve Quirk (1992) tarafından araştırılmış olan ve yapısal ağ blokları içine yerleştirilmiş yapısal alt ağlardan oluşmaktadır. Yapısal olmayan ve sekizli ağaç sistemine dayanan yöntemlerde hesaplama bölgesinde ağ oluşturulması hücre bölünmesine dayanmaktadır. Bu yaklaşımda ya kaba bir hesaplama ağından ya da tek bir kök hücreden başlanılarak dörtgen prizma elemanlar art arda bölünerek akışın ve geometrinin gerektirdiği hesaplama ağı oluşturulmaktadır. Sonuçta elde edilen hesaplama ağı ya tümüyle yapısal olmamakta ya da sekizli ağaç yapısında olmaktadır. De Zeeuw ve Powell (1991), Quirk (1992), Berger ve Melton (1995) ile Melton (1996) bu yaklaşımı iki ve üç boyutlu problemlere başarıyla uygulamıştır.

Günümüzde Kartezyen hesaplama ağı yaklaşımının üç boyutlu karmaşık geometrilere uygulanması yaygınlaşmışsa da bu yaklaşım 1970'li yılların sonlarından itibaren uygulanmaktadır. Purvis ve Burhalter (1979) iki boyutlu tam potansiyel denklemi Kartezyen hesaplama ağı kullanarak çözmüştür. Euler denklemlerinin çözümü 1980'li yılların ortalarından itibaren Clark, Salas ve Hassan (1986) ile Grossman ve Whitaker (1986) gibi araştırmacılar tarafından incelenmiştir. İlk üç boyutlu sürtünmesiz çözümler 1980 sonlarında Gaffney, Hassan ve Salas (1987) tarafından elde edilmiştir.

Kartezyen yaklaşımı endüstriyel problemlere başarıyla uygulanmıştır. Bunlar arasında Boeing firması tarafından kullanılan ve tam potansiyel denklemi çözen TRANAIR yazılımı ile Euler simülasyonu yapan Tidd, Strash, Epstein, Luntz, Nachson ve Rubin (1991) tarafından geliştirilen MGAERO yazılımı bulunmaktadır. Lednicer, Tidd ve Birch (1994), Levy, Warner and Nelson (1994), Melton, Enamoto ve Berger (1993) ve Aftosmis, Melton ve Berger (1995) ise karmaşık geometrilere bileşke bazlı bir yaklaşımı kullanmaktadır.

Kartezyen ayrıştırmalar sonucunda ortaya çıkan kesilmiş hücreler sınır şartlarının uygulanmasında çeşitli problemler oluşturmaktadır. Bu problemler Berger ve LeVeque (1990), Berger ve Melton (1995), Coirier ve Powell (1993), Forrer (1996) ile Melton (1996) tarafından tartışılmıştır.

Coirier (1994) tarafından belirtildiği gibi Kartezyen yaklaşımda dörtgen prizmaların *h*-tipi ayrıştırması sonucunda ortaya çıkan isotropik hücreler sürtünmesiz akışları modellemede çok başarılı olmasına karşılık sınır tabaka içerisinde ve diğer sürtünmeli akışlarda etkili olamamaktadır. Coirier (1994)

ile Karman (1995) sürtünmesiz Kartezyen yaklaşımın sürtünmeli akışlara da uygulanabilmesi için çalışmalar yapmışlardır.

Kartezyen yöntemler ilk olarak 1975 yılında yapısal ve yapısal olmayan çözüm yöntemlerine alternatif olarak önerilmiştir. Bu yöntemlerdeki en büyük amaç otomatik hesaplama ağı üretimini sağlamak ve çözüm adaptasyonunu kolaylaştırmaktır. Ancak, bilgisayarların yetersizliği nedeniyle, bu yöntemlerle 1980'li yıllara kadar çok fazla ilgilenilmemiştir. Bunun en önemli sebebi ise kartezyen yöntemlerin karmaşık bir veri yapısına gerek duymasıdır.

Clarke, Salas ve Hassan (1986), Kartezyen yöntemleri çoklu kanat profilleri üzerindeki iki boyutlu zamana bağlı olmayan sürtünmesiz akışları çözmek için kullanmışlardır. Aynı problemin yapısal yöntemlerle çözümü ise örtüşmeli hesaplamalı ağları gibi oldukça karmaşık yöntemler gerektirmektedir. Daha sonra, Mitchel, Salas ve Hassan (1988) iki boyutlu Euler denklemlerinin çözümü için alternatif bir Kartezyen hesaplama ağı yaratma yöntemi geliştirmişlerdir.

Tidd, Strash, Epstein, Luntz, Nachshon ve Rubin (1992), Kartezyen yaklaşımı tüm bir uçak etrafındaki üç boyutlu zamana bağlı olmayan sürtünmesiz akışı çözmek için kullanmışlardır.Üç boyutlu problemlerin çözümü için gerekli olan çoklu ağ yöntemini yakınsamayı hızlandırmak için kullanmışlardır. Epstein, Luntz ve Nachshon (1992), benzer yöntemleri genel uçak geometrileri etrafındaki sürtünmesiz akışlara uygulamışlardır.

Kartezyen yöntemler yapısal olmayan bir yaklaşıma dayandığı için genellikle sonlu hacim yöntemi ile birlikte kullanılmaktadır. Ancak, Morinishi (1992) iki boyutlu sıkıştırılabilen Euler denklemlerinin Kartezyen hesaplama ağları üzerindeki çözümü için sonlu fark yöntemini kullanmıştır. Zaman boyutundaki tümleme ise Runge-Kutta şeması ile gerçekleştirilmiştir.

De Zeeuw (1993), zamana bağlı olmayan sürtünmesiz iç ve dış akış problemlerinde iki boyutlu Euler denklemlerinin çözümü için Kartezyen yaklaşımını dörtlü ağaç ve bağlı liste veri yapıları ile birlikte kullanmıştır. Testere dişi çevrimi adı verilen bir çoklu ağ yöntemini başarıyla uygulayarak, kullanılan veri yapısından dolayı Kartezyen yöntemlerin çoklu ağ uygulamalarına çok uygun olduğunu göstermiştir. Böylelikle hafıza gereksinimini önemli bir ölçüde arttırmadan hesaplama zamanını yarı yarıya azaltmıştır. Buna ek olarak, küçük kesik hücrelerle ilgili güçlükleri özel bir yerel zaman adımı uygulaması ve uzayda ikinci dereceden hassasiyet ile ortadan kaldırmıştır.

Coirier (1994), iki boyutlu Euler ve Navier-Stokes denklemlerinin hibrid bir hesaplama ağı üzerindeki çözümü için yazılım geliştirilmiştir. Bu çalışmada, sınır tabakayı etkin bir şekilde çözebilmek için hibrid bir hesaplama ağı kullanılmıştır. Ancak, bu çalışma, Kartezyen yöntemlerde katı cisme tam bir uyum gerekmediği için, bu yöntemin tam bir uygulaması değildir

Kartezyen yöntemler modern yaklaşımlar olup, bu yöntemlerde kullanılan çözüm teknikleri genelde daha klasik yaklaşımlar olan yapısal yöntemler için geliştirmişlerdir. Quirk (1992) çeşitli çözüm tekniklerinin Kartezyen yöntemlere uygulanabilirliğini araştırarak bir takım iyileştirmeler önermiştir.

Aftosmis (1995), üç boyutlu Kartezyen hesaplama ağı üretimi için iç-dış testi (inside-outside test) ve çokgen klipsleme (polygon clipping) gibi alternatif yöntemler geliştirmiştir. Daha sonra, Afdosmis (1997) Kartezyen yöntemleri bileşke bazlı geometrileri kullanarak üç boyutlu geometrilere uygulamıştır. Böylelikle, kirli yüzeylerle ilgili güçlükler ilk defa ortadan kaldırılmıştır. Bu durumda, geometri girdisi grafik ortamda yaratıldıktan sonra Kartezyen yöntemlerle analiz için aktarılmakta ve daha karmaşık geometriler için akış analizi mümkün olmaktadır. Bu işlemden önce herhangi bir yöntemin verimliliğini azaltan kirli yüzeyler manuel olarak temizlenmektedir.

Coirier ve Powell (1995), zamana bağlı olmayan transonik sürtünmesiz akışlarda Kartezyen yaklaşımının hassasiyetini araştırmıştır. Düzgün ve adaptif olarak hassaslaştırılan Kartezyen hesaplama ağları ile elde edilen sonuçları yapısal yöntemlerle elde edilmiş sonuçlarla karşılaştırmıştır.

Pember, Bell, Colella, Crutchfield ve Welcome (1995), Kartezyen yaklaşımın karmaşık geometriler etrafındaki zamana bağlı sıkıştırılabilir akışların çözümüne özellikle uygun olduğunu göstermiştir. Buna ek olarak, Kartezyen yöntemlerin zamana bağlı olmayan sürtünmesiz akışların çözümünde de çok verimli olduğunu göstermiştir.

Kartezyen yöntemlerin en önemli dezavantajlarından birisi bağlantılılığın (connectivity) belirlenmesinin kullanılan karmaşık veri yapısından dolayı çok zaman almasıdır. Khokhlov (1998), bu güçlüğü özel bir algoritma kullanarak ortadan kaldırmıştır. Özel hücreleri bir araya getirerek bazı kutular oluşturmuş ve bunların arasındaki bağlantıyı ekstra işaretleyiciler kullanarak sağlamıştır. Böylelikle, bağlantılılık bilgisi için ağacın taranmasındaki gerekli zaman aralığı bellek gereksinimi çok fazla arttırılmadan önemli ölçüde azaltılmıştır.

Kartezyen yöntemler genellikle uzayda birinci veya ikinci dereceden hassastır. Forer ve Jeltsch (1998), iki boyutlu problemlerin çözümünde uzayda daha yüksek dereceden hassasiyet elde edebilmek için özel bir teknik geliştirmiştir. Buna ek olarak, kesik hücreleri de özel bir şekilde ele almıştır. Böylelikle, Kartezyen yöntemlerde kalıtımsal olarak bulunan kararsızlık problemi önlenmiştir.

Wu ve Li (2003), daha önce sınır tabakada kalıtımsal olarak bulunan isotropik olmayan yapıyı ele almak üzere geliştirilmiş bir hassaslaştırma yöntemini sürtünmesiz akış problemlerine uygulamıştır. Böylelikle iki boyutlu sürtünmesiz akış problemlerinde eğik ve normal şok dalgalarını yakalamak mümkün olmuştur.

Qian, Causon, Ingram ve Mingham (2003), Kartezyen yaklaşımını iki akışkanlı hidrolik akış problemlerinin çözümünde kullanmıştır. Bu çalışmada, sıkıştırılamaz akışlar için geçerli Euler denklemleri

yapay sıkıştırılabilirlik faktörü kullanılarak sıkıştırılabilir akışları modellemek için kullanılmıştır. Çözüm alanı su ve havayı kapsamakta olup, iki akışkan arasındaki arayüz temas süreksizliği olarak tanımlanmıştır.

Hunt (2004), Afdosmis (1995) tarafından önerilen veri yapısını ve yöntemleri üç boyutlu Euler denklemlerinin çözümü için kullanmıştır. Geliştirilen yazılım zamana bağlı olan ve olmayan problemler için geçerlidir. Hareketli sınırlarda hücre birleştirmesi adı verilen bir yöntem kullanarak, zamana bağlı problemlerde yerel zaman adımı uygulaması yapılamamasından dolayı kesik hücrelerle ilgili ortaya çıkan güçlükleri ortadan kaldırmıştır. Buna ek olarak, Kartezyen yöntemlerde paralel hesaplamaları kullanarak üç boyutlu karmaşık problemleri kabul edilebilir zaman aralıkları içinde çözmüştür.

Li ve Wu (2004), izotropik olan ve olmayan hassaslaştırma yöntemlerini kuvvetli şok dalgalarının bulunduğu iki boyutlu sürtünmesiz akış problemlerinde kullanmıştır. Çözüm adaptasyonu kullanarak akış alanının kritik bölgelerini incelemek için Kartezyen yöntemlerin etkinliğini göstermiştir.

French (2004), korunabilir hücre köşeli Euler çözücüsünü Kartezyen hesaplama ağlarında kullanmıştır. Daha önceki çalışmalarda çok kademeli zaman adımı kullanılırken, bu çalışmada Lax-Wendroff zaman adımlaması kullanılmıştır. Modern bir yaklaşım olan Kartezyen yöntemlerin birçok klasik çözüm yöntemlerine uygun olduğunu göstermiştir.

Dodone ve Grossman (2004), kesik hücreleri daha etkin olarak ele alabilmek için eğrilikle düzeltilmiş simetri tekniğini (curvature-corrected symmetry technique) geliştirmiştir. Kesik hücrelerin ele alınmasının Kartezyen yöntemlerin en kritik bölümü olup, daha etkin yöntemler gerektirmektedir.

Keats ve Lien (2004), daha önce sıkıştırılamayan laminer akışlar için geliştirilmiş izotropik olmayan hassaslaştırmayı iki boyutlu zamana bağlı olmayan Euler denklemlerine uygulamıştır. Böylelikle fazla sayıda hesaplama hücrelerinin oluşması önlenerek, hesaplamalar için önemli kazanımlar elde etmiş ve izotropik olmayan hassaslaştırmanın öneminin zamana bağlı akışlarda zamana bağlı olmayan akışlara göre çok daha önemli olduğunu göstermiştir.

BÖLÜM 2
İKİ BOYUTLU KARTEZYEN AĞ ÜRETİCİSİNİN HAZIRLANMASI

2.1 DÖRTLÜ AGAÇ VERİ YAPISI

Kartezyen ağlar, düzenli olmayan hesaplama ağlarının özel bir türüdür ve düzenli olmayan ağlar için ağ noktaları ve komşular arasında sıralama bilgisi düzenli ağlarda olduğu gibi doğrudan bilinmemektedir. Diğer bir deyişle, akı hesaplamaları, çoklu ağ yöntemi ve yeniden yapılandırma yöntemleri için komşuluk ilişkilerinin bilinmesi gerekmektedir. Bu yüzden iki boyutlu Kartezyen ağlar için dörtlü veri yapısı kullanılmıştır.

Kartezyen ağlarda toplam hücre sayısı başlangıçta bilinmemektedir. Bu nedenle, dinamik veri yapısı kullanılarak hücre sayısının program çalışırken değişebilmesi mümkün kılınmıştır.

Literatürde komşuluk ilişkilerini tanımlamak için bir çok yöntem kullanılmıştır. İki boyutlu diziler, bağlanmış liste ve ikili ağaç veri yapısı bu yöntemlere örnektir. Bu çalışmada, bir çok avantajından dolayı dörtlü ağaç veri yapısı kullanılmıştır.

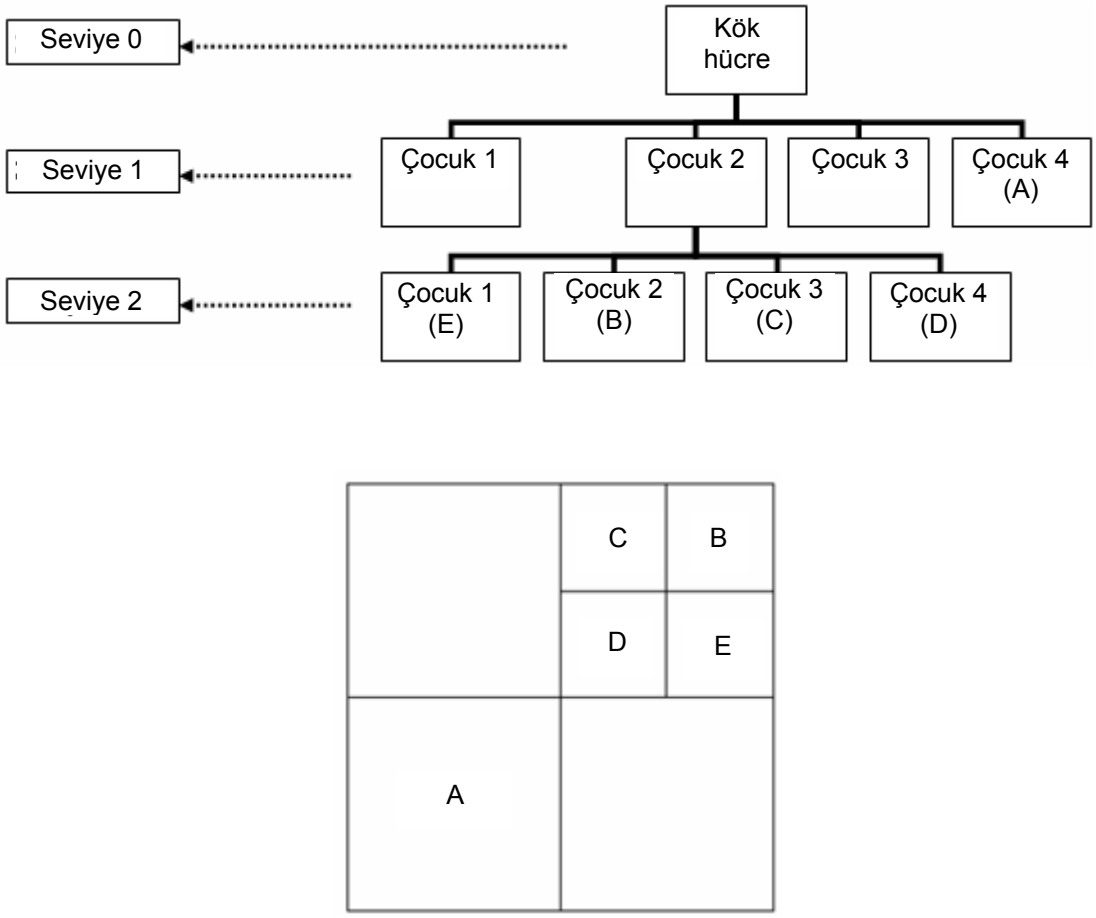
Öncelikle, dörtlü ağaç veri yapısından sekizli ağaç veri yapısına geçmek çok kolaydır ve bu durum iki boyutlu Kartezyen ağ yaratmak için hazırlanmış programın üç boyutlu hale getirilmesini kolaylaştırmaktadır. Ayrıca, çözüm adaptasyonu gibi bölgesel alanlarda yapılan hücre sayısında azalma veya artmalar kolay olduğundan ve çoklu ağ yönteminin uygulanmasını kolaylaştıran bir veri yapısı olduğundan dörtlü ağaç veri yapısı tercih edilmiştir.

Dörtlü ağaç veri yapısı, bir soyağacı olarak düşünülebilir ve bu soyağacının en büyük bireyi kök hücre olarak adlandırılmaktadır. Dörtlü ağaç veri yapısında kök hücreyi onun dört çocuğu ve çocuklarının çocukları takip etmektedir ve bu durum Şekil 2.1'de gösterilmektedir.

Geliştirilen yazılımda, bütün hücreler dokuz işaretçi (pointer) ile tanımlanmıştır. Bu işaretçilerden biri hücrenin ailesi, dördü hücrenin çocukları ve kalan dördü ise hücrenin kenar komşularıdır. Bu işaretçiler ve diğerleri aşağıda verilmiştir.

- 1 işaretçi: Hücrenin ebeveyni
- 4 işaretçi: Hücrenin çocukları
- 4 işaretçi: Hücrenin kenar komşuları

- 2 işaretçi: Hücre merkezinin *x*, *y* kordinatları
- 1 işaretçi: Hücrenin seviyesi
- 1 işaretçi: Çoklu ağ yönteminde anlatıacak hesaplama hücrelerini tanımlayan “compcell” olarak adlandırılmış değer
- 1 işaretçi: Çoklu ağ yönteminde anlatılacak “perform” olarak adlandırılmış değer



Şekil 2.1 Dörtlü ağaç veri yapısına örnek

Hücrenin ebeveynini belirten işaretçi eğer sıfır olarak atanmışsa, hücre kök hücredir ve ebeveyni yoktur. Eğer hücrenin çocukları sıfır olarak atanmışlarsa, hücreler hesaplama veya yaprak (computational or leaf) hücreler olarak adlandırılmaktadırlar. Ayrıca, hücrelerin komşularını gerektiği zaman belirlemek yerine, hücre yaratıldıktan sonra hemen belirleyip, bu bilgiyi depolamak çok daha fazla avantajlıdır. Üç boyutlu bir hücrenin komşularını belirlemek daha karmaşık bir iş olduğundan dolayı komşu belirleme

işlemi üç boyutlu Kartezyen ağ yaratma bölümünde detaylı olarak açıklanacaktır. Eğer hücrenin bir komşusu uzak alan (far field) ise, bu komşu sıfır olarak atanır.

Hücrenin merkez kordinatları ve hücrenin seviyesi de çok önemli parametrelerdir. İki boyutlu bir hücre yerine üç boyutlu bir hücrenin merkez kordinatlarının bulunması işlemi sonraki bölümde açıklanacaktır. Hücrenin seviyesi merkez ve köşe kordinatların hesaplanmasında, hücrenin bir kenarının uzunluğunun hesaplanmasında kullanıldığı için çok önemli bir parametredir. Ayrıca, Kartezyen ağlar yaratılırken bir seviye kuralı çok önemlidir. Bu kural iki komşu arasındaki seviye farkının biri geçememesi kuralıdır ve bu kural, komşular arasındaki geçişin daha düzgün ve akı hesaplamalarının daha kolay olmasını sağlamaktadır. Köşegen komşuların bulunması da bu kural sayesinde çok basitleşmektedir. Son olarak, bir seviye kuralı veri yapısının daha karmaşık hale gelmesini engellemektedir.

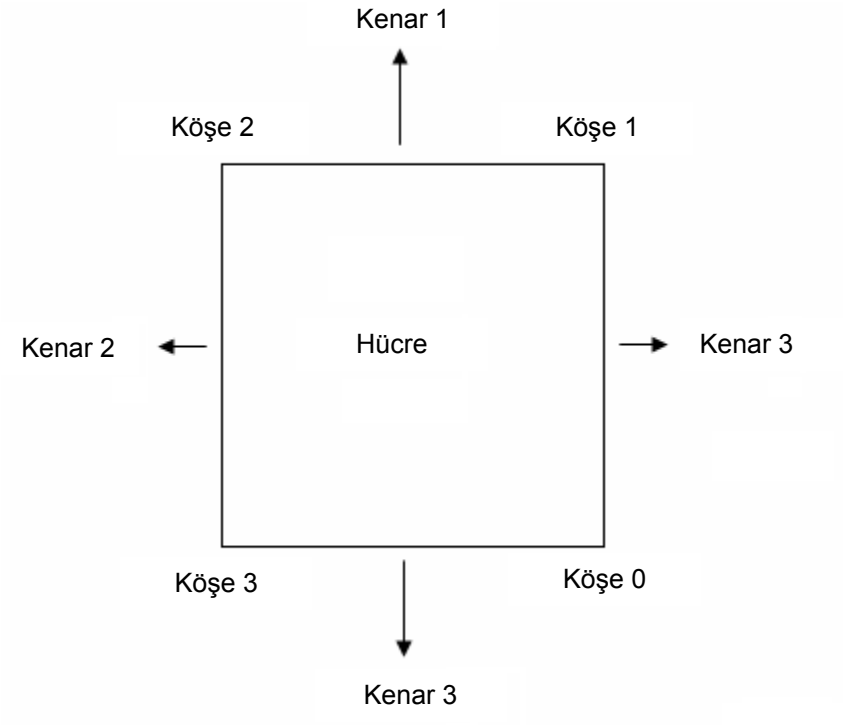
Bu arada, Şekil 2.1'de görüldüğü gibi kök hücrenin seviyesi sıfır ve çocuklarının seviyesi birdir. Diğer bir deyişle, bir çocuk hücrenin seviyesi her zaman ebeveyninin seviyesinden bir fazladır.

İki boyutlu problemler için iki tür komşu vardır. Bunlardan ilki, belirlendikten sonra depolanan kenar komşuları ve gerektiği zaman belirlenen köşegen komşularıdır. Köşegen komşuları, kenar komşuları kadar çok kullanılmadığından ve belirlenmeleri kenar komşuları sayesinde çok kolay olduğundan depolanmazlar.

Geliştirilen yazılımda dört çeşit yaprak hücre bulunmaktadır. Bunlar, dış (outside), iç (inside), kesik (cut) ve ayrık (split) yaprak hücrelerdir. Akı hesaplamalarında iç hücreler kullanılmadığından, iç hücreler yaprak yada hesaplama hücre olarak ele alınmayabilir. Yaprak hücreler için ayrıca depolanan bazı parametreler bulunmaktadır. Bunlar;

- 4 işaretçi: Hücrenin korunabilir değişkenleri
- 4 işaretçi: Hücrenin köşeleri
- 1 işaretçi: Hücrenin çeşidi
- 1 işaretçi: Hücrenin toplam kare endeksi (total square index)
- 2 işaretçi: Yoğunlaştırma ve seyretme kriterleri
- 2 işaretçi: Hız vektörünün gradyanı ve dönümü
- 2 işaretçi: *x* ve *y* yönündeki hücrenin kenarlarının iz düşümü
- 4 işaretçi: Zorlayıcı fonksiyonlar (Forcing Functions)

Yaprak hücrelerinin her bir köşesi üç parametre ile tanımlanmaktadır. Bunlar, köşenin *x*, *y* kordinatları ve ϕ değeridir. Buradaki ϕ değeri, köşenin geometri içinde kalıp kalmadığını belirlemek için kullanılmaktadır. Yaprak hücrelerinin çeşidinin belirlenmesi ve ϕ değerinin hesaplanması ileride anlatılacaktır. Köşelerin numaralandırılması Şekil 2.2'de görülmektedir.

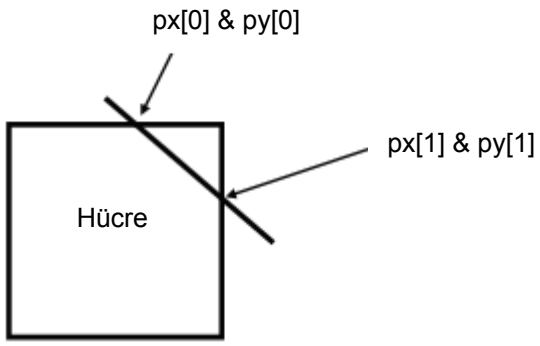


Şekil 2.2 Bir yaprak hücrenin köşe ve kenarlarının numaralandırılması

Kesik ve ayrık hücrelerin merkez kordinatları ve alanları otomatik olarak bilinmediğinden hesaplanmaları gerekmektedir. Bu yüzden bu iki hücre çeşidi için ayrıca bazı parametrelerin depolanması gerekmektedir. Kesik hücreler için bunlar;

- 3 işaretçi: Hücrenin merkezinin x, y kordinatları ve hücrenin alanı
- 4 işaretçi: Hücrenin x, y kesim noktaları (px[0], px[1], py[0] ve py[1])

Bu parametrelerin iki katı da ayrık hücreler için depolanmaktadır çünkü ayrık hücreler genelde iki ayrı hücreymiş gibi düşünülebilir. Kesim noktalarının numaralandırılması da Şekil 2.3'te görülmektedir.



Şekil 2.3 Kesim noktalarının gösterimi

Hücrenin merkez kordinatları ve alanı küçük kareler yeniden yapılandırma (least squares reconstruction) yöntemi için gerekli olduğundan, bu değerlerin hesaplanmasını göstermekte yarar vardır. Görüldüğü gibi köşelerin ve ayrıca kesim nokalarının numaralandırılması saat yönünün tersinedir. Bu durum hesaplamayı kolaylaştırmaktadır.

Öncelikle, kesik hücrelerin geometrinin dışında kalan yerleri üçgenlere bölünmektedir. Üçgenlere bölme (triangulation), ilk kesim noktasından başlar. İlk kesim noktasını takip eden saat yönü tersindeki dışarda bulunan köşeler ve ikinci kesim noktası alınır. Böylece tüm köşelerden geçen, tüm dış alanı kapsayan üçgenler ouşturulmuş olur. Sonra, her üçgeni oluşturan vektörlerin çapraz çarpımı yapılır. Bu çarpım her bir üçgenin alanını verir ve tüm üçgen alanlarının toplamı da kesik hücrenin dış alanıdır. Ayrıca, bu dış alanın merkez kordinatları da aşağıdaki denklem kullanılarak bulunmaktadır.

$$C = \frac{\sum_{i=1}^{Üçgenler} (AC)_i}{\sum_{i=1}^{Üçgenler} A_i} \tag{2.1}$$

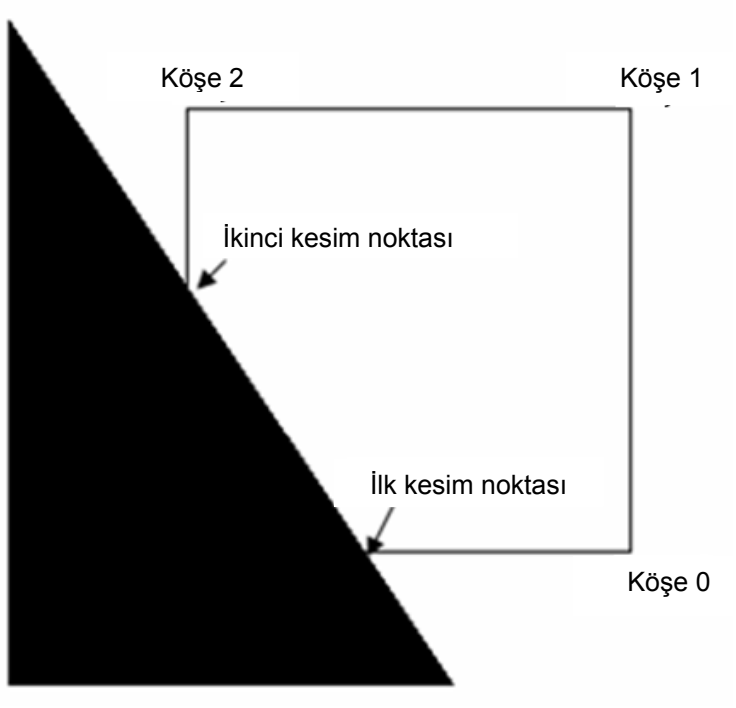
Yukarıdaki denklemde, C_i ve A_i değerleri her bir üçgenin kitle merkezi ve alanıdır. Üçgen oluşturmaya bir örnek Şekil 2.4'te görülmektedir. Bir kesik hücrenin dışında kalan köşeler, oluşturulan ve ilerde anlatılacak olan bir tablo kullanılarak otomatik olarak bulunur. Bu tablo, köşe tablosu (cornerTable) olarak adlandırılmaktadır.

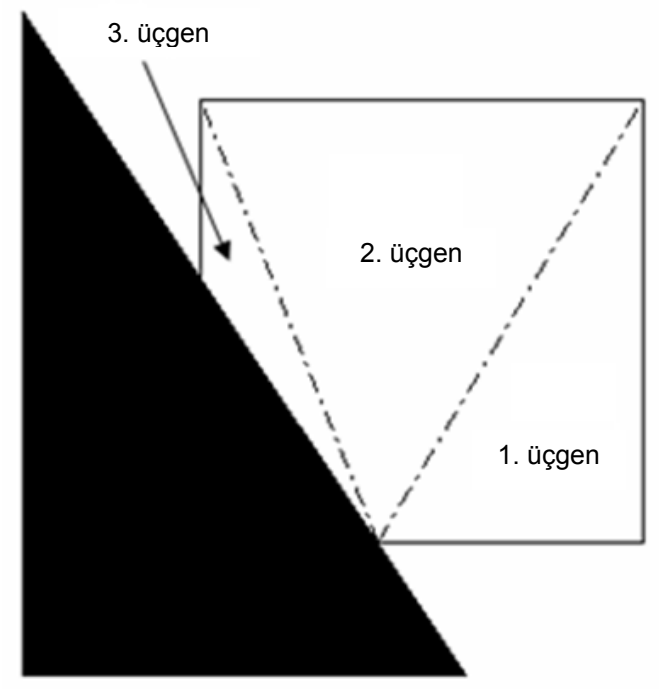
2.2 GEOMETRİNİN TANIMLANMASI VE GEOMETRİK ADAPTASYON

2.2.1 Geometrinin Tanımlanması

Kartezyen ağı yaratmadan önce çevresinde akışın olacağı geometri tanımlanmalıdır. Geometri geliştirilen yazılımda doğru parçaları olarak tanımlanır ve bu doğru parçalarını oluşturan noktaların sıralanma yönü saat yönünün tersidir. Bu noktalar kapalı bir döngü oluşturmak zorundadır. Tanımlanan geometrinin maksimum uzunluğu bulunur ve bu uzunluk kullanıcı tarafından belirlenen bir sayı ile çarpılır. Elde edilen sonuç, kök hücrenin bir kenarının uzunluğudur.

Ayrıca geliştirilen kodda boyutsuzlaştırma işlemi (nondimensionalization) uygulanmamaktadır. Çünkü boyutsuzlaştırma işlemi uygulandığında, hesaplama hatalarının çok daha fazla arttığı gözlenmiştir. Geliştirilen kod dış akış çözücü olduğundan geometri genellikle kök hücrenin merkezine yerleştirilmektedir. Fakat gerekli görüldüğünde geometriyi merkezden kaydırmak da mümkündür.

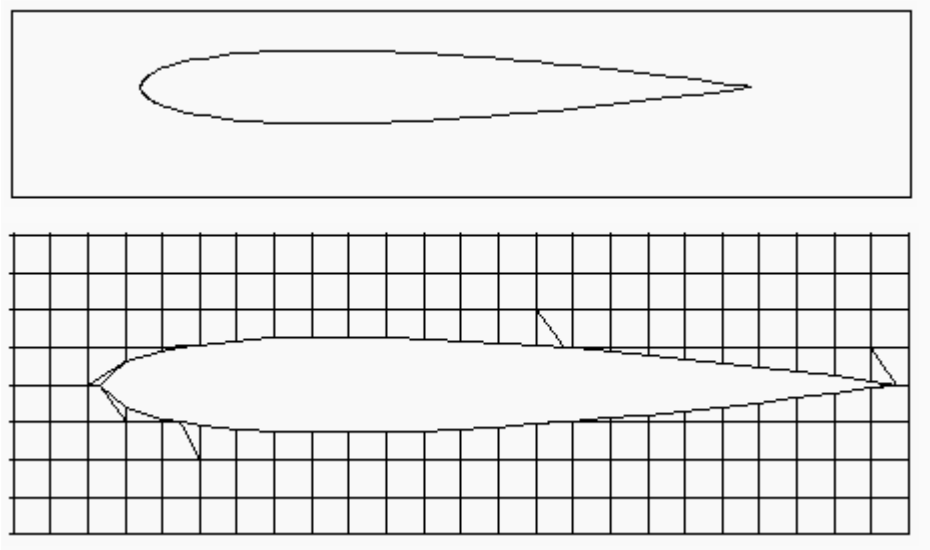




Şekil 2.4 Üçgen oluşturma işlemine bir örnek

2.2.2 Eşit Ağ Adaptasyonu

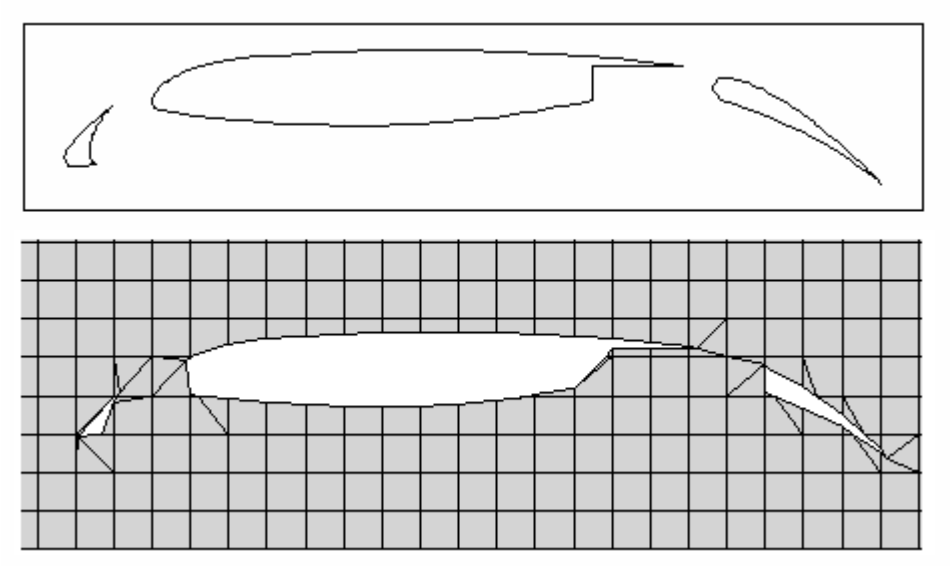
Eşit ağ adaptasyonunda, yaprak hücreler herhangi bir kriter olmadan bölünmektedir. Bu adaptasyonda bir seviye kuralını kontrol etmeye gerek yoktur. Ayrıca her hücre bölünmesinden sonra oluşan yeni çocuk hücrelerin merkez kordinatları ve komşuları hemen belirlenmelidir. Bu adaptasyonun amacı, diğer adaptasyonlar uygulanmadan önce yeteri kadar küçük hücreler elde edilmesidir. Eşit ağ adaptasyonu sırasında toplam hücre sayısı eksponansiyel olarak yükseldiği için bu tip geometrik adaptasyonun uygulanması diğer geometrik adaptasyon tiplerine göre çok daha pahalıdır. Fakat yinede başlangıçta yeterince yoğun bir ağ elde etmek önemlidir. Literatürde iki veya üç seviye eşit ağ adaptasyonunun uygulanmasının yeterli olduğu yazmaktadır. Fakat geliştirilen kodda seyreltme işlemide uygulandığından dolayı eşit ağ adaptasyon seviyesi için bir sınırlama bulunmamaktadır. Tanımlanan geometriler (NACA0012 ve üç elemanlı kanatçık profilleri) ve çevrelerinde eşit ağ adaptasyonu sonucu yaratılan Kartezyen ağlar Şekil 2.5 ve Şekil 2.6'da görülmektedir.



Şekil 2.5 NACA0012 kanatçık profili ve çevresinde yaratılan düzgün ağ (uniform mesh)

İstenilen seviyede eşit ağ adaptasyonu uygulandıktan sonra bir diğer geometrik adaptasyon olan kutu adaptasyonuna geçebilmek için yaprak hücre çeşitlerinin belirlenmesi gerekmektedir. Bir yaprak hücrenin köşelerinin tanımlanan geometrinin içinde mi dışında mı kaldığını iç-dış testi belirlemektedir.

Literatürde bir çok iç-dış testi vardır. Fakat bunlardan en popüler olan ikisi, ışın fırlatma (ray-casting) ve sayı döndürme (winding number) yöntemleridir.



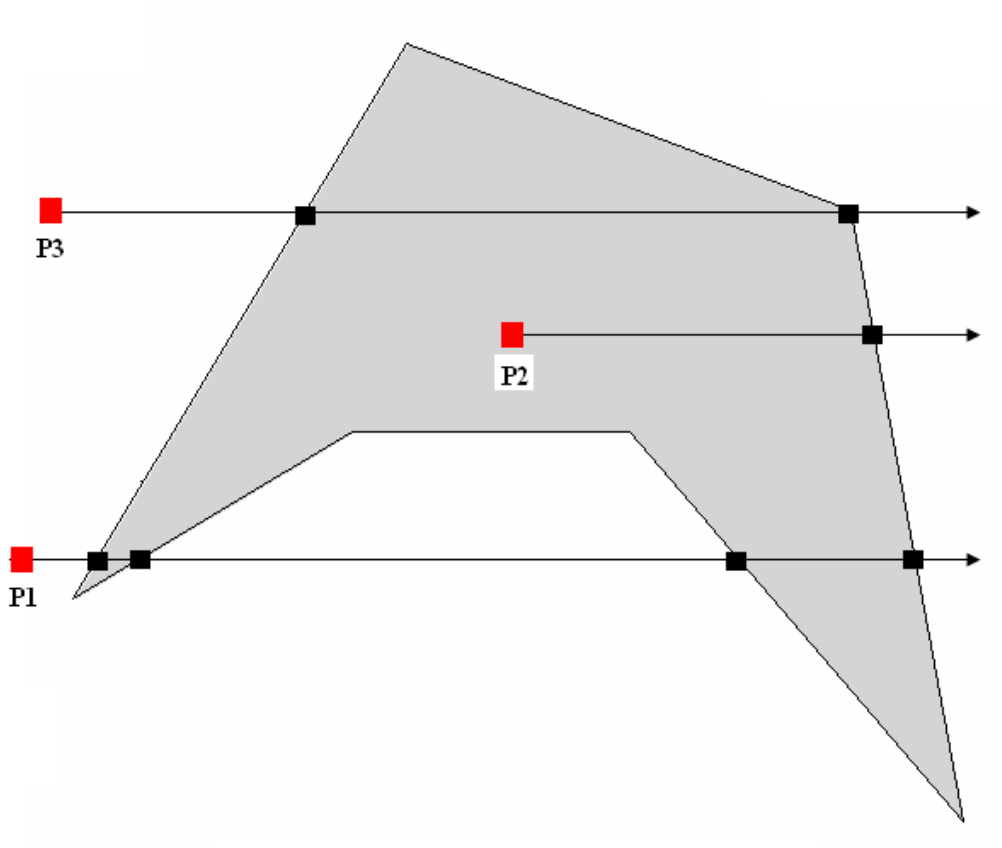
Şekil 2.6 Üç elemanlı kanatçık profili ve çevresinde yaratılan düzgün ağ (uniform mesh)

Bu çalışmada, ışın fırlatma yöntemi birçok avantajından dolayı tercih edilmiştir. Fakat bu iki yöntem içinde bir kısıtlama bulunmaktadır. Bu kısıtlama şöyle özetlenebilir. Geometrinin tanımlanması için verilen noktalar kesinlikle kapalı bir döngü oluşturmalıdır. Bu kapalı döngünün içinde deliklerin bulunması iç-dış testlerinin uygulanmasında bir sorun teşkil etmekedir.

Işın Fırlatma Yöntemi: Önceden de bahsedildiği gibi bu yöntem, hücrenin köşe noktalarının tanımlanan geometri içinde kalıp kalmadığının tespit edilmesi için gereklidir. Öncelikle, test edilecek noktadan sabit *x* veya *y* doğruları boyunca iki boyutlu problemler için ışın fırlatılır. Bu ışının, geometriyi oluşturan doğru parçalarından kaçını kestiği hesaplanır ve hesaplanan değer tek sayı çıkarsa, ele alınan köşe geometrinin içinde, çift sayı çıkarsa geometrinin dışında demektir.

Işın fırlatma yöntemini daha detaylı anlatabilmek için Şekil 2.7'de verilen örnekteki üç farklı durumu incelemek faydalı olacaktır. İlk durumda, pozitif *x* yönünde P1 noktasından fırlatılan ışın, basit kapalı geometriyi dört kez kesmiştir. Yani ışın atma yöntemine göre bu nokta kesim sayısı çift sayı olduğundan geometrinin dışında bulunmaktadır. İkinci durumda, pozitif *x* yönünde P2 noktasından fırlatılan ışın, basit kapalı geometriyi bir kez kesmiştir. Yani ışın atma yöntemine göre bu nokta kesim sayısı tek sayı olduğundan geometrinin içinde bulunmaktadır. Son durumda, pozitif *x* yönünde P3 noktasından fırlatılan ışın, basit kapalı geometriyi üç kez kesmiştir. Diğer bir deyişle, bu ışın tarafından geometriyi oluşturan üç ayrı doğru kesilmiştir. Yani ışın atma yöntemine göre bu nokta kesim sayısı tek sayı olduğundan geometrinin içinde bulunmaktadır. Fakat şekilde de görüldüğü gibi P3 noktası geometrinin içinde değil, dışında bulunan bir noktadır. Bu problem, bir noktadan fırlatılan ışının geometriyi

oluşturan doğruların bitiş noktalarından biriyle kesişmesi sonucu ortaya çıkmaktadır. Çünkü, bu bitiş noktaları iki kez sayılmaktadır. Bu problemi ortadan kaldırmak için bitiş noktalarındaki kesişimleri bir kez saymak yerine, fırlatılan ışının yönünü değiştirmenin daha faydalı olduğu görülmüştür. Çünkü, iki boyutlu problemlerde sorun sadece bitiş noktalarından kaynaklanmasına rağmen, ışın fırlatma yöntemi üç boyutlu problemlere uygulandığında oluşan sorunlar daha karmaşık hale gelmektedir ve bu yüzden ışının yönünü değiştirmek en faydalı çözümdür. Pozitif *x* yönündeki ışın bitiş noktası ile kesiştiği için P3 noktasından pozitif y yönünde bir ışın fırlatılır ve görüldüğü gibi bu ışının geometri ile kesişme sayısı sıfırdır ve bu yüzden P3 noktası geometrinin dışındadır.

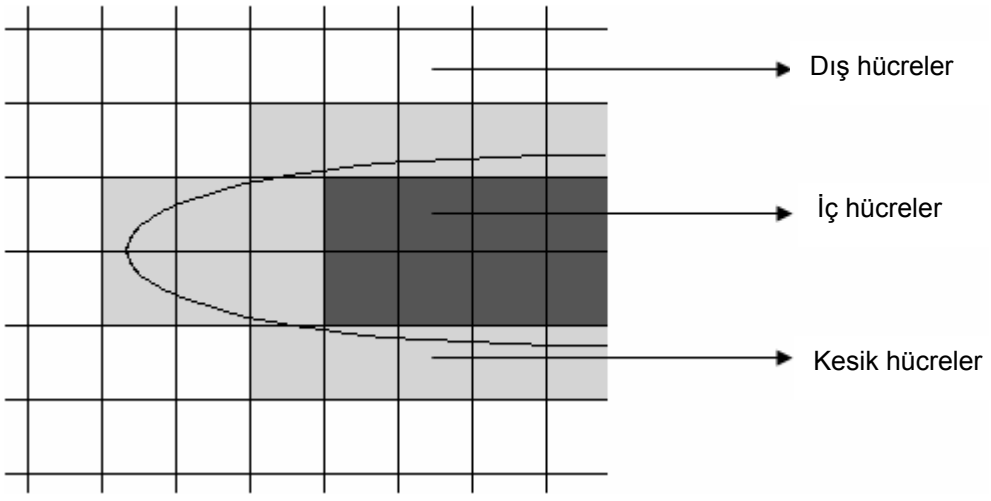


Şekil 2.7 Işın fırlatma yöntemine bir örnek

Işın atma yönteminin kuralını çiğneyen bir özel durum daha bulunmaktadır. Eğer bir nokta, geometriyi oluşturan herhangi bir doğru parçası ile kesişiyorsa, bu noktadan fırlatılan ışının geometriyi kesme sayısı tek veya çift çıkabilir. Bu yüzden, ışın fırlatma yöntemi bir noktaya uygulanmadan önce, bu noktanın geometriyi oluşturan herhangi bir doğru parçası ile kesişip kesişmediği kontrol edilmeli, eğer kesişmiyorsa ışın atma yöntemi uygulanmalıdır. Zaten kesişiyorsa, bu nokta otomatik olarak geometrinin içinde olarak atanmalıdır.

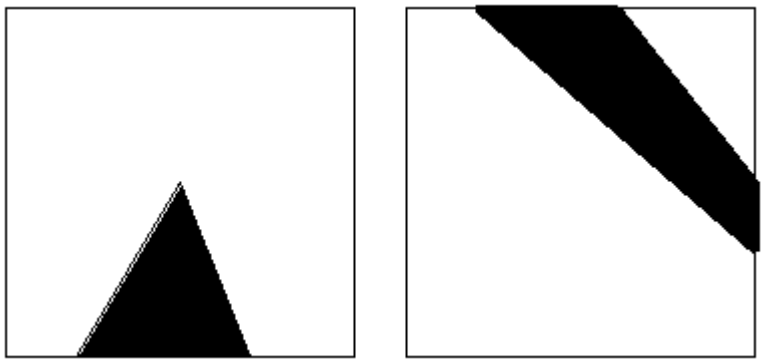
Öncedende belirtildiği gibi ışın atma yönteminin sayı döndürme (winding number) yöntemi ile karşılaştırıldığında bir çok avantajı bulunmakadır. Bunlardan ilki, ışın fırlatma yönteminin geometriyi oluşturan tüm doğru parçalarını ele alması gerekmemektedir. Örneğin bir noktadan pozitif *x* yönünde fırlatılan bir ışının geometrinin tüm parçalarını kesip kesmediğini kontrol etmektense, öncelikle bu noktanın *y* kordinatının, hangi doğru parçalarının bitiş noktalarının *y* kordinatlarının arasında kalıp kalmadığı tespit edilir. Eğer herhangi bir doğru parçasının her iki bitiş noktasınında test edilen noktanın *y* kordinatından büyük veya küçük olduğu tespit edilirse, bu parçaların ışınla kesişip kesişmediği test edilmez. Çünkü kesişmedikleri otomatik olarak bilinmektedir. Bu yüzden, sayı döndürme yöntemine göre ışın fırlatma yöntemi daha hızlıdır. Ayrıca, ışın fırlatma yönteminde sonuçlar yuvarlama hatalarından (floating round off errors) etkilenmezler. Son olarak, ışın fırlatma yöntemini üç boyutlu problemlere uygulamak da çok basittir.

Bir yaprak hücrenin tüm köşelerine iç-dış testi uygulandıktan sonra, hücrenin çeşidi belirlenmelidir. Her köşe için atanan ϕ değerlerinin yardımı ile hücrenin çeşidi belirlenmektedir. Bir hücrenin köşesinin ϕ değeri eğer 1 ise bu köşe dışarda, -1 ise bu köşe geometrinin içindedir. Köşelerin ϕ değeri ileride anlatılacak olan karelerle ilerleme (marching squares) yöntemi için çok önemlidir. Önceden de bahsedildiği gibi dört çeşit yaprak hücre çeşidi vardır. Eğer bir hücrenin köşelerinin ϕ değerlerinin hepsi 1 olarak atanmış ise hücre dış hücredir. Eğer bir hücrenin köşelerinin ϕ değerlerinin hepsi -1 olarak atanmış ise hücre iç hücredir. Geriye kalan tüm hücrelerde kesik hücre olarak atanırlar. Şekil 2.8'de iç, dış ve kesik hücreler görülmektedir.



Şekil 2.8 Hücre çeşitlerine örnekler

Bu sınıflandırma yapıldıktan sonra bazı özel durumlar için yeni bir çeşit yaprak hücre gerekli olduğu görüldü. Şekil 2.9'da iki özel durum gösterilmektedir. Şekil 2.9'daki iki hücrenin de tüm köşelerinin ϕ değerleri 1 olarak atanmıştır ve yukarıda anlatıldığı gibi bu hücreler ilk olarak dış hücreler olarak atanmışlardır. Fakat bu hücreleri dış hücre olarak kabul edip, akı hesaplaması yapmak büyük yanlıştır. Bu yüzden, bu gibi özel olan hücreler geliştirilen programda kesik hücre olarak atanmaktadır. Bu hücreler ileride detaylı olarak anlatılacaklardır.



Şekil 2.9 Özel durumlara iki örnek

Hücre çeşitleri belirlendikten sonra geometri çevresindeki iç hücreler hariç diğer yaprak hücrelere üç çeşit adaptasyon uygulanmaktadır. Bu adaptasyonların hepsine birden geometrik adaptasyon denilmektedir.

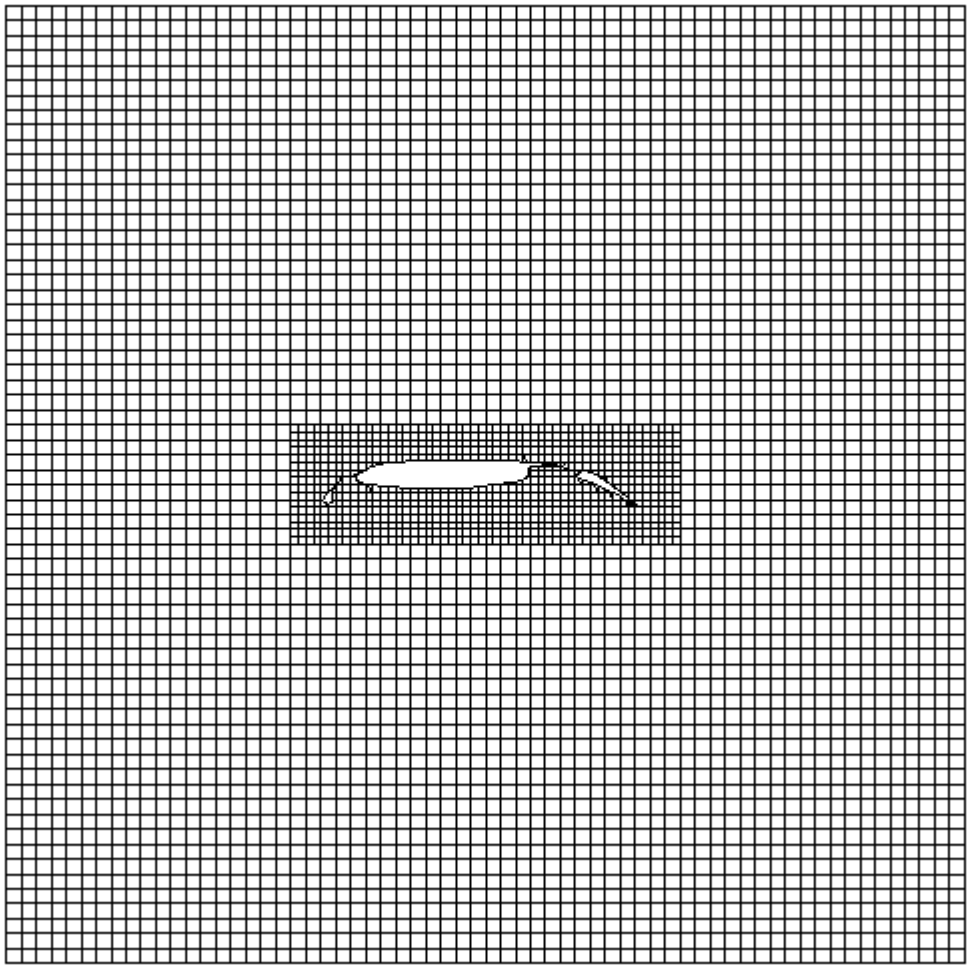
2.2.3 Geometrik Adaptasyon

Geometrinin tanımlanmasından ve eşit ağ adaptasyonundan sonra problemin çözümüne geçilmeden önce geometri çevresine geometrik adaptasyon uygulanır. Kutu adaptasyonu, kesik&ayrık hücre adaptasyonu ve eğrilik adaptasyonu olmak üzere üç tip geometrik adaptasyon vardır.

2.2.3.1 Kutu Adaptasyonu

Eşit ağ adaptasyonundan sonra, duvara yakın yerlerde daha küçük hücreler elde edilebilmesi için kutu adaptasyonu uygulanır. Akış alanı içerisinde bulunan cisim hayali bir kutu içerisine alınarak kutu adaptasyonuna başlanır. Tanımlanan geometri tarafından kesilen hücreler yeterli derecede hassas hesaplama ağı elde edilene kadar bölünür. Böylelikle cisim etrafında daha küçük hücrelerin elde edilmesi mümkün olur. Unutulmamalıdır ki, kutu içinde kalan hücreler bölünmeden önce çocuklarının bir seviye kuralına uyup uymadıkları kontrol edilir ve eğer uyuyorlarsa, hücre bölünür. Eğer herhangi biri uymuyorsa,

öncelikle bölünecek olan hücrenin komşusu bölünür ve böylelikle bir seviye kuralı hiçbir zaman ihlal edilmemiş olur. Şekil 2.6'daki eşit ağ adaptasyonuna, kutu adaptasyonu uygulanırsa Şekil 2.10 elde edilir.



Şekil 2.10 Kutu adaptasyonuna bir örnek

Kutu adaptasyonu ile kesik&ayrık hücre adaptasyonu arasında uygulanması gereken ara işlemler bulunmaktadır. Bunlar, kesik hücrelerin belirlenmesi, sınıflandırılması ve karelerle ilerleme (marching squares) yönteminin uygulanmasıdır.

2.2.3.1.1 Kesik Hücrelerin Belirlenmesi ve Sınıflandırıması

İç-dış testinden sonra yaprak hücreler, iç, dış ve kesik olmak üzere üç gruba ayrılmışlardı. Fakat önceden de belirtildiği gibi bu sınıflandırmayı bozan bazı özel hücreler için yeni bir yaprak hücre çeşidine

gereksinim duyulmuştur. Bu yüzden, bu özel hücrelerin çeşidi ayrık olarak belirlenmiştir. Ayrık hücreler diğer hücreler gibi tek bir hesaplama alanından veya ayrı iki hesaplama alanınından oluşabilirler. Tek hesaplama alanından oluşan ayrık hücrelerde akı hesaplaması kesik hücrelerinkine benzemektedir. Öte yandan, ayrı iki hesaplama alanından oluşan ayrık hücrelerde akı hesaplanması, her alan için ayrı ayrı yapılmaktadır. Ayrık hücreler, veri yapısını çok daha karmaşık hale getirmelerine rağmen çoklu ağ yönteminin uygulanabilmesi için zorunludurlar. Bu zorunluluk ayrıntılı bir şekilde ileride anlatılacaktır. Ayrık hücreler, veri yapısını karmaşık hale getirmenin dışında hesaplama zamanını ve hafıza kullanımını da arttırmaktadır. Bu dezavantajları yüzünden literatürdeki bazı çalışmalar, ayrık hücre kullanmaktan kaçınmışlardır.

Bu çalışmada, çoklu ağ yöntemini verimli bir şekilde kullanabilmek için ayrık hücrelerden faydalanılmıştır. İç-dış testinden sonra üç gruba ayrılan yaprak hücrelerden, özel olanları belirlenir ve ayrık hücre olarak atanırlar. Bu aşamaya ayrık hücrelerin belirlenmesi denilmektedir. Mesela, dış hücre olarak atanan bir yaprak hücrenin geometri tarafından kesilmiş bir kenarı varsa, bu hücrenin çeşidi dış hücreden ayrık hücreye çevrilmelidir. Aynı şekilde iç-dış testinden sonra iç hücre olarak atanan bir yaprak hücreninde bir kenarı geometri tarafından kesilmiş ise bu hücrede ayrık hücre olarak atanır. Bilindiği gibi kesik hücrelerin de sadece iki tane kesim noktası bulunmalıdır. Eğer ikiden fazla kesim noktası bulunan bir kesik hücre varsa, bu hücrede aynı diğerleri gibi özel bir hücredir ve ayrık hücre olarak atanmalıdır. Bu durumlar Şekil 2.11'de örneklendirilmiştir.

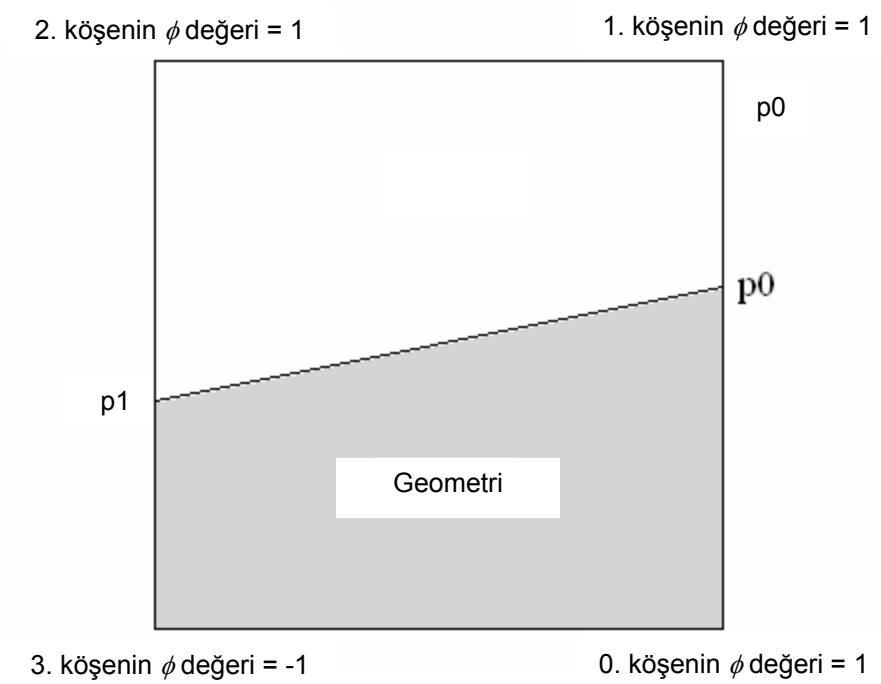
Şekil 2.11 Ayrık hücrelere üç örnek

Dörtlü ağaç veri yapısında da bahsedildiği gibi herbir yaprak hücrenin toplam kare endeksi adında bir işaretçisi vardır. Bu işaretçi akı hesaplama işlemini çok kolaylaştırmaktadır. Şekil 2.2'de bir hücrenin kenarlarının ve köşelerinin nasıl numaralandığı örneklenmektedir. Yaprak hücreyi oluşturan köşe

noktalarının ϕ değerlerine bakılarak toplam kare endeksi hesaplanır. Her köşenin eğer ki ϕ değeri -1 ise kendine ait kare endeksi vardır ve bu değerler, aşağıda gösterilmektedir.

- Eğer 0. köşenin ϕ değeri = -1, bu köşenin kare endeksi= 1
- Eğer 1. köşenin ϕ değeri = -1, bu köşenin kare endeksi= 2
- Eğer 2. köşenin ϕ değeri = -1, bu köşenin kare endeksi= 4
- Eğer 3. köşenin ϕ değeri = -1, bu köşenin kare endeksi= 8

Mesela Şekil 2.12'de verilen hücrenin 0. ve 3. köşelerinin ϕ değerleri -1'dir ve yukarıdaki bilgiler ışığında bu hücrenin toplam kare endeksi dokuz bulunur (1+8). Dış hücrelerin toplam kare endeksi sıfır ve iç hücrelerin toplam kare endeksi onbeştir.



Şekil 2.12 Toplam kare endeksinin hesaplanmasına bir örnek

Ayrık hücreler belirlendikten sonra bu hücreler sınıflandırılırlar. Bu sınıflandırma, ayrık hücrelerin toplam kare endekslerine göre yapılmaktadır. Eğer bir yaprak hücre, iç-dış testinden sonra kesik hücre olarak atandıysa ve toplam kesim sayısı dört olarak bulunduysa bu hücre ayrık hücre olarak atanır fakat kesik hücre iken bulunan toplam kare endeksi aynı kalır. Eğer bir yaprak hücre, iç-dış testinden sonra dış hücre olarak atandıysa fakat kenarları geometri tarafından kesik ise toplam kesişme sayısı bulunur ve bu sayı iki ise toplam kare endeksi -15, dört ise -25 olarak atanır. Eğer bir yaprak hücre, iç-dış testinden

sonra iç hücre olarak atandıysa fakat kenarları geometri tarafından kesik ise toplam kesişme sayısı bulunur ve bu sayı iki ise toplam kare endeksi -20, dört ise -30 olarak atanır. Son olarak, dörtten fazla kesim noktası bulunan kesik hücrelerin toplam kare endeksi -40 olarak atanır ve bu hücreler kesim sayısı dört ve daha az olan hücreler elde edilene kadar bölünürler. Genelde böyle hücrelerle çok nadiren karşılaşılmakta ve bir veya iki bölmeden sonra sorun çözülmektedir.

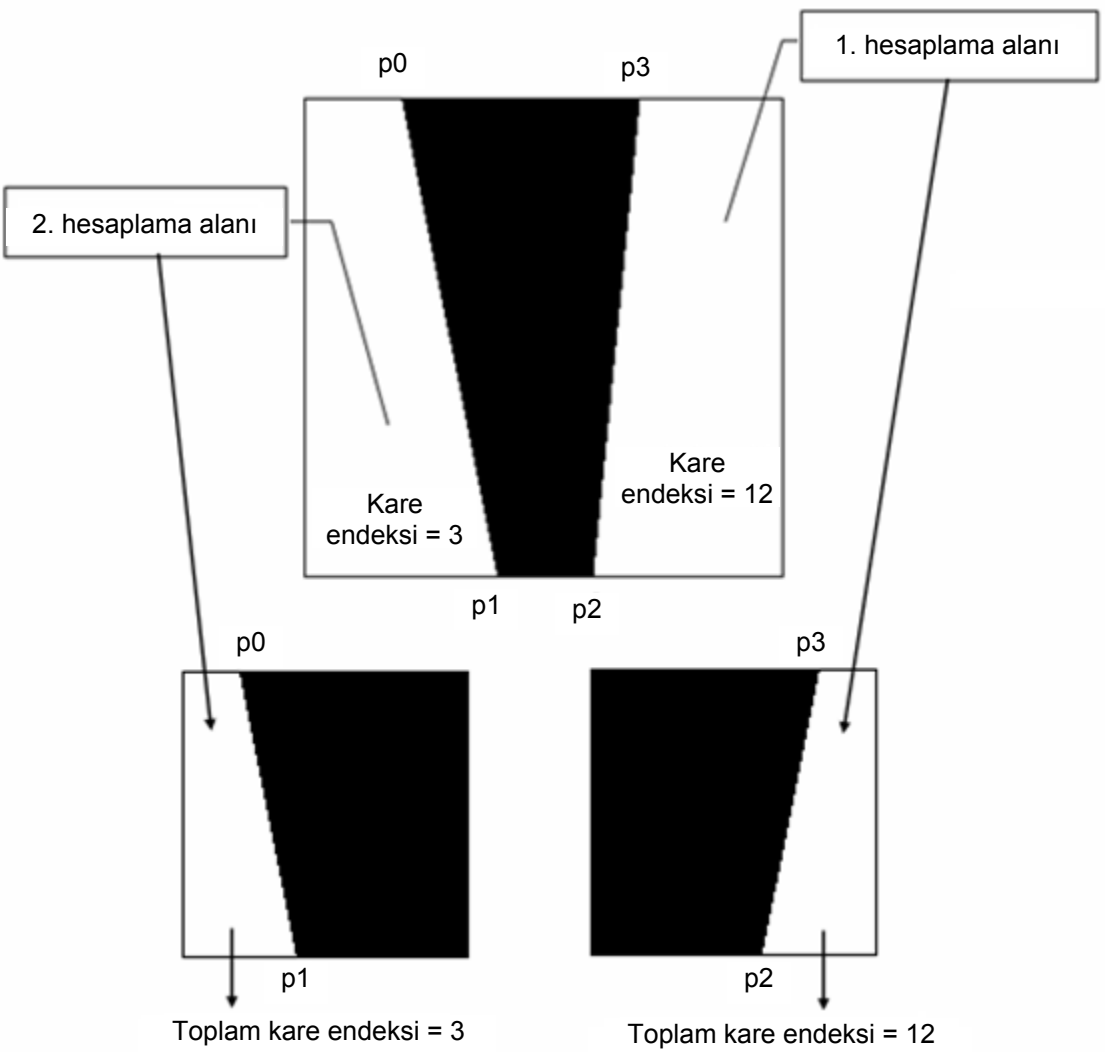
Ayrık hücreler sınıflandırıldıktan sonra, kesim noktaları numaralandırılmalıdır ve numaralandırma gelişigüzel yapılmamalıdır. Ayrık hücrelerin alanlarının, merkez kordinatlarının hesaplanması ve ayrıca akı hesaplanması zorlaşmasın diye numaralandırma belirli bir sıra izlemelidir. Ayrıca iki hesaplama alanından oluşan ayrık hücrelerin akı hesaplaması, kesik hücrelerin hesaplanmasına benzemesi için her bir ayrı hesaplama alanının sanki kesik hücreymiş gibi özel bir kare endeksi daha vardır. Bu bilgiler ekler kısmında sunulmaktadır. Mesela, toplam kare endeksi -25 olan Şekil 2.13'teki yaprak hücre iki ayrı hesaplama alanından oluşmaktadır ve ilk hesaplama alanı kesik hücre gibi düşünülseydi, bu kesik hücrenin kesim noktalarının ilki p0, ikincisi p1 ve toplam kare endeksi de 3 olurdu. Ayrıca ikinci hesaplama alanıda kesik bir hücre gibi düşünülseydi, bu kesik hücrenin kesim noktalarının ilki p2, ikincisi p3 ve toplam kare endeksi de 12 olurdu. Mesela birinci hesaplama alanındaki akılar hesaplanmak istendiğinde, bu alan toplam kare endeksi 3 olan bir kesik hücrenin akısının hesaplandığı gibi hesaplanır.

2.2.3.1.2 Karelerle İlerleme Yöntemi

Kutu adaptasyonundan ve ayrık hücrelerin belirlenmesinden sonraki aşama kesik ve ayrık hücrelerin kesim noktalarının kordinatlarının belirlenmesi işlemidir. Geliştirilen kodda bu işlem karelerle ilerleme yöntemi kullanılarak yapılmaktadır. Literatürde, bu yöntem yerine, iki boyutlu problemler için doğru veya poligon klipsleme yöntemleri kullanılmaktadır. Polygon veya doğru klipsleme yöntemlerinde her kenarın kesik olup olmadığı test edilir ve kesik olan kenarlarda kesim noktasının kordinatları tespit edilir. Sonra kesik ve kesik olmayan kenarlar ayrı ayrı hafızada depolanır. Bu durum, hafızanın verimsiz kullanıldığının bir göstergesidir. İki boyutlu problemler için hafızayı verimsiz kullanmak büyük problem yaratmasada, üç boyutlu problemler için verimli hafıza kullanımı çok önemlidir. Karelerle ilerleme yönteminde tüm kenarları depolamak yerine sadece toplam kare endeksi adı verilen işaretçinin depolanması yeterlidir. Bu işaretçi kullanılarak kesik ve kesik olmayan kenarlar bir tablo sayesinde otomatik olarak bulunurlar.

Şekil 2.12'deki hücre örnek alınarak kesik olan ve olmayan kenarların tespit edilmesi ve kenarlardan geçen akının karelerle ilerleme yöntemi ile hesaplanması daha ayrıntılı olarak bu bölümde anlatılacaktır. Bilindiği gibi Şekil 2.12'deki hücre, kare endeksi 9 olan bir kesik hücredir. Bu hücrenin kesim noktalarının hangi kenarlarda olduğunu tespit etmek için Şekil 2.14'te verilen doğru tablosundan (lineTable) yararlanılır. Bu tabloda kare endeksi 9 olan hücreler için kesim noktaları kırmızı bir kutu içine alınarak gösterilmiştir. Bu kutudaki ilk sayı (0), ilk kesim noktasının, p0, 0. kenarda olduğunu ve ikinci sayı

(2), ikinci kesim noktasının, p1, 2. kenarda olduğunu belirtmektedir. Böylece kesik kenarlar otomatik olarak bulunmuştur. Bu bilgi ışığında bu kenarlardaki kesim noktalarının kordinatları iki doğru kesişimi denkleminden kolaylıkla bulunur.



Şekil 2.13 Bir ayrık hücrenin kesim noktalarının numaralandırılması ve ayrı hesaplama alanlarının kesik hücre gibi düşünüldüğünde aldıkları toplam kare endeksleri

Ayrıca, akı hesaplamak da karelerle ilerleme yöntemi kullanılarak çok kolaydır. Bu işlem içinde köşe tablosundan (cornerTable) faydalanılır. Önceden bahsedildiği gibi Şekil 2.12'deki hücrenin toplam kare endeksi dokuzdur ve bu kare endekse karşılık gelen sıra, köşe tablosunda kırmızı kutu içine alınmıştır. Bu kutu içindeki ilk sayı, dışarda kalan ilk köşeyi verir. Yani saat yönünün tersi yönünde ilk kesim noktasından sonra gelen ilk dışarda kalan köşe, 1. köşedir. Akı hesaplamasına öncelikle ilk kesim noktası (p0) ve 0. köşe arasında kalan kenardan başlanır. Bu hesaptan sonra akının hesaplanacağı ikinci

kenara geçilir. İkinci kenar, kırmızı kutu içerisinde bulunan ilk sayı (1) ile ikinci sayı (2) arasında kalan kenardır. Yani 1. köşe ile 2. köşe arasında kalan kenardan (1. kenar) geçen akı hesaplanır. Daha sonra üçüncü kenar, kırmızı kutu içerisinde bulunan ikinci sayı (2) ile üçüncü sayı (-1) arasında kalan kenardır. Yalnız farkedildiği üzere üçüncü sayı tabloda -1 olarak gösterilmektedir. Böyle bir köşe olmadığına göre köşe olarak -1 görüldüğünde geliştirilen kod artık dışarda köşe kalmadığını anlar ve -1 olarak atanan köşe yerine ikinci kesim noktasını, p1, koyar. Yani son akının hesaplanacağı kenar, 2. köşe ile ikinci kesim noktası (p1) arasında kalan kenardır.

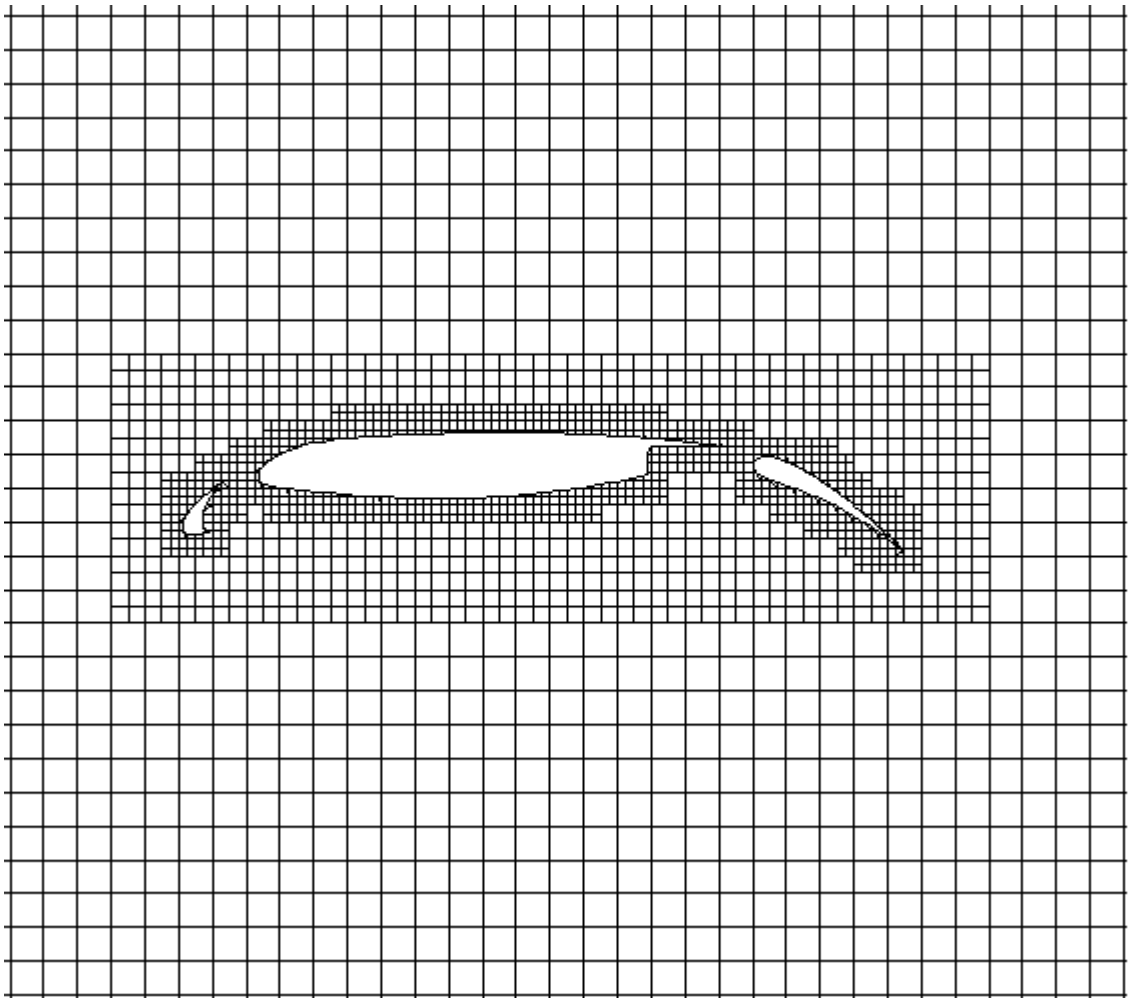
```
int lineTable[16][6] =
{{-1, -1, -1, -1, -1, -1}, //0
 {0 ,  3, -1, -1, -1, -1}, //1
 {1 ,  0, -1, -1, -1, -1}, //2
 {1 ,  3, -1, -1, -1, -1}, //3
 {2 ,  1, -1, -1, -1, -1}, //4
 {2 ,  1,  0,  3, -1, -1}, //5
 {2 ,  0, -1, -1, -1, -1}, //6
 {2 ,  3, -1, -1, -1, -1}, //7
 {3 ,  2, -1, -1, -1, -1}, //8
 {0 ,  2, -1, -1, -1, -1}, //9
 {3 ,  2,  1,  0, -1, -1}, //10
 {1 ,  2, -1, -1, -1, -1}, //11
 {3 ,  1, -1, -1, -1, -1}, //12
 {0 ,  1, -1, -1, -1, -1}, //13
 {3 ,  0, -1, -1, -1, -1}, //14
 {-1, -1, -1, -1, -1, -1}   };
```

```
int cornerTable[16][4] =
{{-1, -1, -1, -1}, //0
{1 ,  2,  3, -1}, //1
{2 ,  3,  0, -1}, //2
{2 ,  3, -1, -1}, //3
{3 ,  0,  1, -1}, //4
{-1, -1, -1, -1}, //5
{3 ,  0, -1, -1}, //6
{3 , -1, -1, -1}, //7
{0 ,  1,  2, -1}, //8
{1 ,  2, -1, -1}, //9
{-1, -1, -1, -1}, //10
{2 , -1, -1, -1}, //11
{0 ,  1, -1, -1}, //12
{1 , -1, -1, -1}, //13
{0 , -1, -1, -1}, //14
{-1, -1, -1, -1}   };
```

Şekil 2.14 Karelerle ilerleme yöntemi için gerekli olan tablolar

2.2.3.2 Kesik&Ayrık Hücre Adaptasyonu

Kutu adaptasyonundan sonra geometri çevresinde daha yoğun bir ağ yaratmak ve kesik hücrelerle dış hücreler arasındaki geçişin daha düzgün olması sağlamak için bu adaptasyondan faydalanılır. Bu adaptasyonda da bir seviye kuralı kontrol edilerek hücre bölünmesi yapılmalıdır. Şekil 9'daki kutu adaptasyonlu ağa kesik ve ayrık hücre adaptasyonu uygulanırsa Şekil 2.15 elde edilir.

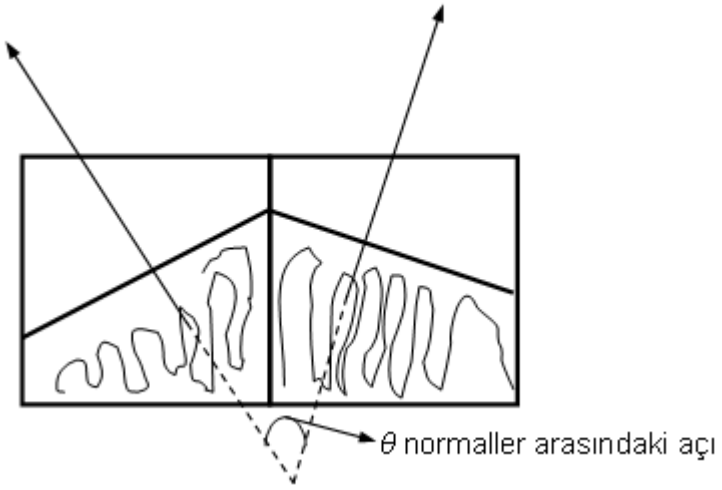


Şekil 2.15 Kesik ve ayrık hücre adaptasyonuna bir örnek

2.2.3.3 Eğrilik Adaptasyonu

Ağ üretimi işlemindeki son adım, eğrilik adaptasyonudur. Eğrilik adaptasyonunun amacı, parçanın yüksek eğriliğe sahip olduğu alanlarda, parçayı yeteri kadar doğru betimleyecek çözünürlüğün olduğunu kesinleştirmektir. Çünkü bu alanlar yüksek gradyanlar ile ilişkili olacaktırlar.

Eğrilik adaptasyonunda, hücrenin yoğunlaştırılıp yoğunlaştırılmayacağına karar vermek için komşu kesik hücreleri kesen kenarlar değerlendirilirler. Bu yüzden öncelikle komşu kesik hücreler belirlenmelidir sonra eğrilik yoğunlaştırma kriteri geliştirmek için eğrilik ölçülmelidir. Eğriliği ölçmek için akış alanını gösteren kesen kenarların normal vektörleri arasında kalan açı kullanılır. Eğer açının değeri eşik değerini aşarsa, hücre yoğunlaştırılır. Bu durum Şekil 2.16'da gösterilmiştir.



Şekil 2.16 İki komşu kesik hücrenin normalleri

İki kesik komşu hücrenin normal vektörleri aşağıdaki denklemler kullanılarak elde edilir.

$$\mathbf{n_1} = \Delta y\mathbf{i} - \Delta x\mathbf{j} = (y_1 - y_0)\mathbf{i} - (x_1 - x_0)\mathbf{j} \tag{2.2}$$

$$\mathbf{n_2} = \Delta y\mathbf{i} - \Delta x\mathbf{j} = (y_1 - y_0)\mathbf{i} - (x_1 - x_0)\mathbf{j} \tag{2.3}$$

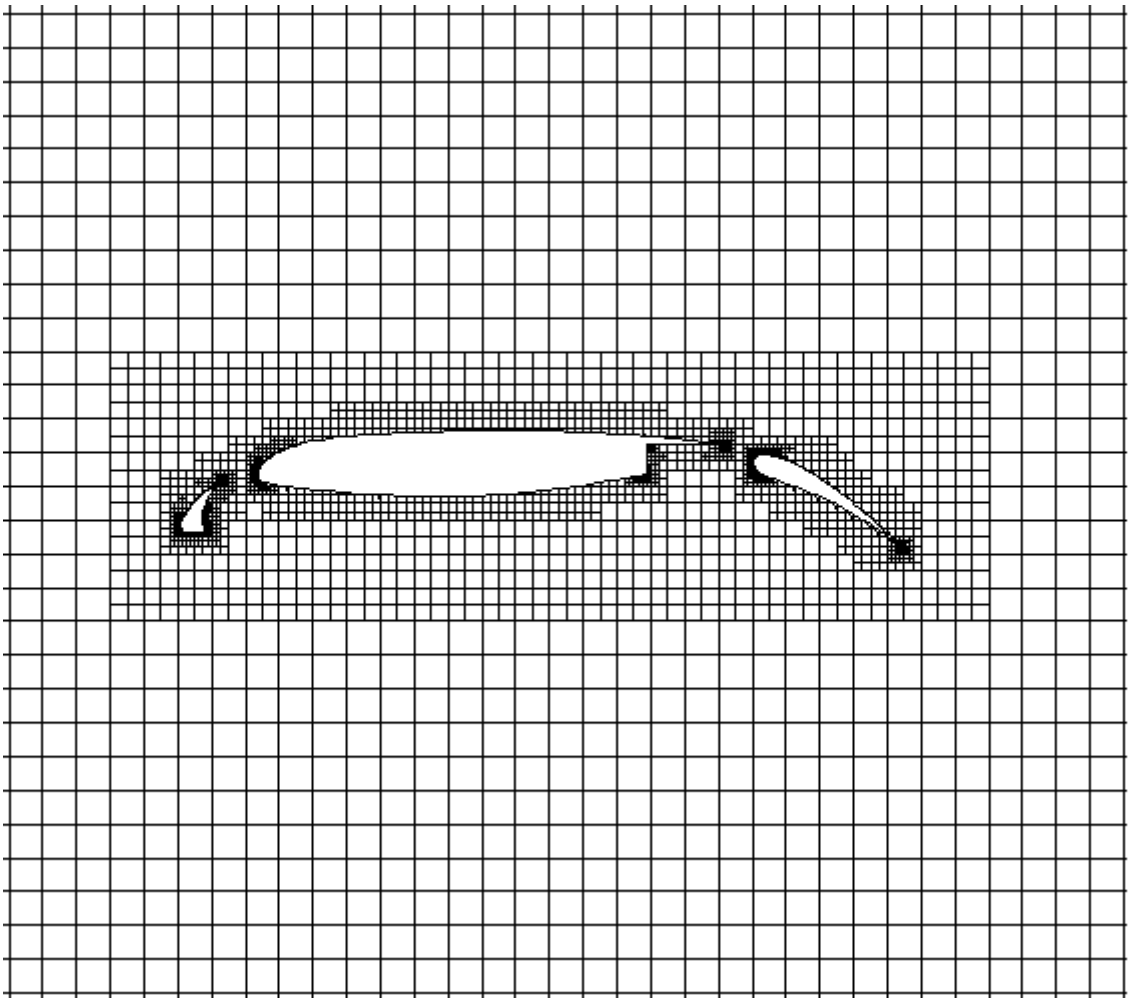
Sonra, normal vektörler arasında kalan açı bağlantısı aşağıdaki denklemle elde edilir.

$$\mathbf{n}_1\mathbf{n}_2 = n_{1x}n_{2x} + n_{1y}n_{2y} = |\mathbf{n}_1||\mathbf{n}_2|\cos\theta \Rightarrow \cos\theta = \frac{n_{1x}n_{2x} + n_{1y}n_{2y}}{|\mathbf{n}_1||\mathbf{n}_2|} \tag{2.4}$$

Sonra θ değeri eşik değerinden büyük ise, hücre yoğunlaştırılır.

$$\cos\theta = \frac{n_{1x}n_{2x} + n_{1y}n_{2y}}{|\mathbf{n}_1||\mathbf{n}_2|} < \cos\theta_{esik} \tag{2.5}$$

Şekil 2.14'teki hesaplama ağına eğrilik adaptasyonu uygulandığında Şekil 2.17'deki hesaplama ağı elde edilir.



Şekil 2.17 Eğrilik adaptasyonuna bir örnek

BÖLÜM 3
ÜÇ BOYUTLU KARTEZYEN AĞ ÜRETİCİSİNİN HAZIRLANMASI

3.1 GİRİŞ

Genel olarak, Hesaplamalı Akışkanlar Dinamiği'nde sonuçların hassas olması istendiğinden çok ufak hücrelerden oluşan hesaplama ağlarının kullanılması gerekmektedir. Ancak, hesaplama ağında çok ufak hücrelerin kullanılması hesaplama zamanını ve bellek gereksinimini önemli ölçüde arttırmaktadır. Bazen, bir problemin çözümü için gerekli hesaplama zamanı kabul edilemez seviyelere ulaşabilmektedir. Bu duruma alternatif olarak, küçük hücreler tüm hesaplama alanı yerine sadece gerektiği yerde kullanılabilmektedir. Bu durumda, daha küçük hücrelerin kullanılacağı kritik alanların belirlenmesi gerekmektedir. Bu yöntem adaptasyon olarak adlandırılmaktadır.

Kartezyen yöntemlerin en önemli özelliklerinden biri herhangi bir geometriye değişik adaptasyon çeşitlerinin rahatlıkla uygulanabilmesidir. Bu nedenle de oldukça düşük sayıda hücre kullanarak, hassas sonuçlar elde edilebilmektedir. Bu da hesaplama zamanında ve bellek gereksiniminde önemli kazanımlar sağlamaktadır.

Geometrik ve çözüm adaptasyonu olmak üzere iki tip adaptasyon vardır. Geometrinin tanımlanmasından sonra bu adaptasyonlar kullanılarak problemin çözümü için uygun hesaplama ağ veya ağları elde edilmektedir.

Son olarak, üç boyutlu Kartezyen ağın üretilmesi, iki boyutlu Kartezyen ağın üretimine çok benzemektedir. Burada sadece dörtlü ağaç veri yapısı yerine, hücrelerin çocuk sayısı sekize yükseldiğinden sekizli ağaç veri yapısı kullanılmaktadır. Dörtlü ağaç veri yapısında olduğu gibi sekizli ağaç veri yapısı ile de komşuluk ve aile ilişkilerinin depolanmasının yanı sıra akı değerlerinin, hücre tiplerinin ve merkez koordinatlarının depolanması da sağlanmaktadır.

3.2 SEKİZLİ AĞAÇ VERİ YAPISI

Üç boyutlu problemler için geliştirilen yazılımda, hücreler arasındaki komşuluk ve akrabalık ilişkilerini depolamak için sekizli ağaç veri yapısı kullanılmaktadır. Tıpkı iki boyutlu hücrelerde olduğu gibi üç boyutlu Kartezyen ağda da, komşuları dış sınırlar olan en büyük kübe kök hücre denilmektedir. Sekizli

ağaç veri yapısında her hücre 15 işaretçi ile tanımlanmaktadır. Bunlardan biri ebeveyni, sekizi çocukları ve son altı tanesi de yüzey komşularıdır. Aslında üç boyutlu bir hücrenin maksimum toplam komşu sayısı 26'dır. Bu 26 komşunun hepsini depolamaktansa en çok kullanılacak olan 6 yüzey komşusunu depolayıp geri kalan 20 komşusunu (kenar ve köşe komşuları) gerektiği zaman belirlemek daha akıllıca bir çözümdür. Her hücre için depolanan toplam işaretçiler aşağıda verilmektedir.

- 1 işaretçi: Hücrenin ebeveyni
- 8 işaretçi: Hücrenin çocukları
- 6 işaretçi: Hücrenin yüzey komşuları
- 3 işaretçi: Hücre merkezinin *x*, *y* ve *z* kordinatları
- 1 işaretçi: Hücrenin seviyesi
- 1 işaretçi: Çoklu ağ yönteminde anlatıacak hesaplama hücrelerini tanımlayan "compcell" olarak adlandırılmış değer
- 1 işaretçi: Çoklu ağ yönteminde anlatıacak "perform" olarak adlandırılmış değer

Kök hücrenin karşılıklı kenarlarının orta noktalarında bölünmesi ile sekiz küçük hücre meydana gelmektedir. Bu hücreler kök hücrenin çocukları olup kök hücrede oluşan sekiz çocuğun ebeveyn hücresi olarak atanmaktadır. Ayrıca her hücreye ait bir seviye sayısı bulunmaktadır. Kök hücrenin seviyesi sıfırdır ve her çocuğun seviyesi, ebeveyninin seviyesinin bir fazlasıdır. Yani kök hücrenin çocuklarının seviyesi bir, onların da çocuklarının seviyesi ikidir. Seviyesi atanan çocukların merkez koordinatları da, ebeveyninin merkez koordinatı, tüm etki alanının boyutu ve hücrenin seviyesi kullanılarak elde edilmektedir. Aşağıda verilen denklemler sırasıyla 1. çocuktan 8. çocuğa kadar olan çocuk hücrelerin merkez koordinatlarını veren denklemlerdir. Aşağıdaki denklemlerde *L* alan boyunu, *c* indisi ise hücrenin merkezini göstermektedir.

$$x_c = (x_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \tag{3.1}$$

$$x_c = (x_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \tag{3.2}$$

$$x_c = (x_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \tag{3.3}$$

$$x_c = (x_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \tag{3.4}$$

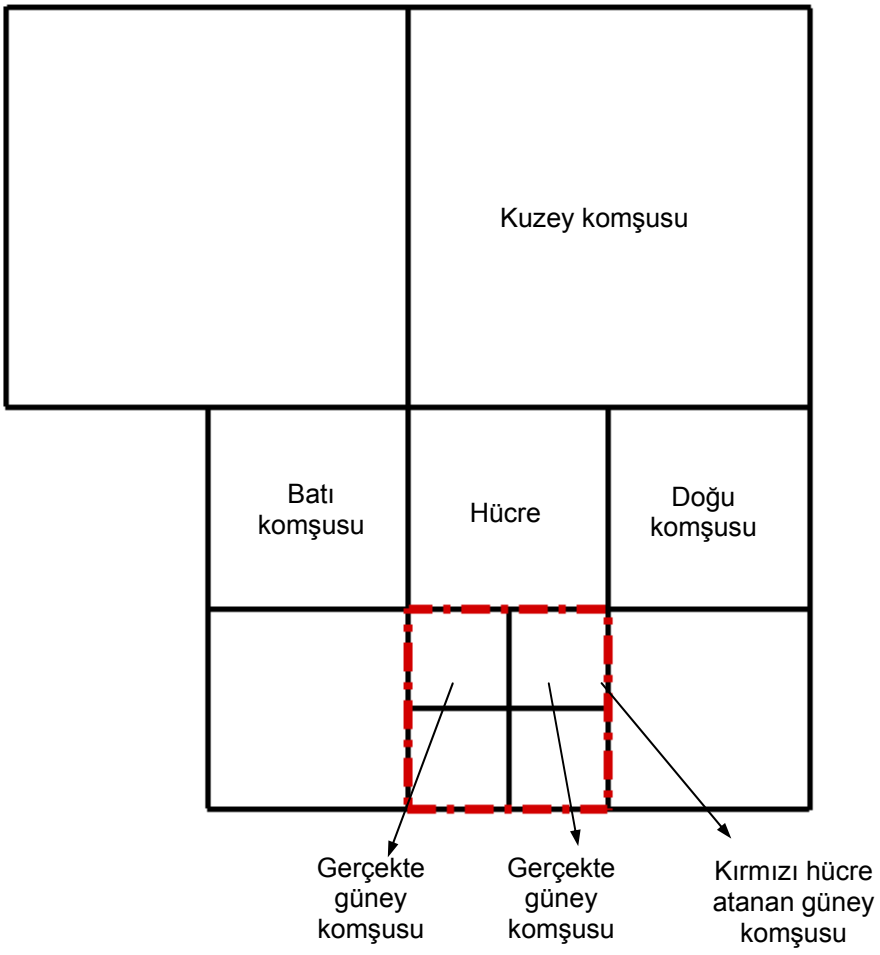
$$x_c = (x_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \tag{3.5}$$

$$x_c = (x_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \tag{3.6}$$

$$x_c = (x_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \tag{3.7}$$

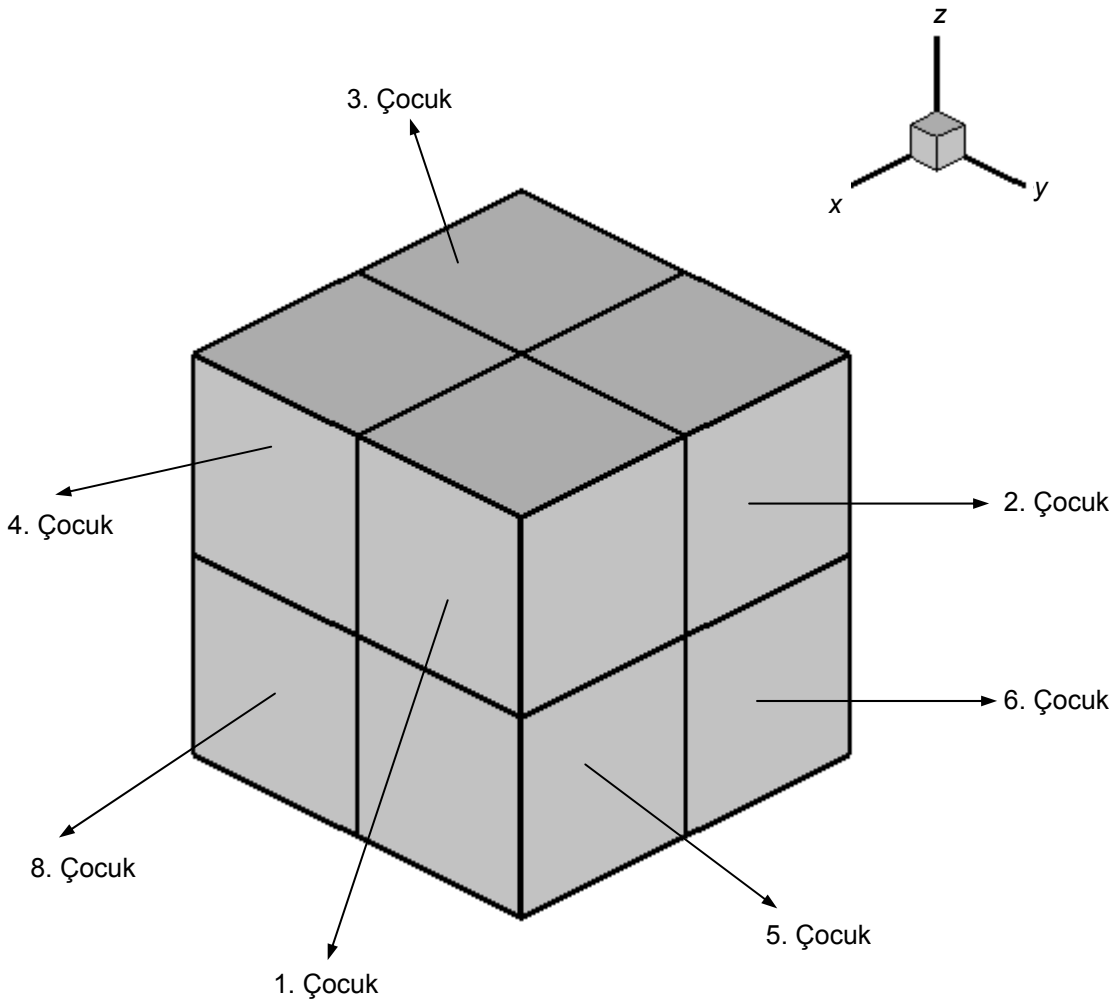
$$x_c = (x_c)_{ebeveyn} + \frac{L}{2^{(n+1)}} \quad y_c = (y_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \quad z_c = (z_c)_{ebeveyn} - \frac{L}{2^{(n+1)}} \tag{3.8}$$

Kartezyen ağı oluşturan hücreler içerisinde çocukları olmayan hücreler yaprak hücre olarak adlandırılmaktadırlar. Ayrıca, bu hücrelerde akı hesaplamaları yapıldığından hesaplama hücreleri de denilmektedir. Akı hesaplamalarının ve hücre içinde gradyan hesaplamalarının yapılabilmesi için hücrelerin komşu hücrelerinin de bilinmesi gereklidir. Her hücrenin doğu, batı, kuzey, güney, alt ve üst olmak üzere altı komşusu bulunmaktadır. Eğer komşusu, ele alınan hücre ile aynı seviyede veya bir seviye düşük ise o hücre, komşu hücre olarak atanmaktadır. Fakat bir seviye fazla ise, ele alınan hücrenin gerçekte iki komşu hücresi vardır. Ancak, bu iki hücreyi, komşu hücre olarak atamak yerine, bu hücrelerin ebeveyni komşu hücre olarak atanır. Şekil 3.1, iki boyutlu ağ için bu duruma bir örnektir.



Şekil 3.1 İki boyutlu ağ için komşular

Şekil 3.2'de görüldüğü gibi üç boyutlu ağda ele alınan her hücre, ebeveyninin çocuğu olduğu için üç komşusu otomatik olarak bilinmektedir. Diğer üç komşusu ebeveyninin komşusu ile ilişkilidir. Bu şekilde, bütün hücreler için aynı işlemler yapılmaktadır. Komşu arayışı, sadece birinci çocuk hücre için detaylı bir şekilde açıklanacaktır, diğer çocuklar için ise işlem aynıdır. İlk çocuk için, ilk grup komşuları, otomatik olarak bilinen, batı, güney ve alt komşularıdır. Bu komşular, ebeveyn hücrenin ikinci, dördüncü ve beşinci çocuklarıdır ve bu çocuklar aynı seviyededirler. İkinci grup komşular, ebeveyn hücrenin komşuları ile ilişkilidir ve ebeveyn hücrenin komşusunun çocukları değerlendirilmelidir. Birinci çocuk hücre için, bunlar kesinlikle ebeveynin doğu, kuzey ve üst komşularıdır. Bir seviye kuralına göre ebeveynin komşuları için üç seçenek vardır.



Şekil 3.2 Üç boyutlu ağ için komşular

(i) Birinci seçenek, ebeveynin doğu komşusunun hücreden bir seviye düşük yaprak hücre olduğu durumdur. Bu durumda, hücrenin doğu komşusu doğrudan ebeveynin doğu komşusudur.

(ii) İkinci seçenek, ebeveynin doğu komşusunun hücre ile aynı seviyede yaprak çocuk hücrelere sahip olduğu durumdur. Bu durumda, doğu komşusu, ebeveyn hücrenin doğu komşusunun ikinci çocuğudur.

(iii) Üçüncü seçenek ise ebeveynin doğu komşusunun çocuklarının yaprak çocuk hücrelere (bu yaprak çocuk hücrelerin seviyeleri, hücrenin seviyesinden bir seviye yüksektir) sahip olduğu durumdur. Bu durumda, dört gerçek doğu komşusu bulunmaktadır. Bunlar ebeveyn hücrenin doğu komşusunun ikinci çocuğunun ikinci, üçüncü, beşinci ve yedinci çocuklarıdır. Veri yapısını karmaşık hale getireceğinden, doğu komşularının ebeveynlerinden ayrı hafızaya alınması istenilmemektedir. Bu nedenle, ebeveynin doğu komşusunun ikinci çeyrekteki çocuğu, hücrenin doğu komşusu olarak hafızaya alınmaktadır. Bu sayede, komşu olarak hafızaya alınan hücrenin seviyesi ya hücreden bir seviye düşük yada aynı seviyede olmaktadır.

Aynı sebeplerle, birinci çocuğun kuzey komşusu, eğer ebeveyninin kuzey komşusu yaprak hücreyse, ebeveyninin kuzey komşusudur. Eğer ebeveynin kuzey komşusunun çocukları varsa, komşunun dördüncü çocuğu, onunda çocuklarının olup olmadığına bakılmaksızın hücrenin kuzey komşusu olarak hafızaya alınmaktadır. Son olarak, birinci çocuğun üst komşusu, eğer ebeveyninin üst komşusu yaprak hücreyse, ebeveyninin üst komşusudur. Eğer ebeveynin üst komşusunun çocukları varsa, komşunun beşinci çocuğu, onun da çocuklarının olup olmadığına bakılmaksızın hücrenin üst komşusu olarak hafızaya alınmaktadır. Herhangi bir çeyrekteki çocuğun her bir komşusunu belirlemek için kurallar görüldüğü gibi belirli bir düzen içerisinde planlanmıştır.

Üç boyutlu yaprak hücreler üç çeşittir. Bunlar, iç, dış ve kesik hücrelerdir. Üç boyutlu ağ yaratma iki boyutluya göre çok daha fazla karmaşık olduğundan ayrık hücrelerden kaçınılmıştır. Üç boyutlu ağda, ayrık hücrelerle karşılaşıldığında bu hücreler düzenli hücreler (iç, dış veya kesik hücreler) elde edilinceye kadar bölünürler. Yaprak hücreler için depolanan işaretçilerde aşağıda verilmektedir.

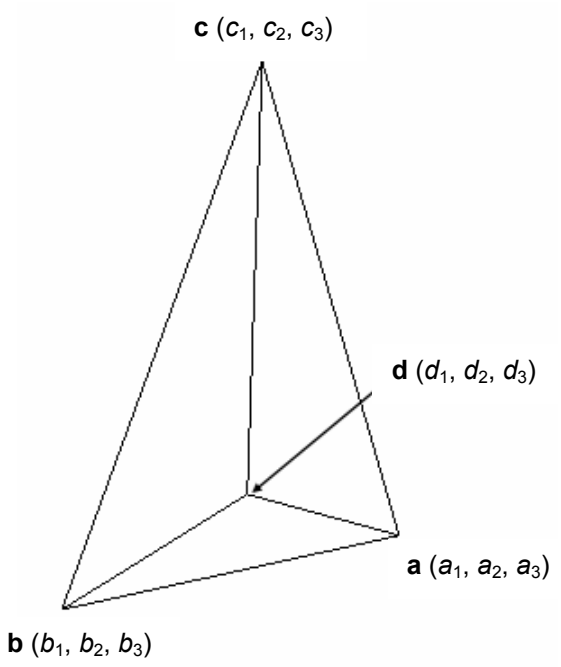
- 5 işaretçi: Hücrenin korunabilir değişkenleri
- 8 işaretçi: Hücrenin köşeleri
- 1 işaretçi: Hücrenin çeşidi
- 1 işaretçi: Hücrenin toplam küp endeksi (total cube index)
- 2 işaretçi: Yoğunlaştırma ve seyretme kriterleri
- 3 işaretçi: Hız vektörünün gradyanı, dönümü ve entropi dalgasının gücü (strength of the entropy wave)
- 5 işaretçi: Zorlayıcı fonksiyonlar (Forcing Functions)

Kesik hücreler için depolanan işaretçiler de aşağıda verilmektedir.

- 3 işaretçi: Hücrenin merkezinin *x*, *y* ve *z* kordinatları

- 9 işaretçi: Hücrenin kesik yüzeylerini oluşturan üçgenlerin köşe noktalarının *x*, *y* ve *z* kordinatları

İki boyutlu kesik hücrelerin alanları ve merkez kordinatları üçgenlere bölme işlemi ile hesaplanmaktadır. Üç boyutlu kesik hücrelerin hacimleri ve merkez kordinatları da dört yüzlü şekillere (tetrahedrons) bölme işlemi ile hesaplanmaktadır. Eğer Şekil 3.3'teki gibi bir dört yüzlü şeklin dört köşesinin kordinatları bilinirse, hacmi aşağıdaki denklemler yardımıyla bulunur.



Şekil 3.3 Dört yüzlü şekil ve onun köşe noktaları

$$\forall = \left|\vec{\mathbf{A}} \bullet (\vec{\mathbf{B}} \times \vec{\mathbf{C}})\right| \tag{3.9}$$

$$\vec{\mathbf{A}} = (a_1 - d_1)\vec{\mathbf{i}} + (a_2 - d_2)\vec{\mathbf{j}} + (a_3 - d_3)\vec{\mathbf{k}} \tag{3.10}$$

$$\vec{\mathbf{B}} = (b_1 - d_1)\vec{\mathbf{i}} + (b_2 - d_2)\vec{\mathbf{j}} + (b_3 - d_3)\vec{\mathbf{k}} \tag{3.11}$$

$$\vec{\mathbf{C}} = (c_1 - d_1)\vec{\mathbf{i}} + (c_2 - d_2)\vec{\mathbf{j}} + (c_3 - d_3)\vec{\mathbf{k}} \tag{3.12}$$

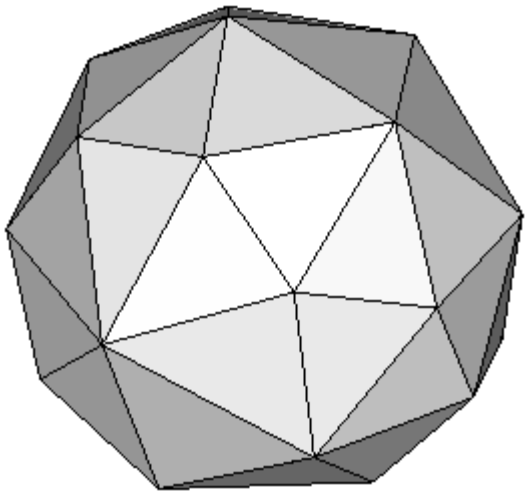
3.2 GEOMETRİNİN TANIMLANMASI VE GEOMETRİK ADAPTASYON

3.2.1 Geometrinin Tanımlanması

Kartezyen yöntemlerde tüm işlem geometrinin ve akış alanının tanımlanmasıyla başlamaktadır. Daha sonra, bu alan üzerinde uygun bir hesaplama ağı oluşturularak problem bu ağ üzerinde çözülmektedir.

Gambit yazılımında girdi geometrisi için üçgenlerden oluşan bir yüzey ağı ve eğer gerekirse dört yüzlü şekillerden (tetrahedron) oluşan bir hacim ağı yaratılmaktadır. Bu ağı oluşturan düğüm noktalarının koordinatları ve bu koordinatlar ile oluşturulan üçgenlerin ve dört yüzlü şekillerin geliştirilen yazılıma aktarılması gerekmektedir. Bu yüzden Gambit'te yaratılan ağ, çözücü olarak POLYFLOW seçildikten sonra hesaplama ağının yazılımdan ".neu" uzantılı çıktısı alınır. Gambit'te yaratılan yüzey ağına ve bu ağlar için alınan çıktıya bir örnek sırasıyla Şekil 3.4 ve Tablo 3.1'de görülmektedir.

Tablo 3.1'de kırmızı renkle gösterilen sayı toplam düğüm noktası sayısını, yeşil renkle gösterilen sayı ise toplam yüzey ağını oluşturan üçgenlerin sayısını belirtmektedir. Ayrıca italik olarak yazılan sayılar her bir düğüm noktasının *x*, *y* ve *z* koordinatlarıdır. Koyu yazılan sayılar ise, her bir üçgeni oluşturan düğüm noktalarını gösteren sayılardır. Geometrinin tanımlanmasından sonra problemin çözümü için gerekli hesaplama ağının oluşturulması için geometrik adaptasyon uygulanmaktadır.



Şekil 3.4 Gambit'te yaratılan üçgenlerden oluşan yüzey ağı

Tablo 3.1 Gambit'ten alınan çıktı

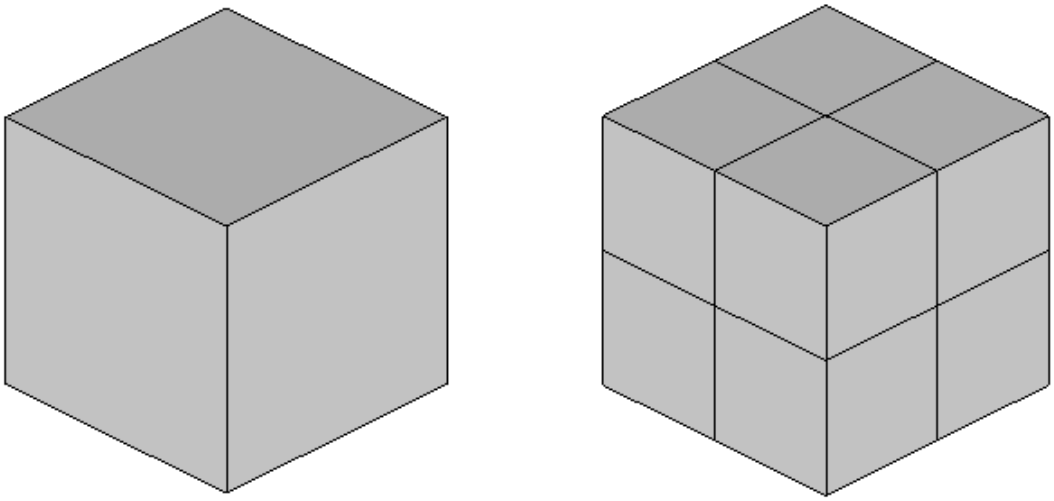
```
        CONTROL INFO 2.4.6
** GAMBIT NEUTRAL FILE
sphere
PROGRAM:                Gambit     VERSION:  2.4.6
 Jun 2009
  NUMNP     NELEM    NGRPS     NBSETS      NDFCD      NDFVL
     25        46        1          0          2          3

ENDOFSECTION
   NODAL COORDINATES 2.4.6
         1 -1.98288972275e+000  2.61052384440e-001  0.00000000000e+000
         2 -1.45577697798e+000  1.26273488645e-001  1.36556522965e+000
         3 -1.22755301316e+000  1.51559232680e+000  4.42824455988e-001
         4  1.66467887690e+000 -7.90424948610e-001 -7.77221099450e-001
         5 -1.51263359096e-001  1.22057996444e+000  1.57711887523e+000
         6  8.33244842460e-002 -3.52562841708e-001  1.96691547174e+000
         7 -8.31638007524e-001 -1.26804513721e+000  1.30400910826e+000
         8 -1.63860008609e+000 -1.14545623316e+000  5.40349496504e-002
         9 -1.39349606421e+000  1.12940622517e+000 -8.84652642326e-001
        10 -1.45570883917e+000 -2.15792594965e-001 -1.35438005431e+000
        11  1.39315797478e-001  1.96367326596e+000  3.52956673159e-001
        12 -3.82949059967e-001  1.83415696214e+000 -6.99441388313e-001
        13  1.19730956832e+000  4.97101249450e-001  1.52293799788e+000
        14  1.17281148967e+000  1.45628681471e+000  7.09747788310e-001
        15  4.09300059745e-001 -1.51163265674e+000  1.24396148340e+000
        16  1.36943001094e+000 -6.86750019647e-001  1.28570442001e+000
        17 -2.05031944964e-001 -1.98846808698e+000  6.29012609109e-002
        18 -7.44205012438e-001 -1.46075315461e+000 -1.14558243735e+000
        19  1.98870882714e+000  2.01972285618e-001  6.51490344266e-002
        20  1.02712841749e+000  1.52787645233e+000 -7.81409470374e-001
        21 -2.39687456491e-001  8.96476377967e-001 -1.77168846780e+000
        22  1.33810796503e+000 -1.48025166283e+000  1.35359109823e-001
        23  7.18107022266e-001 -1.44912995140e+000 -1.17658178148e+000
        24 -7.38694124748e-002 -5.51126718628e-001 -1.92114618130e+000
        25  1.23662915548e+000  6.40886642735e-002 -1.57055435275e+000
ENDOFSECTION
ELEMENTS/CELLS 2.4.6
       1  3  3        1       2       3
       2  3  3        3       2       5
       3  3  3        5       2       6
       4  3  3        6       2       7
       5  3  3        7       2       8
       6  3  3        8       2       1
       7  3  3        1       3       9
       8  3  3        1       9      10
       9  3  3        1      10       8
      10  3  3        3       5      11
      11  3  3        3      11      12
      12  3  3        3      12       9
      13  3  3        5       6      13
      14  3  3        5      13      14
      15  3  3        5      14      11
      16  3  3        6       7      15
      17  3  3        6      15      16
      18  3  3        6      16      13
      19  3  3        7       8      17
```

Tablo 3.1 Gambit'ten alınan çıktı (devamı)

```
      20  3  3          7       17       15
      21  3  3          8       10       18
      22  3  3          8       18       17
      23  3  3         13       16       19
      24  3  3         13       19       14
      25  3  3         12       11       20
      26  3  3         20       11       14
      27  3  3          9       12       21
      28  3  3          9       21       10
      29  3  3         16       15       22
      30  3  3         22       15       17
      31  3  3         14       19       20
      32  3  3         16       22       19
      33  3  3         21       12       20
      34  3  3         22       17       23
      35  3  3         23       17       18
      36  3  3         18       10       24
      37  3  3         24       10       21
      38  3  3         23       18       24
      39  3  3         24       21       25
      40  3  3         25       21       20
      41  3  3         20       19       25
      42  3  3         19       22        4
      43  3  3         19        4       25
      44  3  3         22       23        4
      45  3  3          4       23       25
      46  3  3         23       24       25
ENDOFSECTION
```

3.2.2 Eşit Ağ Adaptasyonu

Eşit ağ adaptasyonunda, yaprak hücreler herhangi bir kriter olmadan bölünmektedir. Buradaki amaç, diğer adaptasyonlar uygulanmadan önce yeteri kadar küçük hücreler elde edilmesidir. Eşit ağ adaptasyonun her seviyesinde bütün hücreler sekize bölünmektedir. Küp şeklindeki bir çözüm alanına üç seviye eşit ağ adaptasyonunun uygulanması Şekil 3.5'te gösterilmiştir. Şekil 5a'da çözüm alanını oluşturan kök hücre görülmektedir. İlk eşit ağ adaptasyonundan sonra Şekil 5b'de görüldüğü gibi 8 hücre, ikinci eşit ağ adaptasyonundan sonra Şekil 5c'de görüldüğü gibi 64 hücre ve üçüncü eşit ağ adaptasyonundan sonra ise Şekil 5d'deki gibi 512 hücre oluşmaktadır. Eşit ağ adaptasyonunda, eğer seviye sayısı n ile gösterilirse toplam hücre sayısı 8^n şeklinde olmaktadır. Görüldüğü gibi eşit ağ adaptasyonunda toplam hücre sayısı oldukça hızlı bir şekilde artmaktadır.



(a) Kök hücre (b) Birinci seviye adaptasyon

(c) İkinci seviye adaptasyon (d) Üçüncü seviye adaptasyon

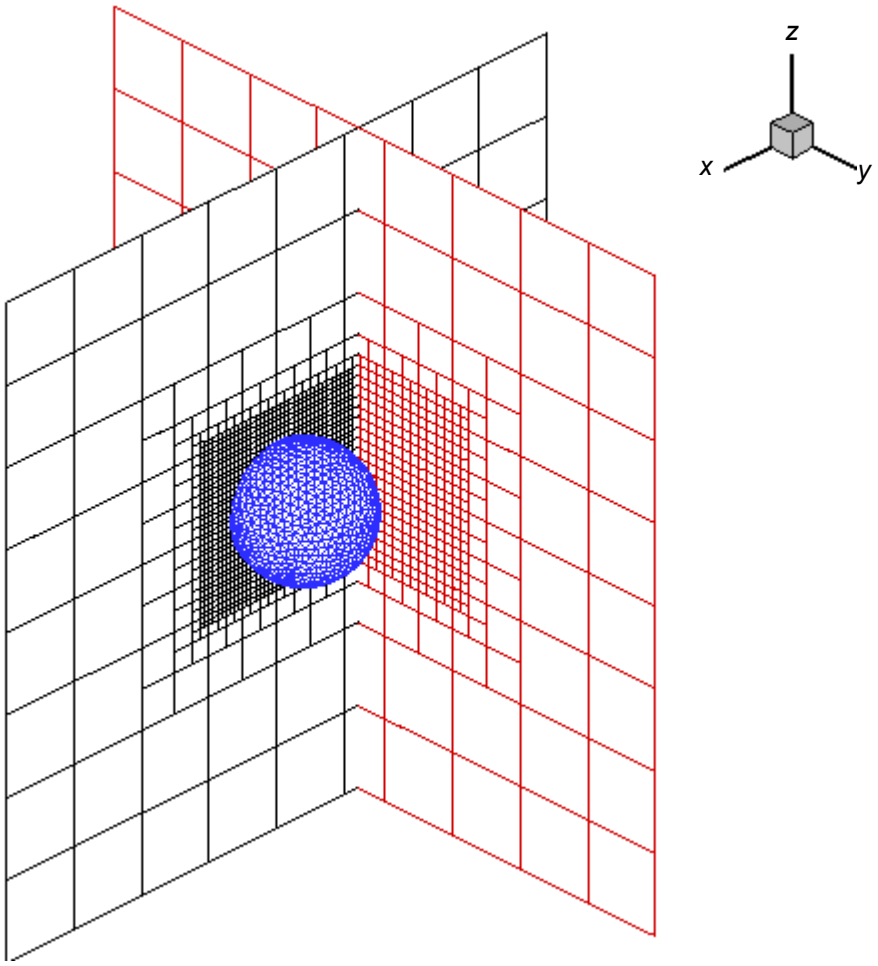
Şekil 3.5 Üç seviyeli eşit ağ adaptasyonu

3.2.3 Geometrik Adaptasyon

Geometrik adaptasyonda, ağ üretimi iki ana aşamada gerçekleştirilmektedir. İlk adım kutu adaptasyonu, ikinci adımsa kesik hücre adaptasyonudur.

3.2.3.1 Kutu Adaptasyonu

Eşit ağ adaptasyonundan sonra, geometriye yakın yerlerde istenen çözünürlükte ağ üretilmesini sağlamak için kutu adaptasyonu uygulanmaktadır. Akış alanı içerisinde bulunan cisim hayali bir kutu içerisine alınmaktadır. Geometrinin üç Kartezyen yönünde minimum ve maksimum koordinatlarının belirli bir katsayı ile çarpılmasından elde edilen sayılar hayali kutunun boyutlarını belirlemektedir. Bu kutu ile temas halinde olan yaprak hücreler veya tamamen kutu içerisinde kalan yaprak hücreler, geometri için önceden belirlenen yoğunlaştırma kriterini sağlayıncaya kadar yoğunlaştırılmaktadır. Böylelikle, cisim etrafında daha küçük hücrelerin elde edilmesi mümkün olmaktadır. Kutu adaptasyonu sonucunda elde edilen bir ağdan *xz* ve *yz* yüzeylerinde kesitler alınmıştır. Bu kesitler Şekil 3.6'da gösterilmektedir.



Şekil 3.6 Kutu adaptasyonu sonucunda elde edilen hesaplama ağından kesitler

Kutu adaptasyonu sırasında, eğer bir hücre yoğunlaştırılacaksa (refinement), ilk olarak komşuların seviyesi (level of the neighbors) incelenmelidir. Eğer komşusu bir seviye düşükse, hücre yoğunlaştırılmadan önce o komşu yoğunlaştırılmaktadır. Bu yolla, bir seviye kuralı (one level rule), ağ üretimi işleminde korunmuş olmaktadır. Bir seviye kuralı, iki komşu hücre seviyelerinin arasındaki farkın biri geçmemesidir. Bu kural sayesinde düzgün bir ağ elde edilmektedir. Aynı zamanda, bu kural akı hesaplamalarını da kolaylaştırmaktadır. Bir hücre sekiz eşit parçaya bölünerek yoğunlaştırılmaktadır. Hücrenin bu sekiz eşit parçası, o hücrenin çocukları olarak tanımlanmaktadır.

Kutu adaptasyonundan sonraki adım kesik hücre adaptasyonudur. Fakat kesik hücre adaptasyonundan önce tanımlanan geometrinin yüzeyleri tarafından kesilen kesik hücrelerin kesik kenarları belirlenmelidir. Geliştirilen kodda kesik hücrelerin kesilen yüzeyleri, küplerle ilerleme yöntemi (marching cubes algorithm) kullanılarak kolaylıkla bulunmaktadır. İki boyutlu problemlerde olduğu gibi yine her yaprak hücrenin bir toplam küp endeksi vardır. Bu endeks ve aşağıda verilen tablo (Tablo 3.2) yardımıyla yaprak hücrenin kesik olan kenarları kolaylıkla tespit edilir. Ama öncelikle üç boyutlu bir hücrenin kenarlarının ve köşelerinin nasıl numaralandırıldığını belirtmek gerekir. Numaralandırma Şekil 3.7'de görülmektedir.

Tablo 3.2 Küplerle ilerleme yöntemi için gerekli olan tablo

```
int triangleTable[256][16] =
{{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 8, 3, 9, 8, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2,10, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 1, 2, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 2, 10, 0, 2, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 8, 3, 2, 10, 8, 10, 9, 8, -1, -1, -1, -1, -1, -1, -1},
{3, 11, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 11, 2, 8, 11, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 11, 2, 1, 9, 11, 9, 8, 11, -1, -1, -1, -1, -1, -1, -1},
{3, 10, 1, 11, 10, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 10, 1, 0, 8, 10, 8, 11, 10, -1, -1, -1, -1, -1, -1, -1},
{3, 9, 0, 3, 11, 9, 11, 10, 9, -1, -1, -1, -1, -1, -1, -1},
{9, 8, 10, 10, 8, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 8, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 3, 0, 7, 3, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 1, 9, 4, 7, 1, 7, 3, 1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 4, 7, 3, 0, 4, 1, 2, 10, -1, -1, -1, -1, -1, -1, -1},
{9, 2, 10, 9, 0, 2, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1},
{2, 10, 9, 2, 9, 7, 2, 7, 3, 7, 9, 4, -1, -1, -1, -1},
{8, 4, 7, 3, 11, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 4, 7, 11, 2, 4, 2, 0, 4, -1, -1, -1, -1, -1, -1, -1},
{9, 0, 1, 8, 4, 7, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1},
```

Tablo 3.2 Küplerle ilerleme yöntemi için gerekli olan tablo (devamı)

```
{4, 7, 11, 9, 4, 11, 9, 11, 2, 9, 2, 1, -1, -1, -1, -1},
{3, 10, 1, 3, 11, 10, 7, 8, 4, -1, -1, -1, -1, -1, -1, -1},
{1, 11, 10, 1, 4, 11, 1, 0, 4, 7, 11, 4, -1, -1, -1, -1},
{4, 7, 8, 9, 0, 11, 9, 11, 10, 11, 0, 3, -1, -1, -1, -1},
{4, 7, 11, 4, 11, 9, 9, 11, 10, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 5, 4, 0, 8, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 5, 4, 1, 5, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 5, 4, 8, 3, 5, 3, 1, 5, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 9, 5, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 1, 2, 10, 4, 9, 5, -1, -1, -1, -1, -1, -1, -1},
{5, 2, 10, 5, 4, 2, 4, 0, 2, -1, -1, -1, -1, -1, -1, -1},
{2, 10, 5, 3, 2, 5, 3, 5, 4, 3, 4, 8, -1, -1, -1, -1},
{9, 5, 4, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 11, 2, 0, 8, 11, 4, 9, 5, -1, -1, -1, -1, -1, -1, -1},
{0, 5, 4, 0, 1, 5, 2, 3, 11, -1, -1, -1, -1, -1, -1, -1},
{2, 1, 5, 2, 5, 8, 2, 8, 11, 4, 8, 5, -1, -1, -1, -1},
{10, 3, 11, 10, 1, 3, 9, 5, 4, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 5, 0, 8, 1, 8, 10, 1, 8, 11, 10, -1, -1, -1, -1},
{5, 4, 0, 5, 0, 11, 5, 11, 10, 11, 0, 3, -1, -1, -1, -1},
{5, 4, 8, 5, 8, 10, 10, 8, 11, -1, -1, -1, -1, -1, -1, -1},
{9, 7, 8, 5, 7, 9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 3, 0, 9, 5, 3, 5, 7, 3, -1, -1, -1, -1, -1, -1, -1},
{0, 7, 8, 0, 1, 7, 1, 5, 7, -1, -1, -1, -1, -1, -1, -1},
{1, 5, 3, 3, 5, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 7, 8, 9, 5, 7, 10, 1, 2, -1, -1, -1, -1, -1, -1, -1},
{10, 1, 2, 9, 5, 0, 5, 3, 0, 5, 7, 3, -1, -1, -1, -1},
{8, 0, 2, 8, 2, 5, 8, 5, 7, 10, 5, 2, -1, -1, -1, -1},
{2, 10, 5, 2, 5, 3, 3, 5, 7, -1, -1, -1, -1, -1, -1, -1},
{7, 9, 5, 7, 8, 9, 3, 11, 2, -1, -1, -1, -1, -1, -1, -1},
{9, 5, 7, 9, 7, 2, 9, 2, 0, 2, 7, 11, -1, -1, -1, -1},
{2, 3, 11, 0, 1, 8, 1, 7, 8, 1, 5, 7, -1, -1, -1, -1},
{11, 2, 1, 11, 1, 7, 7, 1, 5, -1, -1, -1, -1, -1, -1, -1},
{9, 5, 8, 8, 5, 7, 10, 1, 3, 10, 3, 11, -1, -1, -1, -1},
{5, 7, 0, 5, 0, 9, 7, 11, 0, 1, 0, 10, 11, 10, 0, -1},
{11, 10, 0, 11, 0, 3, 10, 5, 0, 8, 0, 7, 5, 7, 0, -1},
{11, 10, 5, 7, 11, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{ 5, 10, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 0, 1, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 8, 3, 1, 9, 8, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1},
{1, 6, 5, 2, 6, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 6, 5, 1, 2, 6, 3, 0, 8, -1, -1, -1, -1, -1, -1, -1},
{9, 6, 5, 9, 0, 6, 0, 2, 6, -1, -1, -1, -1, -1, -1, -1},
{5, 9, 8, 5, 8, 2, 5, 2, 6, 3, 2, 8, -1, -1, -1, -1},
{2, 3, 11, 10, 6, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 0, 8, 11, 2, 0, 10, 6, 5, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 2, 3, 11, 5, 10, 6, -1, -1, -1, -1, -1, -1, -1},
{5, 10, 6, 1, 9, 2, 9, 11, 2, 9, 8, 11, -1, -1, -1, -1},
{6, 3, 11, 6, 5, 3, 5, 1, 3, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 11, 0, 11, 5, 0, 5, 1, 5, 11, 6, -1, -1, -1, -1},
{3, 11, 6, 0, 3, 6, 0, 6, 5, 0, 5, 9, -1, -1, -1, -1},
{6, 5, 9, 6, 9, 11, 11, 9, 8, -1, -1, -1, -1, -1, -1, -1},
```

Tablo 3.2 Küplerle ilerleme yöntemi için gerekli olan tablo (devamı)

```
{5, 10, 6, 4, 7, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 3, 0, 4, 7, 3, 6, 5, 10, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 5, 10, 6, 8, 4, 7, -1, -1, -1, -1, -1, -1, -1},
{10, 6, 5, 1, 9, 7, 1, 7, 3, 7, 9, 4, -1, -1, -1, -1},
{6, 1, 2, 6, 5, 1, 4, 7, 8, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 5, 5, 2, 6, 3, 0, 4, 3, 4, 7, -1, -1, -1, -1},
{8, 4, 7, 9, 0, 5, 0, 6, 5, 0, 2, 6, -1, -1, -1, -1},
{7, 3, 9, 7, 9, 4, 3, 2, 9, 5, 9, 6, 2, 6, 9, -1},
{3, 11, 2, 7, 8, 4, 10, 6, 5, -1, -1, -1, -1, -1, -1, -1},
{5, 10, 6, 4, 7, 2, 4, 2, 0, 2, 7, 11, -1, -1, -1, -1},
{0, 1, 9, 4, 7, 8, 2, 3, 11, 5, 10, 6, -1, -1, -1, -1},
{9, 2, 1, 9, 11, 2, 9, 4, 11, 7, 11, 4, 5, 10, 6, -1},
{8, 4, 7, 3, 11, 5, 3, 5, 1, 5, 11, 6, -1, -1, -1, -1},
{5, 1, 11, 5, 11, 6, 1, 0, 11, 7, 11, 4, 0, 4, 11, -1},
{0, 5, 9, 0, 6, 5, 0, 3, 6, 11, 6, 3, 8, 4, 7, -1},
{6, 5, 9, 6, 9, 11, 4, 7, 9, 7, 11, 9, -1, -1, -1, -1},
{10, 4, 9, 6, 4, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 10, 6, 4, 9, 10, 0, 8, 3, -1, -1, -1, -1, -1, -1, -1},
{10, 0, 1, 10, 6, 0, 6, 4, 0, -1, -1, -1, -1, -1, -1, -1},
{8, 3, 1, 8, 1, 6, 8, 6, 4, 6, 1, 10, -1, -1, -1, -1},
{1, 4, 9, 1, 2, 4, 2, 6, 4, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 1, 2, 9, 2, 4, 9, 2, 6, 4, -1, -1, -1, -1},
{0, 2, 4, 4, 2, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 3, 2, 8, 2, 4, 4, 2, 6, -1, -1, -1, -1, -1, -1, -1},
{10, 4, 9, 10, 6, 4, 11, 2, 3, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 2, 2, 8, 11, 4, 9, 10, 4, 10, 6, -1, -1, -1, -1},
{3, 11, 2, 0, 1, 6, 0, 6, 4, 6, 1, 10, -1, -1, -1, -1},
{6, 4, 1, 6, 1, 10, 4, 8, 1, 2, 1, 11, 8, 11, 1, -1},
{9, 6, 4, 9, 3, 6, 9, 1, 3, 11, 6, 3, -1, -1, -1, -1},
{8, 11, 1, 8, 1, 0, 11, 6, 1, 9, 1, 4, 6, 4, 1, -1},
{3, 11, 6, 3, 6, 0, 0, 6, 4, -1, -1, -1, -1, -1, -1, -1},
{6, 4, 8, 11, 6, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 10, 6, 7, 8, 10, 8, 9, 10, -1, -1, -1, -1, -1, -1, -1},
{0, 7, 3, 0, 10, 7, 0, 9, 10, 6, 7, 10, -1, -1, -1, -1},
{10, 6, 7, 1, 10, 7, 1, 7, 8, 1, 8, 0, -1, -1, -1, -1},
{10, 6, 7, 10, 7, 1, 1, 7, 3, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 6, 1, 6, 8, 1, 8, 9, 8, 6, 7, -1, -1, -1, -1},
{2, 6, 9, 2, 9, 1, 6, 7, 9, 0, 9, 3, 7, 3, 9, -1},
{7, 8, 0, 7, 0, 6, 6, 0, 2, -1, -1, -1, -1, -1, -1, -1},
{7, 3, 2, 6, 7, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 11, 10, 6, 8, 10, 8, 9, 8, 6, 7, -1, -1, -1, -1},
{2, 0, 7, 2, 7, 11, 0, 9, 7, 6, 7, 10, 9, 10, 7, -1},
{1, 8, 0, 1, 7, 8, 1, 10, 7, 6, 7, 10, 2, 3, 11, -1},
{11, 2, 1, 11, 1, 7, 10, 6, 1, 6, 7, 1, -1, -1, -1, -1},
{8, 9, 6, 8, 6, 7, 9, 1, 6, 11, 6, 3, 1, 3, 6, -1},
{0, 9, 1, 11, 6, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 8, 0, 7, 0, 6, 3, 11, 0, 11, 6, 0, -1, -1, -1, -1},
{7, 11, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{6, 11, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 8, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 9, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 1, 9, 8, 3, 1, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1},
{10, 1, 2, 6, 11, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
```

Tablo 3.2 Küplerle ilerleme yöntemi için gerekli olan tablo (devamı)

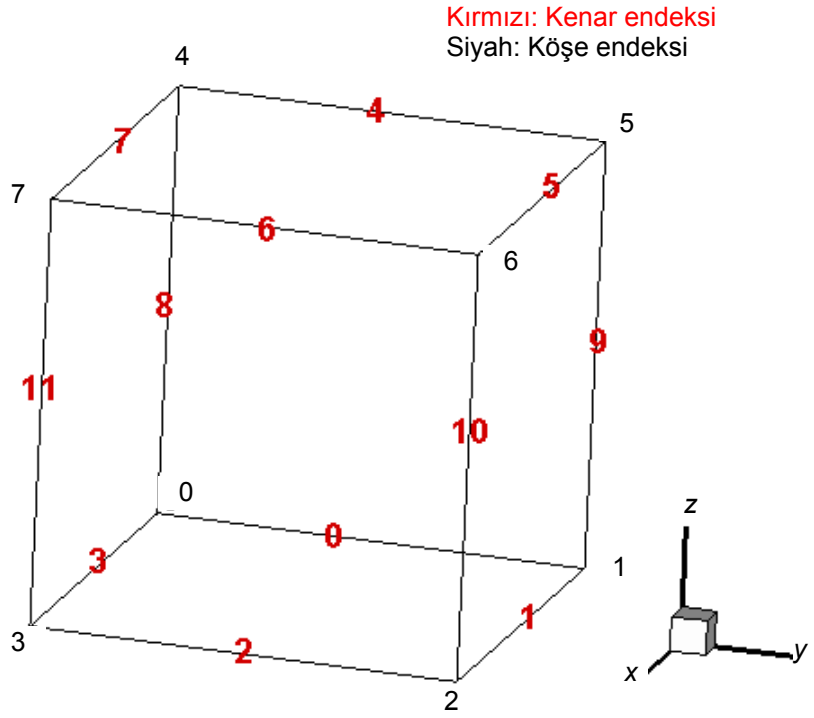
```
{1, 2, 10, 3, 0, 8, 6, 11, 7, -1, -1, -1, -1, -1, -1, -1},
{2, 9, 0, 2, 10, 9, 6, 11, 7, -1, -1, -1, -1, -1, -1, -1},
{6, 11, 7, 2, 10, 3, 10, 8, 3, 10, 9, 8, -1, -1, -1, -1},
{7, 2, 3, 6, 2, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{7, 0, 8, 7, 6, 0, 6, 2, 0, -1, -1, -1, -1, -1, -1, -1},
{2, 7, 6, 2, 3, 7, 0, 1, 9, -1, -1, -1, -1, -1, -1, -1},
{1, 6, 2, 1, 8, 6, 1, 9, 8, 8, 7, 6, -1, -1, -1, -1},
{10, 7, 6, 10, 1, 7, 1, 3, 7, -1, -1, -1, -1, -1, -1, -1},
{10, 7, 6, 1, 7, 10, 1, 8, 7, 1, 0, 8, -1, -1, -1, -1},
{0, 3, 7, 0, 7, 10, 0, 10, 9, 6, 10, 7, -1, -1, -1, -1},
{7, 6, 10, 7, 10, 8, 8, 10, 9, -1, -1, -1, -1, -1, -1, -1},
{6, 8, 4, 11, 8, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 6, 11, 3, 0, 6, 0, 4, 6, -1, -1, -1, -1, -1, -1, -1},
{8, 6, 11, 8, 4, 6, 9, 0, 1, -1, -1, -1, -1, -1, -1, -1},
{9, 4, 6, 9, 6, 3, 9, 3, 1, 11, 3, 6, -1, -1, -1, -1},
{6, 8, 4, 6, 11, 8, 2, 10, 1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 3, 0, 11, 0, 6, 11, 0, 4, 6, -1, -1, -1, -1},
{4, 11, 8, 4, 6, 11, 0, 2, 9, 2, 10, 9, -1, -1, -1, -1},
{10, 9, 3, 10, 3, 2, 9, 4, 3, 11, 3, 6, 4, 6, 3, -1},
{8, 2, 3, 8, 4, 2, 4, 6, 2, -1, -1, -1, -1, -1, -1, -1},
{0, 4, 2, 2, 4, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 9, 0, 2, 3, 4, 2, 4, 6, 4, 3, 8, -1, -1, -1, -1},
{1, 9, 4, 1, 4, 2, 2, 4, 6, -1, -1, -1, -1, -1, -1, -1},
{8, 1, 3, 8, 6, 1, 8, 4, 6, 6, 10, 1, -1, -1, -1, -1},
{10, 1, 0, 10, 0, 6, 6, 0, 4, -1, -1, -1, -1, -1, -1, -1},
{4, 6, 3, 4, 3, 8, 6, 10, 3, 0, 3, 9, 10, 9, 3, -1},
{10, 9, 4, 6, 10, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 5, 7, 6, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 4, 9, 5, 11, 7, 6, -1, -1, -1, -1, -1, -1, -1},
{5, 0, 1, 5, 4, 0, 7, 6, 11, -1, -1, -1, -1, -1, -1, -1},
{11, 7, 6, 8, 3, 4, 3, 5, 4, 3, 1, 5, -1, -1, -1, -1},
{9, 5, 4, 10, 1, 2, 7, 6, 11, -1, -1, -1, -1, -1, -1, -1},
{6, 11, 7, 1, 2, 10, 0, 8, 3, 4, 9, 5, -1, -1, -1, -1},
{7, 6, 11, 5, 4, 10, 4, 2, 10, 4, 0, 2, -1, -1, -1, -1},
{3, 4, 8, 3, 5, 4, 3, 2, 5, 10, 5, 2, 11, 7, 6, -1},
{7, 2, 3, 7, 6, 2, 5, 4, 9, -1, -1, -1, -1, -1, -1, -1},
{9, 5, 4, 0, 8, 6, 0, 6, 2, 6, 8, 7, -1, -1, -1, -1},
{3, 6, 2, 3, 7, 6, 1, 5, 0, 5, 4, 0, -1, -1, -1, -1},
{6, 2, 8, 6, 8, 7, 2, 1, 8, 4, 8, 5, 1, 5, 8, -1},
{9, 5, 4, 10, 1, 6, 1, 7, 6, 1, 3, 7, -1, -1, -1, -1},
{1, 6, 10, 1, 7, 6, 1, 0, 7, 8, 7, 0, 9, 5, 4, -1},
{4, 0, 10, 4, 10, 5, 0, 3, 10, 6, 10, 7, 3, 7, 10, -1},
{7, 6, 10, 7, 10, 8, 5, 4, 10, 4, 8, 10, -1, -1, -1, -1},
{6, 9, 5, 6, 11, 9, 11, 8, 9, -1, -1, -1, -1, -1, -1, -1},
{3, 6, 11, 0, 6, 3, 0, 5, 6, 0, 9, 5, -1, -1, -1, -1},
{0, 11, 8, 0, 5, 11, 0, 1, 5, 5, 6, 11, -1, -1, -1, -1},
{6, 11, 3, 6, 3, 5, 5, 3, 1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 10, 9, 5, 11, 9, 11, 8, 11, 5, 6, -1, -1, -1, -1},
{0, 11, 3, 0, 6, 11, 0, 9, 6, 5, 6, 9, 1, 2, 10, -1},
{11, 8, 5, 11, 5, 6, 8, 0, 5, 10, 5, 2, 0, 2, 5, -1},
{6, 11, 3, 6, 3, 5, 2, 10, 3, 10, 5, 3, -1, -1, -1, -1},
{5, 8, 9, 5, 2, 8, 5, 6, 2, 3, 8, 2, -1, -1, -1, -1},
{9, 5, 6, 9, 6, 0, 0, 6, 2, -1, -1, -1, -1, -1, -1, -1},
```

Tablo 3.2 Küplerle ilerleme yöntemi için gerekli olan tablo (devamı)

```
{1, 5, 8, 1, 8, 0, 5, 6, 8, 3, 8, 2, 6, 2, 8, -1},
{1, 5, 6, 2, 1, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 3, 6, 1, 6, 10, 3, 8, 6, 5, 6, 9, 8, 9, 6, -1},
{10, 1, 0, 10, 0, 6, 9, 5, 0, 5, 6, 0, -1, -1, -1, -1},
{0, 3, 8, 5, 6, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{10, 5, 6, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 5, 10, 7, 5, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{11, 5, 10, 11, 7, 5, 8, 3, 0, -1, -1, -1, -1, -1, -1, -1},
{5, 11, 7, 5, 10, 11, 1, 9, 0, -1, -1, -1, -1, -1, -1, -1},
{10, 7, 5, 10, 11, 7, 9, 8, 1, 8, 3, 1, -1, -1, -1, -1},
{11, 1, 2, 11, 7, 1, 7, 5, 1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 1, 2, 7, 1, 7, 5, 7, 2, 11, -1, -1, -1, -1},
{9, 7, 5, 9, 2, 7, 9, 0, 2, 2, 11, 7, -1, -1, -1, -1},
{7, 5, 2, 7, 2, 11, 5, 9, 2, 3, 2, 8, 9, 8, 2, -1},
{2, 5, 10, 2, 3, 5, 3, 7, 5, -1, -1, -1, -1, -1, -1, -1},
{8, 2, 0, 8, 5, 2, 8, 7, 5, 10, 2, 5, -1, -1, -1, -1},
{9, 0, 1, 5, 10, 3, 5, 3, 7, 3, 10, 2, -1, -1, -1, -1},
{9, 8, 2, 9, 2, 1, 8, 7, 2, 10, 2, 5, 7, 5, 2, -1},
{1, 3, 5, 5, 3, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 7, 0, 7, 1, 1, 7, 5, -1, -1, -1, -1, -1, -1, -1},
{9, 0, 3, 9, 3, 5, 5, 3, 7, -1, -1, -1, -1, -1, -1, -1},
{9, 8, 7, 5, 9, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{5, 8, 4, 5, 10, 8, 10, 11, 8, -1, -1, -1, -1, -1, -1, -1},
{5, 0, 4, 5, 11, 0, 5, 10, 11, 11, 3, 0, -1, -1, -1, -1},
{0, 1, 9, 8, 4, 10, 8, 10, 11, 10, 4, 5, -1, -1, -1, -1},
{10, 11, 4, 10, 4, 5, 11, 3, 4, 9, 4, 1, 3, 1, 4, -1},
{2, 5, 1, 2, 8, 5, 2, 11, 8, 4, 5, 8, -1, -1, -1, -1},
{0, 4, 11, 0, 11, 3, 4, 5, 11, 2, 11, 1, 5, 1, 11, -1},
{0, 2, 5, 0, 5, 9, 2, 11, 5, 4, 5, 8, 11, 8, 5, -1},
{9, 4, 5, 2, 11, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 5, 10, 3, 5, 2, 3, 4, 5, 3, 8, 4, -1, -1, -1, -1},
{5, 10, 2, 5, 2, 4, 4, 2, 0, -1, -1, -1, -1, -1, -1, -1},
{3, 10, 2, 3, 5, 10, 3, 8, 5, 4, 5, 8, 0, 1, 9, -1},
{5, 10, 2, 5, 2, 4, 1, 9, 2, 9, 4, 2, -1, -1, -1, -1},
{8, 4, 5, 8, 5, 3, 3, 5, 1, -1, -1, -1, -1, -1, -1, -1},
{0, 4, 5, 1, 0, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{8, 4, 5, 8, 5, 3, 9, 0, 5, 0, 3, 5, -1, -1, -1, -1},
{9, 4, 5, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 11, 7, 4, 9, 11, 9, 10, 11, -1, -1, -1, -1, -1, -1, -1},
{0, 8, 3, 4, 9, 7, 9, 11, 7, 9, 10, 11, -1, -1, -1, -1},
{1, 10, 11, 1, 11, 4, 1, 4, 0, 7, 4, 11, -1, -1, -1, -1},
{3, 1, 4, 3, 4, 8, 1, 10, 4, 7, 4, 11, 10, 11, 4, -1},
{4, 11, 7, 9, 11, 4, 9, 2, 11, 9, 1, 2, -1, -1, -1, -1},
{9, 7, 4, 9, 11, 7, 9, 1, 11, 2, 11, 1, 0, 8, 3, -1},
{11, 7, 4, 11, 4, 2, 2, 4, 0, -1, -1, -1, -1, -1, -1, -1},
{11, 7, 4, 11, 4, 2, 8, 3, 4, 3, 2, 4, -1, -1, -1, -1},
{2, 9, 10, 2, 7, 9, 2, 3, 7, 7, 4, 9, -1, -1, -1, -1},
{9, 10, 7, 9, 7, 4, 10, 2, 7, 8, 7, 0, 2, 0, 7, -1},
{3, 7, 10, 3, 10, 2, 7, 4, 10, 1, 10, 0, 4, 0, 10, -1},
{1, 10, 2, 8, 7, 4, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 1, 4, 1, 7, 7, 1, 3, -1, -1, -1, -1, -1, -1, -1},
{4, 9, 1, 4, 1, 7, 0, 8, 1, 8, 7, 1, -1, -1, -1, -1},
{4, 0, 3, 7, 4, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
```

```
{4, 8, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{9, 10, 8, 8, 10, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 9, 3, 9, 11, 11, 9, 10, -1, -1, -1, -1, -1, -1, -1},
{0, 1, 10, 0, 10, 8, 8, 10, 11, -1, -1, -1, -1, -1, -1, -1},
{3, 1, 10, 11, 3, 10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 2, 11, 1, 11, 9, 9, 11, 8, -1, -1, -1, -1, -1, -1, -1},
{3, 0, 9, 3, 9, 11, 1, 2, 9, 2, 11, 9, -1, -1, -1, -1},
{0, 2, 11, 8, 0, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{3, 2, 11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 8, 2, 8, 10, 10, 8, 9, -1, -1, -1, -1, -1, -1, -1},
{9, 10, 2, 0, 9, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{2, 3, 8, 2, 8, 10, 0, 1, 8, 1, 10, 8, -1, -1, -1, -1},
{1, 10, 2, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{1, 3, 8, 9, 1, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 9, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{0, 3, 8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},
{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}};
```

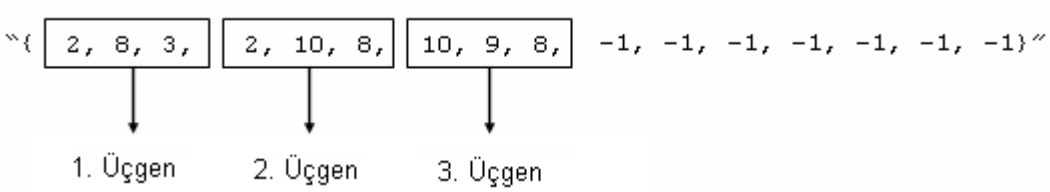
Karelerle ilerleme yönteminde olduğu gibi öncelikle üç boyutlu bir yaprak hücrenin toplam küp endeksi hesaplanır. Hesaplama işlemi kısaca şöyle özetlenebilir. Öncelikle, iç-dış testiyle 8 köşesi test edilmiş olan kesik yaprak hücrenin tanımlanan geometrinin içinde kalan köşelerinin ϕ değerleri -1 olarak atanır. Sonra her köşe için aşağıda verilmiş olan küp endeksleri kullanılarak, hücrenin toplam küp endeksi hesaplanır.



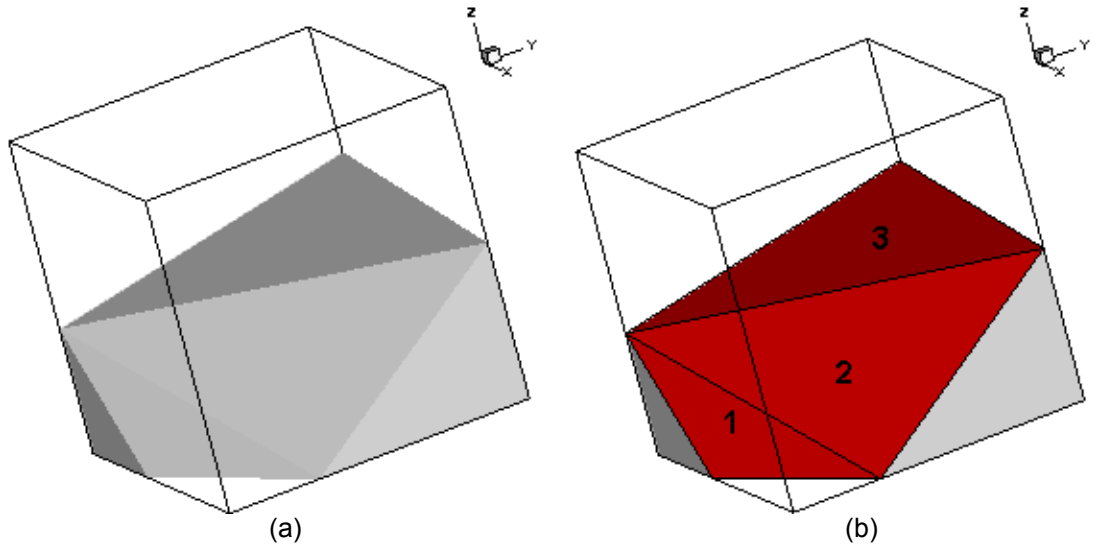
Şekil 3.7 Üç boyutlu bir hücrenin kenarlarının ve köşelerinin numaralandırılması

- Eğer 0. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 1
- Eğer 1. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 2
- Eğer 2. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 4
- Eğer 3. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 8
- Eğer 4. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 16
- Eğer 5. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 32
- Eğer 6. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 64
- Eğer 7. köşenin ϕ değeri = -1, bu köşenin küp endeksi= 128

Mesela 0., 1. ve 2. köşeleri tanımlanan geometrinin içinde kalan bir yaprak hücrenin toplam küp endeksi yedidir. Bu kesik hücrenin hangi kenarlarının kesik olduğu yukarıda verilen üçgen tablosu (triangleTable) ile bulunur. Bu hücrenin toplam küp endeksine tekabül eden sıra aşağıda verilmektedir.



Bu bilgiye göre, bu hücrenin kesik kenarları sırasıyla 2, 8, 3, 10 ve 9'dur. Kesik kenarlar bulunduktan sonra bu kenarlar üzerindeki kesim noktalarının *x*, *y* ve *z* kordinatları doğru-üçgen kesişimi yöntemi kullanılarak hesaplanıp hafızada saklanır. Kesim noktaları hesaplandıktan sonra bu noktalardan geçen üçgenler yukarıda gösterilen sırada çizilir. Diğer bir deyişle, kesik yüzeyi oluşturan köşeleri 2., 8. ve 3. kenarlardan geçen ilk üçgen, köşeleri 2., 10. ve 8. kenarlardan geçen ikinci üçgen, köşeleri 10., 9. ve 8. kenarlardan geçen üçüncü üçgen sırasıyla çizilir. Bu durum Şekil 3.8'te gösterilmektedir.



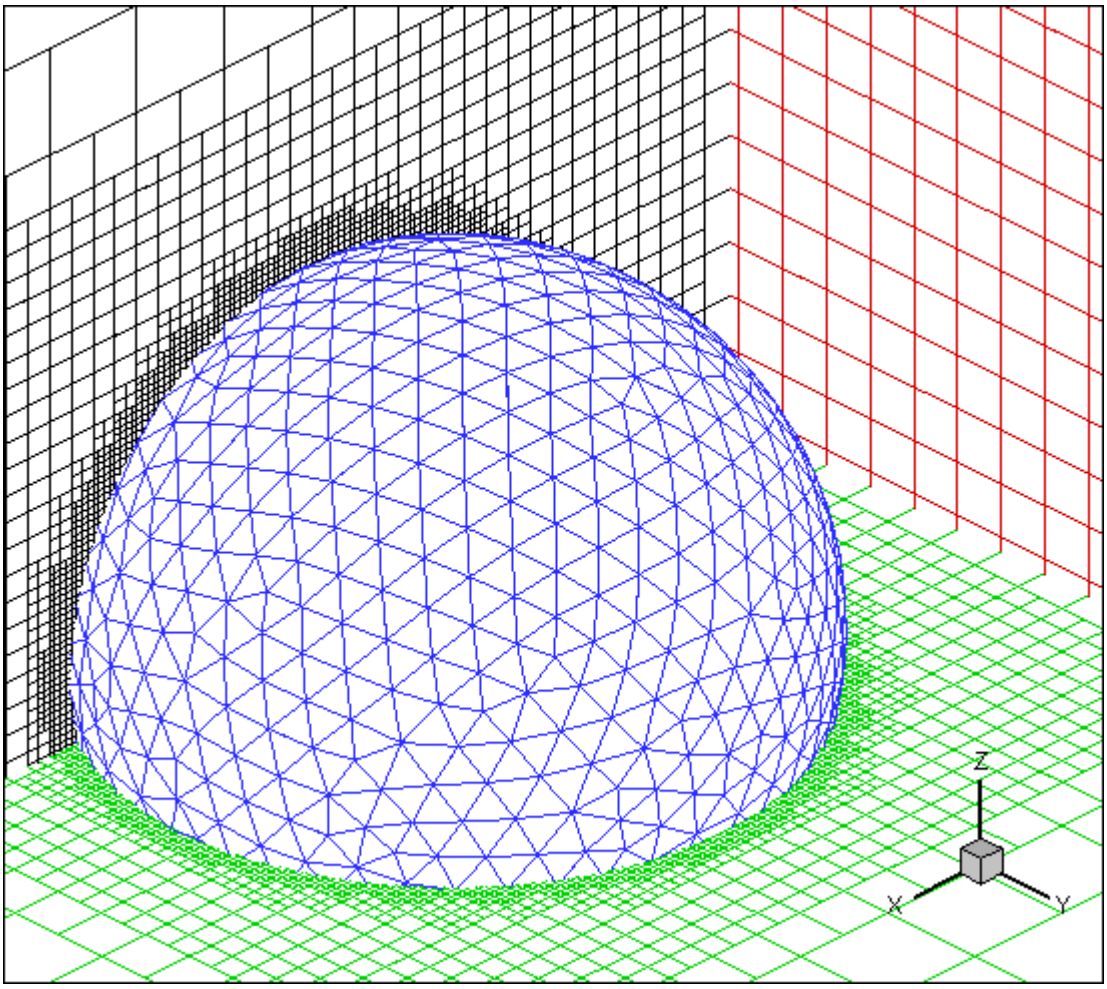
(a) (b)

Şekil 3.8 Küplerle ilerleme yöntemi

Ayrıca görüldüğü gibi kesik yüzeyi oluşturan tüm üçgenlerin normal vektörleri kesik hücrenin geometri dışında kalan yönünü göstermektedir. Bu özellik akı, hacim ve merkez kordinatlarının hesaplanmasını kolaylaştırmaktadır.

3.2.3.2 Kesik Hücre Adaptasyonu

Kutu adaptasyonundan sonra geometri çevresinde daha yoğun bir ağ yaratmak ve kesik hücrelerle dış hücreler arasındaki geçişin daha düzgün olması sağlamak için bu adaptasyondan faydalanılır. Bu adaptasyonda da bir seviye kuralı kontrol edilerek hücre bölünmesi yapılmalıdır. Şekil 3.6'daki kutu adaptasyonlu ağa kesik hücre adaptasyonu uygulanırsa Şekil 3.9 elde edilir.



Şekil 3.9 Kesik hücre adaptasyonuna bir örnek

BÖLÜM 4

EULER ÇÖZÜCÜSÜNÜN GELİŞTİRİLMESİ

4.1 ÜÇ BOYUTLU TEMEL DENKLEMLER

Sıkıştırlabilir akışlar için Euler denklemleri korunabilir şekilde aşağıda verildiği gibi ifade edilebilir.

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{q}\,d\Omega + \oint_{A}(\mathbf{F}\bullet\mathbf{n})\,dA = 0 \tag{4.1}$$

Burada t zamanı, Ω kontrol hacmini, A kontrol yüzeyini, $\mathbf{q}$ korunabilir değişken vektörünü, $\mathbf{F} = \mathbf{F}_x\vec{\mathbf{i}} + \mathbf{F}_y\vec{\mathbf{j}} + \mathbf{F}_z\vec{\mathbf{k}}$ akı vektörünü, $\mathbf{n} = (n_x, n_y, n_z)^T$ ise birim normal vektörü göstermektedir.

$$\mathbf{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \tag{4.2}$$

$$\mathbf{F}_x = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho uH \end{pmatrix}, \quad \mathbf{F}_y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho vH \end{pmatrix}, \quad \mathbf{F}_z = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho wH \end{pmatrix} \tag{4.3}$$

Akı vektörü ile normal vektörünün nokta çarpımı aşağıdaki denklemde verilmektedir.

$$\Phi = \begin{pmatrix} \rho V_n \\ \rho u V_n + p n_x \\ \rho v V_n + p n_y \\ \rho w V_n + p n_z \\ \rho V_n H \end{pmatrix} \tag{4.4}$$

Bu denklemde $V_n = un_x + vn_y + wn_z$ yüzeyden dışarı çıkan normal hız vektörünün değeri, ρ akışkanın yoğunluğu, $\mathbf{v} = u\vec{\mathbf{i}} + v\vec{\mathbf{j}} + w\vec{\mathbf{k}}$ akışkanın hız vektörüdür. Geliştirilen yazılımda kullanılan termodinamik ilişkilerde aşağıdaki denklemlerde verilmektedir.

$$E = e + \frac{u^2 + v^2 + w^2}{2} \tag{4.5}$$

$$H = E + \frac{p}{\rho} \tag{4.6}$$

$$p = \rho e(\gamma - 1) = \rho(\gamma - 1)(E - \frac{u^2 + v^2 + w^2}{2}) \tag{4.7}$$

Bu denklemlerdeki, *H* spesifik toplam entalpi, *E* spesifik toplam enerji, *p* akışkanın statik basıncı, *e* spesifik iç enerji ve $\gamma = c_p/c_v$ spesifik ısı katsayısıdır. Geliştirilen yazılımda, bazı değerlerin başlangıç değerleri birim değere çok yakın seçilmiştir ve böylece hesaplama zamanından tasarruf edilmiştir. Mesela, uzak alan akışkan yoğunluğu bir, statik basıncı da $1/\gamma$ olarak seçilmiştir. Böylece uzak alan ses hızı aşağıdaki denklem kullanılarak bir bulunur.

$$c_\infty = \sqrt{\frac{p_\infty \gamma}{\rho_\infty}} \tag{4.8}$$

Sonuç olarak, uzak alan hız vektörünün büyüklüğü, $\|\mathbf{v}\|$, girdi Mach sayısı olarak elde edilir.

4.2 ÜÇ BOYUTLU TEMEL DENKLEMLERİN AYRIŞTIRILMASI

(4.1) numaralı denklem, uzayda üç boyutlu olarak ayrıştırılırsa

$$\frac{\partial \mathbf{q}}{\partial t} = -\frac{1}{\Omega} \sum_{i=1}^{nKenarlar} \Phi_i A_i \tag{4.9}$$

ve artıklar aşağıdaki gibi tarif edilirse

$$Res(\mathbf{q}) = \sum_{i=1}^{nKenarlar} \Phi_i A_i \tag{4.10}$$

aşağıdaki denklem elde edilir.

$$\frac{\partial \mathbf{q}}{\partial t} = -\frac{Res(\mathbf{q})}{\Omega} \tag{4.11}$$

(4.11) numaralı denklemdeki akı terimlerinin elde edilmesinden sonra korunabilir değişkenlerin güncellenmesi için zaman boyutunda ayrıklaştırma yapılması gerekir. Bu ayrıklaştırma için çok kademeli zaman adımlama (multi-stage time stepping) yöntemi kullanılmıştır. Çok kademeli yöntemlerin detayları Lambert (1973) tarafından incelenmiştir.

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t} \tag{4.12}$$

Zamanda ayrıştırma iki farklı yöntemle olmaktadır. Bunlardan ilki kapalı (implicit), diğeri açık (explicit) yöntemlerdir. Bu yöntemler sırasıyla (4.13) ve (4.14) numaralı denklemlerdir.

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t} = -\frac{1}{\Omega}\left[Res(\mathbf{q}^{n+1})\right] \tag{4.13}$$

$$\frac{\mathbf{q}^{n+1} - \mathbf{q}^{n}}{\Delta t} = -\frac{1}{\Omega}\left[Res(\mathbf{q}^{n})\right] \tag{4.14}$$

(4.13) numaralı denklemdeki $Res(\mathbf{q}^{n+1})$ değeri aşağıda verilmektedir.

$$Res(\mathbf{q}^{n+1}) = Res(\mathbf{q}^{n}) + \frac{\partial (Res)}{\partial \mathbf{q}}(\mathbf{q}^{n+1} - \mathbf{q}^{n}) \tag{4.15}$$

4.2.1 Açık Zaman Adımı Şeması

Üç kademeli zaman adımlama yönteminin uygulanması aşağıdaki gibidir.

$$\mathbf{q}^{0} = \mathbf{q}^{n} \tag{4.16a}$$

$$\mathbf{q}^{1} = \mathbf{q}^{0} - \frac{\upsilon \alpha_1 \Delta t Res(\mathbf{q}^{0})}{\Omega} \tag{4.16b}$$

$$\mathbf{q}^{2} = \mathbf{q}^{0} - \frac{\upsilon \alpha_2 \Delta t Res(\mathbf{q}^{1})}{\Omega} \tag{4.16c}$$

$$\mathbf{q}^{3} = \mathbf{q}^{0} - \frac{\upsilon \alpha_3 \Delta t Res(\mathbf{q}^{2})}{\Omega} \tag{4.16d}$$

$$\mathbf{q}^{n+1} = \mathbf{q}^{3} \tag{4.16e}$$

Yukarıdaki denklemde α'lar kademe katsayılarını göstermekte olup, artıklar bir önceki kademedeki korunabilir değişkenlerin değerleri kullanılarak hesaplandığı için bu formülasyon açık (explicit) bir formülasyondur. Bu yöntemde, Blazek'in (2001) belirtiği gibi Runge-Kutta yönteminin aksine sadece sıfırıncı çözüm ile son artığın saklanması gerekir. *α* değerlerinin kararlılık için ayarlanmasıyla daha büyük zaman adımlarının kullanılması mümkün olur. Blazek (2001) tarafından belirlenen iki boyutlu problemler

için birinci dereceden hassas üç, dört ve beş kademeli yöntemler için kademe katsayılarının ve CFL sayılarının optimize edilmiş değerleri Tablo 4.1'de verilmiştir. CFL sayısı hücre alanlarına göre ne kadar büyük zaman adımları kullanılabileceğini gösteren bir faktördür.

Tablo 4.1 İki boyutlu, birinci derecede hassas çok kademeli yöntemler için kademe katsayıları ve CFL sayıları

	3	4	5
υ	1.5	2.0	2.5
α_1	0.1481	0.0833	0.0533
α_2	0.4	0.2069	0.1263
α_3	1.0	0.4265	0.2375
α_4		1.0	0.4414
α_5			1.0

Tablo 4.2 Üç boyutlu, birinci derecede hassas çok kademeli yöntemler için kademe katsayıları ve CFL sayıları

	2
υ	1.0
α_1	0.4361
α_2	1.0

4.2.2 Zaman Adımının Belirlenmesi

(4.16) numaralı denklemde zaman adımının değeri bilinmemektedir. Eğer zaman adımının değeri çok büyük olarak seçilirse kararlılıktan kaynaklanan problemler ortaya çıkabilmektedir. Eğer zaman adımı çok küçük olarak seçilirse, yakınsama çok yavaş olmakta ve gereksiz hesaplamalar yapılabilmektedir.

Genel olarak, herhangi bir geometri için hazırlanan hesaplama ağında değişik büyüklükte hücreler bulunur. Her bir hücre için kullanılabilecek en büyük zaman adımı, hücrenin büyüklüğüne bağlıdır. Bu

nedenle, herbir hücre için kullanılabilecek en büyük zaman adımı değişik olur. Zamana bağlı olmayan bir çözüm arandığından her hücre için değişik zaman adımı kullanılması mümkündür.

Blazek'e (2001) göre zaman adımı, iki boyutlu hücreler için hücrenin alanı cinsinden aşağıdaki gibi hesaplanabilir.

$$\frac{\Delta t}{A_{hücre}} = \frac{\sigma}{\Psi^x + \Psi^y} \tag{4.17}$$

Yukarıdaki denklemdeki spektral yarıçap Ψ, hücre kenarlarının *x* ve *y* yönlerindeki izdüşümlerinin uzunluğu kullanılarak hesaplanabilir,

$$\Psi^x = \frac{1}{2}\left(|u| + c_{hücre}\right) \sum_{kenarlar} S^x \tag{4.18}$$

$$\Psi^y = \frac{1}{2}\left(|v| + c_{hücre}\right) \sum_{kenarlar} S^y \tag{4.19}$$

Kararlı çözümler elde edebilmek için bulunan zaman adımının bir emniyet faktörü ile çarpılması gerekir. (4.17) numaralı denklem incelendiğinde karakteristik bir uzunluğun karakteristik bir hıza bölündüğü görülür. Buradaki karakteristik hız bir hücre içerisindeki bilgi ilerleyişinin en büyük hızıdır. Başka bir deyişle, zaman adımı bir hücre içerisinde bilginin hareket ettiği mesafeye göre belirlenmektedir.

Üç boyutlu hücreler için zaman adımı aşağıdaki gibi hesaplanabilir.

$$\frac{\Delta t}{\Omega_{hücre}} = \frac{1}{\sum_{i=1}^{nKenarlar} (c_{hücre} + |V_n|)_i A_i} \tag{4.20}$$

4.3 AKILARIN HESAPLANMASI

Üç boyutlu akış çözücülerin en önemli kısımlarından biriside akıların hesaplanması kısmıdır. Bu çalışmada, akılar iki farklı yöntemle hesaplanmıştır. Bunlardan ilki Roe'nun yaklaşık Riemann çözücüsü, diğeri AUSM (Liou-Steffen) akı vektörü ayrıştırmasıdır.

4.3.1 Roe'nun Yaklaşık Reimann Çözücüsü

Bir yüzeyden geçen akı, yüzeyin orta noktasında iki komşu hücrenin korunabilir değerleri kullanılarak hesaplanır. Aşağıdaki denklemlerde bu iki komşunun değerleri L (left=sol) ve R (right=sağ) olarak gösterilmektedir.

$$\Phi(\mathbf{q_L},\mathbf{q_R}) = \frac{1}{2}\left[\Phi(\mathbf{q_L}) + \Phi(\mathbf{q_R})\right] - \frac{1}{2}\sum_{k=1}^{5}\left|\lambda_k\right|\Delta\mathbf{V_k}\,\mathbf{R_k} \tag{4.21}$$

$$\Phi(\mathbf{q_L}) = \begin{pmatrix} \rho V_n \\ \rho u V_n + p n_x \\ \rho v V_n + p n_y \\ \rho w V_n + p n_z \\ \rho V_n H \end{pmatrix}_{\mathbf{L}}, \qquad \Phi(\mathbf{q_R}) = \begin{pmatrix} \rho V_n \\ \rho u V_n + p n_x \\ \rho v V_n + p n_y \\ \rho w V_n + p n_z \\ \rho V_n H \end{pmatrix}_{\mathbf{R}} \tag{4.22}$$

(4.22) numaralı denklemin sağ tarafındaki ikinci terimdeki değerler Roe'nun ortalama değerleridir ve bu değerler aşağıdaki denklemler kullanılarak hesaplanırlar.

$$\rho_{\mathbf{RL}} = \sqrt{\rho_{\mathbf{L}}\rho_{\mathbf{R}}} \tag{4.23}$$

$$u_{\mathbf{RL}} = \frac{u_{\mathbf{R}}\sqrt{\rho_{\mathbf{R}}} + u_{\mathbf{L}}\sqrt{\rho_{\mathbf{L}}}}{\sqrt{\rho_{\mathbf{R}}} + \sqrt{\rho_{\mathbf{L}}}} \tag{4.24}$$

$$v_{\mathbf{RL}} = \frac{v_{\mathbf{R}}\sqrt{\rho_{\mathbf{R}}} + v_{\mathbf{L}}\sqrt{\rho_{\mathbf{L}}}}{\sqrt{\rho_{\mathbf{R}}} + \sqrt{\rho_{\mathbf{L}}}} \tag{4.25}$$

$$w_{\mathbf{RL}} = \frac{w_{\mathbf{R}}\sqrt{\rho_{\mathbf{R}}} + w_{\mathbf{L}}\sqrt{\rho_{\mathbf{L}}}}{\sqrt{\rho_{\mathbf{R}}} + \sqrt{\rho_{\mathbf{L}}}} \tag{4.26}$$

$$H_{\mathbf{RL}} = \frac{H_{\mathbf{R}}\sqrt{\rho_{\mathbf{R}}} + H_{\mathbf{L}}\sqrt{\rho_{\mathbf{L}}}}{\sqrt{\rho_{\mathbf{R}}} + \sqrt{\rho_{\mathbf{L}}}} \tag{4.27}$$

$$c_{\mathbf{RL}} = \sqrt{(\gamma-1)\left(H_{\mathbf{RL}} - \frac{u_{\mathbf{RL}}{}^2 + v_{\mathbf{RL}}{}^2 + w_{\mathbf{RL}}{}^2}{2}\right)} \tag{4.28}$$

$$(V_n)_{\mathbf{RL}} = u_{\mathbf{RL}} n_x + v_{\mathbf{RL}} n_y + w_{\mathbf{RL}} n_z \tag{4.29}$$

Öz değerleri, Roe'nun ortalama değerleri kullanılarak hesaplanan normal hızın kullanılmasıyla hesaplanır. Öz değerleri, (4.30) numaralı, dalga kuvvetleri (4.31) numaralı ve doğru karakteristik vektörleri (right characteristic vectors) (4.32) numaralı denklemlerde verilmektedir.

$$\lambda = \begin{bmatrix} (V_n)_{\mathbf{RL}} - c_{\mathbf{RL}} \\ (V_n)_{\mathbf{RL}} \\ (V_n)_{\mathbf{RL}} \\ (V_n)_{\mathbf{RL}} \\ (V_n)_{\mathbf{RL}} + c_{\mathbf{RL}} \end{bmatrix} \tag{4.30}$$

$$\Delta\mathbf{V} = \begin{bmatrix} \dfrac{\Delta p - \rho_{RL} c_{RL} \Delta V_n}{2c_{RL}^{\,2}} \\ \dfrac{c_{RL}^{\,2}\Delta\rho - \Delta p}{c_{RL}^{\,2}} \\ \dfrac{n_y \Delta w - n_z \Delta v}{n_y^{\,2} + n_z^{\,2}} \\ \dfrac{n_x n_y \Delta v + n_x n_z \Delta w}{n_y^{\,2} + n_z^{\,2}} - \Delta u \\ \dfrac{\Delta p + \rho_{RL} c_{RL} \Delta V_n}{2c_{RL}^{\,2}} \end{bmatrix} \tag{4.31}$$

$$\mathbf{R} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ u_{RL} - c_{RL} n_x & u_{RL} & 0 & \rho_{RL}(n_y^2 + n_z^2) & u_{RL} + c_{RL} n_x \\ v_{RL} - c_{RL} n_y & v_{RL} & -\rho_{RL} n_z & \rho_{RL} n_x n_y & v_{RL} + c_{RL} n_y \\ w_{RL} - c_{RL} n_z & w_{RL} & \rho_{RL} n_y & \rho_{RL} n_x n_z & w_{RL} + c_{RL} n_z \\ H_{RL} - c_{RL}(V_n)_{RL} & \dfrac{\delta}{2} & \rho_{RL}(w_{RL} n_y - v_{RL} n_z) & \rho_{RL}\left[(V_n)_{RL} n_x - u_{RL}\right] & H_{RL} + c_{RL}(V_n)_{RL} \end{bmatrix} \tag{4.32}$$

Bu denklemlerde bazı değerlerin hesaplanması da aşağıdaki denklemlerde gösterilmektedir.

$$\Delta\rho = \rho_R - \rho_L \tag{4.33}$$

$$\Delta p = p_R - p_L \tag{4.34}$$

$$\Delta u = u_R - u_L \tag{4.35}$$

$$\Delta v = v_R - v_L \tag{4.36}$$

$$\Delta w = w_R - w_L \tag{4.37}$$

$$\Delta V_n = (V_n)_R - (V_n)_L \tag{4.38}$$

$$\delta = u_{RL}^2 + v_{RL}^2 + w_{RL}^2 \tag{4.39}$$

4.3.2 AUSM (Liou-Steffen) Akı Vektörü Ayrıştırması

AUSM akı vektörü ayrıştırması yöntemi, basınç ve normal momentum akısında aşağıda görülen adveksiyon terimlerini ayrıştırır.

$$\Phi(\mathbf{q}_L, \mathbf{q}_R) = \frac{1}{2}\left[M_{1/2}\left(\mathbf{F}'(\mathbf{q}_L) + \mathbf{F}'(\mathbf{q}_R)\right) - \left| M_{1/2} \right| \left(\mathbf{F}'(\mathbf{q}_R) - \mathbf{F}'(\mathbf{q}_L)\right) \right] + \mathbf{p}_{1/2}\, 1 \tag{4.40}$$

(4.40) numaralı denklemde görülen, $M_{1/2}$ ara yüzey Mach sayısı, $p_{1/2}$ ara yüzey basınç vektörü ve Φ ise akı vektörü olup aşağıdaki gibi tanımlanabilir.

$$\Phi = V_n \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{pmatrix} + p \begin{pmatrix} 0 \\ n_x \\ n_y \\ n_z \\ 0 \end{pmatrix} = M \begin{pmatrix} \rho c \\ \rho u c \\ \rho v c \\ \rho w c \\ \rho H c \end{pmatrix} + p \begin{pmatrix} 0 \\ n_x \\ n_y \\ n_z \\ 0 \end{pmatrix} = M \cdot \mathbf{F}' + \mathbf{p} \tag{4.41}$$

$$\mathbf{F}'(\mathbf{q_L}) = \begin{pmatrix} \rho c \\ \rho u c \\ \rho v c \\ \rho w c \\ \rho H c \end{pmatrix}_{\mathbf{L}} \quad \text{ve} \quad \mathbf{F}'(\mathbf{q_R}) = \begin{pmatrix} \rho c \\ \rho u c \\ \rho v c \\ \rho w c \\ \rho H c \end{pmatrix}_{\mathbf{R}} \tag{4.42}$$

$$M_{1/2} = \frac{1}{2}\left(M_{\mathbf{L}}^{+} + M_{\mathbf{R}}^{-}\right) \qquad \mathbf{p}_{1/2} = \frac{1}{2}\left(\mathbf{p}_{\mathbf{L}}^{+} + \mathbf{p}_{\mathbf{R}}^{-}\right) \tag{4.43}$$

$$M_{\mathbf{L}}^{+} = \begin{cases} \dfrac{1}{4}\left(M_{\mathbf{L}} + 1\right)^2 & \left|M_{\mathbf{L}}\right| \le 1 \\ \dfrac{1}{2}\left(M_{\mathbf{L}} + \left|M_{\mathbf{L}}\right|\right) & \left|M_{\mathbf{L}}\right| > 1 \end{cases} \tag{4.44}$$

$$M_{\mathbf{R}}^{-} = \begin{cases} -\dfrac{1}{4}\left(M_{\mathbf{R}} - 1\right)^2 & \left|M_{\mathbf{R}}\right| \le 1 \\ \dfrac{1}{2}\left(M_{\mathbf{R}} - \left|M_{\mathbf{R}}\right|\right) & \left|M_{\mathbf{R}}\right| > 1 \end{cases} \tag{4.45}$$

$$\mathbf{p}_{\mathbf{L}}^{+} = \mathbf{p}_{\mathbf{L}} M_{\mathbf{L}}^{+} \begin{cases} 2 - M_{\mathbf{L}} & \left|M_{\mathbf{L}}\right| \le 1 \\ \dfrac{1}{M_{\mathbf{L}}} & \left|M_{\mathbf{L}}\right| > 1 \end{cases} \tag{4.46}$$

$$\mathbf{p}_{\mathbf{R}}^{-} = \mathbf{p}_{\mathbf{R}} M_{\mathbf{R}}^{-} \begin{cases} -2 - M_{\mathbf{R}} & \left|M_{\mathbf{R}}\right| \le 1 \\ \dfrac{1}{M_{\mathbf{R}}} & \left|M_{\mathbf{R}}\right| > 1 \end{cases} \tag{4.47}$$

4.4 SINIR ŞARTLARI

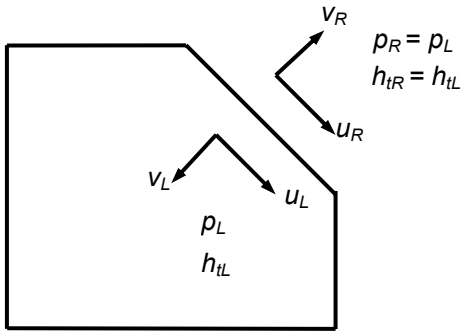
Bu çalışmada dış akışların ele alınması öngörüldüğünden duvar ve uzak alan sınır şartları ele alınacaktır. Seçilen sayısal yöntemde her iki sınır şartınıın da uygulanması sol durum değişkenlerinin ele alınan hücreden alınması ve sağ durum değişkenlerinin düzeltilerek elde edilmesine karşılık gelmektedir.

4.4.1 Duvar Sınır Şartı

Bu bölümde iki boyutlu akış problemleri için sınır şartları anlatılacaktır. Üç boyutlu sınır şartlarıda iki boyutlu sınır şartlarına çok benzemektedir.

Duvar sınır şartı için gerekli fiziksel sınır şartı duvardan akı geçişi olmamasıdır. Viskoz olmayan akışlar için, bu durum hücrenin yüzündeki hızın yüze teğet olması anlamına gelmektedir.

De Zeeuw (1993) tarafından verilen iki yöntemden bir tanesi sağ durum değişkenlerinin bulunması için kullanılabilir. Bu yöntemlerden birincisinde hücenin yüzünden geçen akı sıfır olarak alınarak, sağ akı sadece sol durumdaki basınca eşit olan basınnç kullanılarak hesaplanır. Zamana bağlı olmayan durumlarda, bu eşitliğin nedeni akışkan tarafından etki ettirilen basınç kuvveti eşit bir kuvvetle karşılanmasıdır. İkinci yöntemde ise teğet hız, basınç ve toplam entalpinin sol duruma eşit olduğu hayalet bir sağ durum kullanılmaktadır. Teğet hızın yönü ve büyüklüğü sol durumunkine eşittir. Normal hızın büyüklüğü ise sol durumun büyüklüğü ile aynı olmasına karşılık yönü terstir. Hayalet sağ durum Şekil 4.1'de gösterilmiştir. İkinci yöntemde De Zeeuw (1993) trafından da belirtildiği sadece hücrenin yüzünden geçen akının sıfır olmasının sağlanması yanında normal hız bileşkesinin bulunduğu eğriliğin yüksek olduğu yerleri de göz önüne alınmaktadır. Bu çalışmada duvar sınır şartlarının uygulanmasında ikinci yöntem uygulanmıştır.



Şekil 4.1 Duvar üzerindeki hayalet sağ durum

4.4.2 Uzak Alan Sınır Şartı

Uzak alan sınır şartında sağ durum sol duruma bakılmaksızın başlangıç şartlarından faydalanılarak aşağıdaki gibi hesaplanır.

$$\rho_R = 1 \tag{4.48}$$

$$u_R = M_\infty \cos\alpha \tag{4.49}$$

$$v_R = M_\infty \sin\alpha \tag{4.50}$$

$$p_R = \frac{1}{\gamma} \tag{4.51}$$

4.5 ÇOKLU AĞ YÖNTEMİYLE ÇÖZÜMÜN YAKINSAMASININ HIZLANDIRILMASI

Çoklu ağ yöntemi ilk olarak Poisson denkleminin birim kare içerisindeki çözümü için Fedorenko (1962, 1964) tarafından geliştirilmiştir. Daha sonra, Fedorenko'nun geliştirdiği yöntem diğer matematikçiler tarafından eliptik sınır şartı problemlerine uygun hale getirilmiştir. Fakat çoklu ağ yönteminin tam verimliliği Brandt (1977) tarafından yapılan çalışmalar sonucunda fark edilmiştir. Daha sonra, Brandt doğrusal olmayan problemler için yeni bir çoklu ağ yöntemi geliştirmişlerdir. Bu yöntem, tam tahmin depolama şeması (Full Approximation Storage - FAS Scheme) olarak adlandırılmaktadır. Çoklu ağ yöntemlerinin formülasyonu ile ilgili diğer bir başarı ise iç içe öteleme tekniği ile çoklu ağ yönteminin birleşimine dayanan tam çoklu ağ şemasıdır (Full Multigrid - FMG Scheme). Çoklu ağ yöntemleri şu anda bir çok değişik problemlere uygulanmaktadır. Bunların başında doğrusal ve doğrusal olmayan sınır şartı problemleri bulunmaktadır. Diğer örnekler ise parabolik problemler, eliptik problemler, hiperbolik problemler, karma eliptik ve hiperbolik problemler ve son olarak optimizasyon problemleridir. Çoklu ağ yöntemi ile ilgili son gelişme ise cebirsel çoklu ağ yöntemleridir (Algebraic Multigrid - AMG Methods).

Çoklu ağ yöntemi, çözümün yakınsamasını ivmelendirmek için kullanılan bir yöntemdir. Çoklu ağ yöntemi ile farklı seviyelerdeki ağlar kullanılarak, düşük ve yüksek frekanslardaki hatalar hızlı bir şekilde en aza indirilmektedir. Çoklu ağ yönteminin iki temel ilkesi bulunmaktadır. Bunlardan ilki, hata düzeltmedir (error smoothing). Herhangi bir başlangıç tahmini ile bir çok öteleme metodu (Jacobi veya Gauss Seidel gibi) kullanılarak yüksek frekans hatalarını etkili bir şekilde azaltmaya hata düzeltme yöntemi denilmektedir. Diğer bir ilke ise az yoğun ağ ilkesidir (coarse grid principle). Belirli bir ağ kullanılarak elde edilmiş düzgün değerler, herhangi bir bilgi kaybı olmadan daha az yoğun bir ağ için yaklaşık olarak elde edilir. Bu olaya sınırlama (restriction) denilmektedir. Böylece, az yoğun ağ yöntemi kullanılarak, çok yoğun ağ ile elde edilecek sonuçlara göre daha az zamanda tatmin edici sonuçlar elde edilir ve sonuç olarak az yoğun ağ ilkesi ile elde edilen düşük frekans hataları, yoğun ağ için elde edilen sonuçları düzeltmek için kullanılırlar. Bu olaya da uzatma (prolongation) denilmektedir. Diğer bir deyişle, çoklu ağ yönteminin temel fikri, ötelemeli yöntemlerle azaltılması zor olan düşük frekanslı hataların daha az yoğunluklu ağlarda çözüm yapılarak azaltılmasıdır. Daha az yoğun ağ üzerinde, düşük frekanslı hatalar yüksek frekanslı hatalarmış gibi davrandıklarından çoklu ağ yöntemiyle bu hataların daha verimli bir şekilde azaltılması mümkündür.

4.5.1 Doğrusal Problemler için Çoklu Ağ Yöntemi

Öncelikle çoklu ağ yönteminin açıklanmasını basitleştirmek için matris gösteriminden yararlanmak mümkündür. Aşağıdaki matris denklemi, doğrusal denklem sistemlerini göstermektedir.

$$A\mathbf{u} = \mathbf{f} \tag{4.52}$$

Buradaki **u** değeri, sistemin gerçek çözümü olup, öteleme yöntemiyle elde edilen sonucu da **v** değeri temsil etmektedir. Koyu yazılan semboller, vektörleri göstermek için ve belirli bir seviyedeki ağı olan Ω^h yi işaret etmek için de $\mathbf{u}^h$ ve $\mathbf{v}^h$ tanımları kullanılmaktadır. **u** değeri ile **v** değeri arasındaki fark ise hata terimi olan **e**'yi vermektedir.

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{4.53}$$

Doğrusal problemler için artakalan denklem sistemi kolayca yazılabilmektedir. **r** değeri de artakalan terimini vermektedir.

$$\mathbf{r} = \mathbf{f} - A\mathbf{v} \tag{4.54}$$

Doğrusal problemler için tek bir gerçek çözüm mevcut olduğundan artakalan terimi sadece ve sadece hata terimi sıfır olduğunda sıfır olmaktadır. Başka bir deyişle, bu ilişki göz önünde bulundurularak aşağıdaki denklem sayesinde hata terimleri elde edilmektedir.

$$A\mathbf{e} = \mathbf{r} \tag{4.55}$$

Son olarak, elde edilen hata terimleri kullanılarak, gerçek çözüme daha yakın sonuçlar elde edilmektedir.

$$\mathbf{v}^{yeni} = \mathbf{v} + \mathbf{e} \tag{4.56}$$

Çoklu ağ yöntemi genel hatları ile dört aşamadan oluşmaktadır. Bu aşamalar: (i) yoğun ağ ötelemesi, (ii) sınırlama, (iii) uzatma ve (iv) düzeltme ve son ötelemeler olarak verilmektedir.

(i) Çoklu Ağ Yönteminin İlk Aşaması: Yoğun Ağ Ötelemesi

Ağ aralığı *h* olan yoğun bir ağ üzerinde ilk olarak yaklaşık bir sonuç elde etmek için bir miktar öteleme yapılmalıdır. Bu ötelemelerden sonra $\mathbf{v}^h$ ara çözümü elde edilir. Yapılan ötelemeler sonucunda elde edilen ara çözümün matrise yerleştirilmesi sonucunda *h* aralıklı ağdaki artakalan $\mathbf{r}^h$ elde edilmiş olur. Bu durum aşağıdaki denklemde görülmektedir.

$$\mathbf{r}^h = \mathbf{f}^h - A^h\mathbf{v}^h \tag{4.57}$$

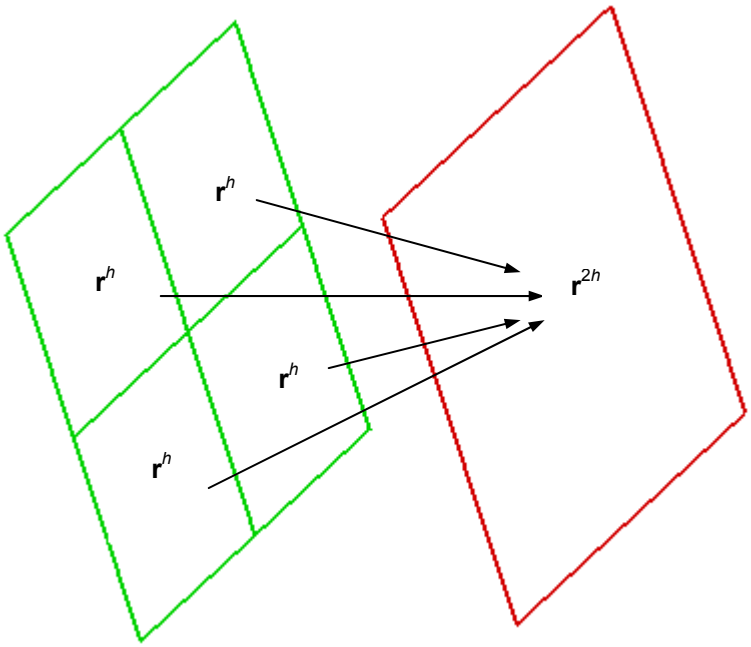
(ii) Çoklu Ağ Yönteminin İkinci Aşaması: Sınırlama

Bu aşamada, ağ aralığı *h* olan yoğun ağ, ağ aralığı *ch* olan daha az yoğun ağa çevrilmekte ve ötelemeler *ch* aralığına sahip ağ üzerinde devam ettirilmektedir. Buradaki *c* değeri birden büyük bir değer olup genellikle iki olarak alınmaktadır. İki alınması Kartezyen ağ sisteminde büyük kolaylık sağlamakta olup, hesaplama hücrelerinden daha az yoğun hücrelere sahip ağa geçiş, sadece çocuk hücrelerden

ebeveyn hücrelere bilgilerin taşınması şeklinde olmaktadır. Bu bilgi taşınımı öncelikle yoğun ötelemede elde edilen $\mathbf{r}^h$ (*h* aralıklı ağdaki artakalan) değerinin $\mathbf{r}^{ch}$ değerine dönüştürülmesi ile başlamaktadır. Bu durum, sınırlama işlemi olarak adlandırılmakta olup, geliştirilen kodda I_h^{2h} olarak gösterilen sınırlama operatörü kullanılmaktadır.

$$\mathrm{I}_h^{2h}\mathbf{r}^h = \mathbf{r}^{2h} \tag{4.58}$$

Bu dönüşüm basitçe *h* değerlerinin ortalama değerleri alınarak daha az yoğun hücrelere aktarılmasıdır. Bu durum Şekil 4.2'de iki boyutlu ısı problemi için artakalan değerinin çocuk hücrelerden nasıl ebeveyn hücreye taşındığını göstermektedir.



Şekil 4.2 Artakalan değerinin çocuk hücrelerden ebeveyn hücreye taşınması

$$\mathbf{r}^{2h} = \frac{\sum \mathbf{r}^h}{\frac{\text{çocuklar}}{4}} \tag{4.59}$$

Ağ aralığı *ch* olan daha az yoğun ağdaki artakalan değerleri hesaplandıktan sonra, bu ağ aralığına sahip hücrelerin katsayı matrisi de yeniden oluşturulmaktadır. İlk tahmin olarak $\mathbf{e}^{ch}$ değeri sıfır olarak alınmakta olup, herhangi bir öteleme yöntemi ile istenilen miktar kadar öteleme yapılıp yeni bir $\mathbf{e}^{ch}$ değeri elde edilmektedir. Geliştirilen kodda kullanılan öteleme yöntemi, Gauss Seidel öteleme yöntemidir. Bu aşamada kullanılan denklem aşağıda verilmiştir.

$$\mathrm{A}^{ch}\mathbf{e}^{ch} = \mathbf{r}^{ch} \tag{4.60}$$

Sınırlama aşamasında ağ yoğunluğu azaltıldığından düşük frekanslı hatalar sanki yüksek frekanslı hatalarmış gibi görünmektedir. Böylece, düşük frekanslı hataların da hızlıca azaltılması mümkün olur.

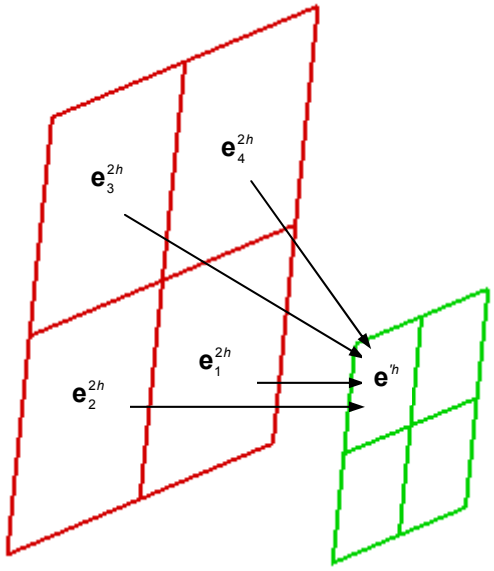
(iii) Çoklu Ağ Yönteminin Üçüncü Aşaması: Uzatma

Sınırlama aşamasında hücreler için $\mathbf{e}^{ch}$ değerleri bulunduktan sonra, bu değerlerin tekrar ağ aralığı *h* olan yoğun ağa aktarılmaları gerekmektedir. Aktarılacak bilgi sayısı, yeni ağdaki hücre sayısından az olduğu için interpolasyon yöntemine gereksinim duyulmaktadır. İnterpolasyon yöntemi ile yoğun ağdaki hücrelerin uzatılmış hata vektörleri $\mathbf{e}^{'h}$ elde edilmektedir. Geliştirilen yazılımda, uygulaması kolay olduğundan dolayı doğrusal interpolasyon kullanılmıştır. Doğrusal interpolasyon operatörü I_{2h}^{h} olarak gösterilmektedir.

$$I_{2h}^{h}\,\mathbf{e}^{2h} = \mathbf{e}^{'h} \tag{4.61}$$

İki boyutlu bir problem için doğrusal interpolasyon ile $\mathbf{e}^{'h}$ Şekil 4.3'te görüldüğü gibi aşağıdaki şekilde bulunmaktadır.

$$\mathbf{e}^{'h} = (9\mathbf{e}_1^{2h} + 3\mathbf{e}_2^{2h} + 3\mathbf{e}_4^{2h} + \mathbf{e}_3^{2h})/16 \tag{4.62}$$



Şekil 4.3 Hata vektörlerinin ebeveyn hücrelerden çocuk hücreye taşınması

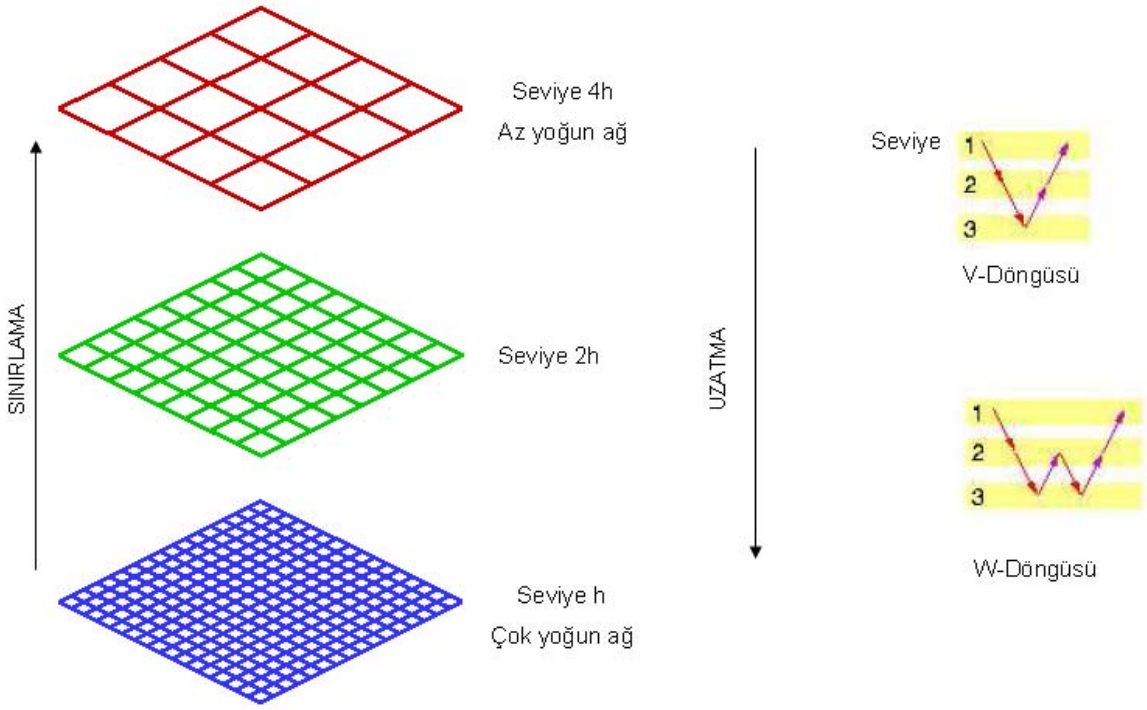
(iv) Çoklu Ağ Yönteminin Dördüncü Aşaması: Düzeltme ve Son Ötelemeler

Uzatma aşamasında hücreler için $\mathbf{e}^{'h}$ değerleri bulunduktan sonra, bu değerler ilk aşamada hesaplanan ara çözüm değerlerine eklenerek, aşağıdaki denklemde görüldüğü gibi düzeltilmiş ara çözüm değerleri elde edilmektedir.

$$\mathbf{v}^{h(yeni)} = \mathbf{v}^{h} + \mathbf{e}^{'h} \tag{4.63}$$

Uzun dalga boyuna sahip hatalar azaltıldığı için düzeltilmiş ara çözüm değerleri ile gerçek çözüm değerleri arasındaki fark azaltılmış olmaktadır. Sınırlama ve uzatma aşamalarında bazı yaklaşık değerler hesaplanarak çözüm elde edildiğinden çözümün doğruluğunu pekiştirmek için düzeltilmiş ara çözüm değerleri kullanılarak birkaç öteleme daha yapılmaktadır. Böylece, hem düşük hem de yüksek frekanslı hatalar azaltılarak, daha doğru bir yaklaşık değer hesaplanmaktadır.

İki boyutlu ısı iletimi problemi, doğrusal bir problem olduğundan, bu problem için geliştirilen yazılıma yukarıdaki yöntem uygulanarak çözüm elde edilmiş ve çoklu ağ yönteminin çözüme sağladığı avantajlar gözlenmiştir. Çözümler, V-döngüsü ve Şekil 4.4'teki seviyeler kullanılarak elde edilmiş ve çoklu ağ yönteminin testi bölümünde sunulmuştur.



Şekil 4.4 Çoklu ağ seti

4.5.2 Doğrusal Olmayan Problemler için Çoklu Ağ Yöntemi

Doğrusal olmayan problemleri çoklu ağ yöntemi kullanarak çözmek için iki ayrı yöntem bulunmaktadır. Bunlardan ilki Newton yöntemi, diğeri ise tam tahmin depolama şemasıdır (Full Approximation Storage - FAS) scheme). Önceki bölümde doğrusal problemlere çoklu ağ yönteminin nasıl uygulandığı incelenmiştir. Geliştirilen yazılımda, doğrusal olmayan Euler denklem sisteminin çoklu ağ

yöntemi kullanarak çözümünün elde edilmesi için tam tahmin depolama şeması (FAS scheme) kullanılmıştır.

Doğrusal duruma benzer olarak, doğrusal olmayan denklem sisteminin aşağıda verilen denklemle gösterilmesi mümkündür. Buradaki tek fark, A( ) operatörünün doğrusal olmadığını göstermek için A**u** gösterimi yerine A(**u**) gösteriminin kullanılmış olmasıdır.

$$A(\mathbf{u}) = \mathbf{f} \tag{4.64}$$

Buradaki **u** değeri, sistemin gerçek çözümü olup, öteleme yöntemiyle elde edilen sonucu ise **v** değeri temsil etmektedir. Hata vektörlerinin ve artakalan vektörlerinin gösterimi de oldukça kolaydır.

$$\mathbf{e} = \mathbf{u} - \mathbf{v} \tag{4.65}$$

$$\mathbf{r} = \mathbf{f} - A(\mathbf{v}) \tag{4.66}$$

Eğer (4.64) numaralı denklemden, (4.66) numaralı denklem olan artakalan denklemi çıkartılırsa aşağıdaki denklem elde edilmektedir.

$$A(\mathbf{u}) - A(\mathbf{v}) = \mathbf{r} \tag{4.67}$$

u – v = e olmasına karşılık, A( ) operatörü doğrusal olmadığından $A(\mathbf{u}) - A(\mathbf{v}) = A(\mathbf{e})$ sonucunun çıkartılması mümkün değildir. Bu durum, basit doğrusal artakalan denklemine sahip olunmadığını göstermektedir. Bu yüzden, artakalan denklemi olarak (4.67) numaralı denklem kullanılacaktır. Eğer gerçek çözüm değeri olan **u** yerine **v** + **e** kullanılırsa, (4.67) numaralı denklem aşağıdaki denkleme dönüşmektedir.

$$A(\mathbf{v} + \mathbf{e}) - A(\mathbf{v}) = \mathbf{r} \tag{4.68}$$

Ω^h seviyesindeki en yoğun ağ kullanılarak elde edilen yaklaşık çözümün $\mathbf{v}^h$ olduğu varsayılırsa, (4.68) numaralı denklemin kullanılmasıyla aşağıdaki denklem elde edilmektedir.

$$A^{2h}\left(\mathbf{v}^{2h} + \mathbf{e}^{2h}\right) - A^{2h}\left(\mathbf{v}^{2h}\right) = \mathbf{r}^{2h} \tag{4.69}$$

Buradaki A^{2h}( ) operatörü, ağ boyutu en yoğun ağ seviyesindeki ağ boyutunun iki katı olan az yoğun ağ operatörünü, $\mathbf{r}^{2h}$ az yoğun ağdaki artakalan vektörünü göstermektedir. $\mathbf{v}^{2h}$ terimi ise Ω^h seviyesindeki en yoğun ağ kullanılarak öteleme yöntemiyle elde edilen yaklaşık çözümün ağırlıklı sınırlama operatörü kullanarak Ω^{2h} seviyesindeki az yoğun ağ için yaklaşık çözüm değerlerini ifade etmektedir. (4.69) numaralı denklem kullanılarak $\mathbf{e}^{2h}$ hata vektörleri elde edildiğinde, aşağıdaki denklemde görüldüğü gibi en yoğun ağ için yaklaşık çözüm vektörleri olan $\mathbf{v}^h$ yenilenmektedir. Böylece, düşük frekans hataları azaltılarak yaklaşık çözüm değerleri elde edilmektedir.

$$\mathbf{v}^{h(yeni)} = \mathbf{v}^h + I_{2h}^h \mathbf{e}^{2h} \tag{4.70}$$

Az yoğun ağdaki artakalan vektörleri

$$\mathbf{r}^{2h} = \mathrm{I}_h^{2h}\mathbf{r}^h = \mathrm{I}_h^{2h}\left(\mathbf{f}^h - \mathrm{A}^h\left(\mathbf{v}^h\right)\right) \tag{4.71}$$

aşağıdaki denklem kullanılarak elde edilmektedir.

$$\mathbf{v}^{2h} = \mathrm{I}_h^{2h}\mathbf{v}^h \tag{4.72}$$

Eğer (4.71) ve (4.72) numaralı denklemler, (4.69) numaralı denklemin içine yerleştirildiğinde ve $\mathbf{u}^{2h} = \mathrm{I}_h^{2h}\mathbf{v}^h + \mathbf{e}^{2h}$ ifadesi göz önüne alındığında aşağıdaki denklem elde edilmektedir.

$$\mathrm{A}^{2h}\left(\mathrm{I}_h^{2h}\mathbf{v}^h + \mathbf{e}^{2h}\right) = \mathrm{A}^{2h}\left(\mathrm{I}_h^{2h}\mathbf{v}^h\right) + \mathrm{I}_h^{2h}\left(\mathbf{f}^h + \mathrm{A}^h\left(\mathbf{v}^h\right)\right) \tag{4.73}$$

Ayrıca, (4.74) numaralı denklemin, aşağıda verilen τ düzeltmesi (τ correction or residual correction)

$$\tau_h^{2h} = \mathrm{A}^{2h}\left(\mathrm{I}_h^{2h}\mathbf{v}^h\right) - \mathrm{I}_h^{2h}\left(\mathrm{A}^h\left(\mathbf{v}^h\right)\right) \tag{4.74}$$

kullanılarak aşağıdaki gibi de yazılması mümkündür.

$$\mathrm{A}^{2h}\left(\mathbf{u}^{2h}\right) = \mathrm{I}_h^{2h}\mathbf{f}^h + \tau_h^{2h} \tag{4.75}$$

(4.73) numaralı denklemin sağ tarafı bilindiğinden, az yoğun ağdaki hata değerlerini elde edebilmek için bu denklem kullanılarak Ω^{2h} seviyesindeki az yoğun ağ için $\mathbf{u}^{2h}$ çözüm değerlerinin bulunması mümkün olmaktadır.

$$\mathbf{e}^{2h} = \mathbf{u}^{2h} - \mathrm{I}_h^{2h}\mathbf{v}^h \tag{4.76}$$

Son olarak, (4.76) numaralı denklem, (4.70) numaralı denklemin içine yerleştirilirse, aşağıdaki denklem elde edilmektedir.

$$\mathbf{v}^{h(\text{yeni})} = \mathbf{v}^h + \mathrm{I}_{2h}^h\left(\mathbf{u}^{2h} + \mathrm{I}_h^{2h}\mathbf{v}^h\right) \tag{4.77}$$

Görüldüğü gibi, çoklu ağ yönteminin kullanıldığı doğrusal problemlerde az yoğun ağlarda yapılan hesapların amacı sadece hata vektörlerinin bulunmasıdır. Buna karşılık, çoklu ağ yönteminin kullanıldığı doğrusal olmayan problemlerde az yoğun ağlarda yapılan hesapların amacı az yoğun ağlardaki düşük frekanslı hataların azaltıldığı yaklaşık değerleri elde edilmesidir.

Doğrusal olmayan iki boyutlu Euler denklem sistemlerinin çoklu ağ tam tahmin depolama yöntemi (Multigrid FAS Method) kullanılarak elde edilen çözümleri, iki boyutlu Kartezyen ağ üreticisi ile Euler çözücüsünün test edilmesi bölümünde sunulmuştur. Fakat öncelikle bu yöntemin geliştirilen yazılıma nasıl uygulandığı açıklanmalıdır.

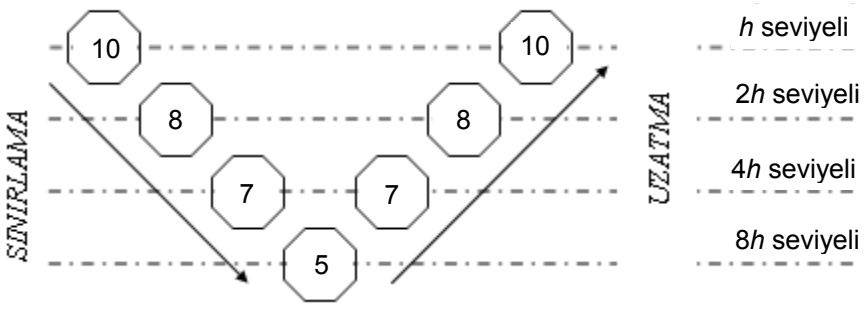
Doğrusal problemlerde olduğu gibi doğrusal olmayan problemler için de çoklu ağ yöntemi dört aşamadan oluşmaktadır. Bu aşamalar: (i) yoğun ağ ötelemesi, (ii) sınırlama, (iii) uzatma ve (iv) düzeltme ve son ötelemeler olarak verilmektedir.

(i) Çoklu Ağ Yönteminin İlk Aşaması: Yoğun Ağ Ötelemesi

Çoklu ağ yöntemi için adapte edilmiş çok kademeli zaman adımlama yöntemi aşağıdaki denklem kullanılarak ağ aralığı *h* olan yoğun bir ağ üzerinde ilk olarak yaklaşık bir sonuç elde etmek için kullanılmaktadır. Bu amaçla bir miktar öteleme yapılması gerekmektedir.

$$\begin{aligned}
&\mathbf{u}_0^h = \text{baslangic degerleri} \\
&\mathbf{u}_1^h = \mathbf{u}_0^h - \beta_1 \frac{\Delta t}{A_{hücre}} (\mathbf{R}(\mathbf{u}_0^h) + \mathbf{FF}^h) \\
&\text{.....} \\
&\mathbf{u}_n^h = \mathbf{u}_0^h - \beta_n \frac{\Delta t}{A_{hücre}} (\mathbf{R}(\mathbf{u}_{n-1}^h) + \mathbf{FF}^h)
\end{aligned} \tag{4.78}$$

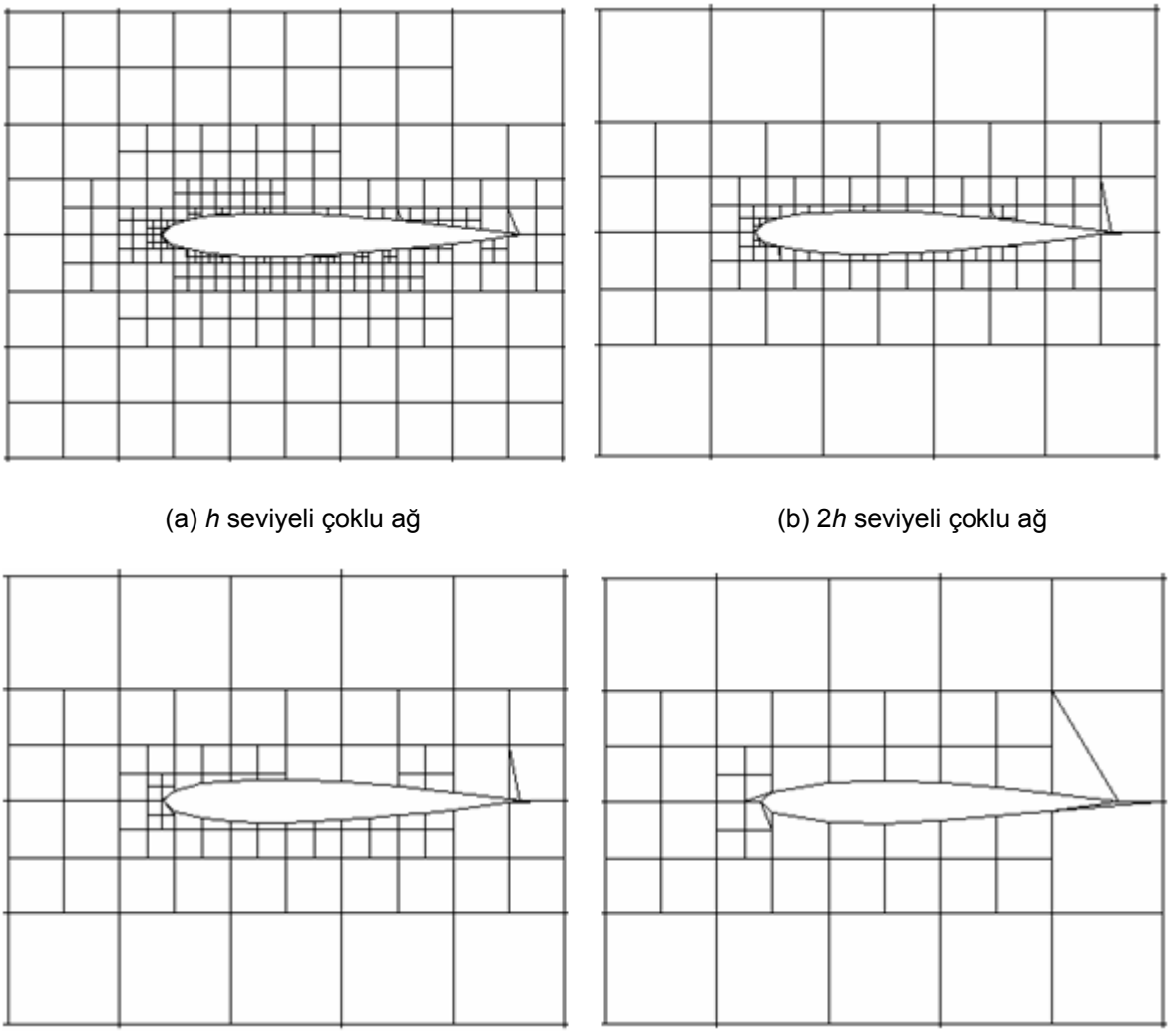
(4.78) numaralı denklemdeki $\mathbf{FF}^h$ terimi, zorlayıcı fonksiyon (forcing function) olarak adlandırılmakta olup en yoğun ağdaki tüm yaprak hücreler için değeri sıfırdır. Şekil 4.5'te gösterilen çoklu ağ seti örnek alınacak olursa, görüldüğü gibi çoklu ağ yönteminin bu aşaması için yapılacak öteleme sayısı on olmaktadır. Bu ötelemelerden sonra $\mathbf{u}_n^h$ ara çözümü elde edilmektedir. Hatanın kısa dalga boyuna sahip salınımsal bileşeninin etkili bir şekilde azaltılması için öteleme sayısının yeteri kadar büyük seçilmesi gerekmektedir. Fakat unutulmamalıdır ki, her ne yapılırsa yapılsın, uzun dalga boyuna sahip salınımsal hataların ağ aralığı *h* olan yoğun ağda yapılan ötelemelerle azaltılması mümkün değildir. Son olarak, yapılan ötelemeler sonucunda elde edilen ara çözümler kullanılarak tüm yaprak hücreler için artakalan değerleri $\mathbf{R}(\mathbf{u}_n^h)$ hesaplanmaktadır.



Şekil 4.5 Dört seviyeli çoklu ağ seti (V-döngüsü)

(ii) Çoklu Ağ Yönteminin İkinci Aşaması: Sınırlama

Bu aşamada, ağ aralığı *h* olan yoğun ağ, daha az yoğun ağlara çevrilmektedir. Şekil 4.5'te gösterilen çoklu ağ seti örnek alınacak olursa, bu set için örnek *h*, 2*h*, 4*h* ve 8*h* seviyeli ağlar Şekil 4.6'da gösterilmiştir.



(a) *h* seviyeli çoklu ağ (b) 2*h* seviyeli çoklu ağ

(c) 4*h* seviyeli çoklu ağ (d) 8*h* seviyeli çoklu ağ

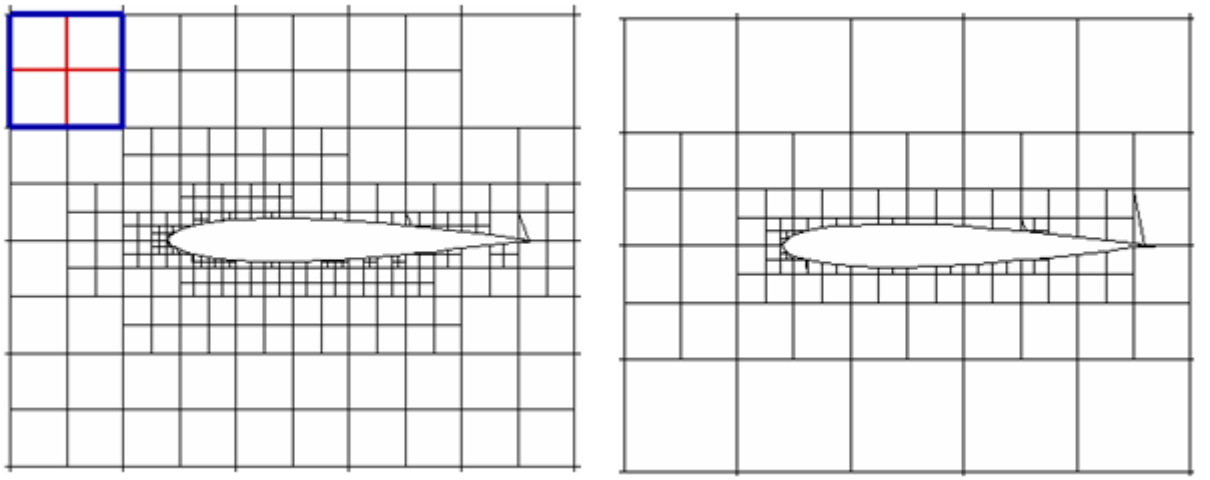
Şekil 4.6 Çoklu ağ seti

Sınırlama aşamasını daha detaylı anlatabilmek için *h* seviyeli ağdan 2*h* seviyeli ağa geçiş ve yapılan işlemler bu bölümde anlatılacaktır. Diğer geçişler de (2*h* seviyeliden 4*h* seviyeli ağa ve 4*h* seviyeli ağdan 8*h* seviyeli ağa geçiş) bu bölümde anlatılanın aynısıdır.

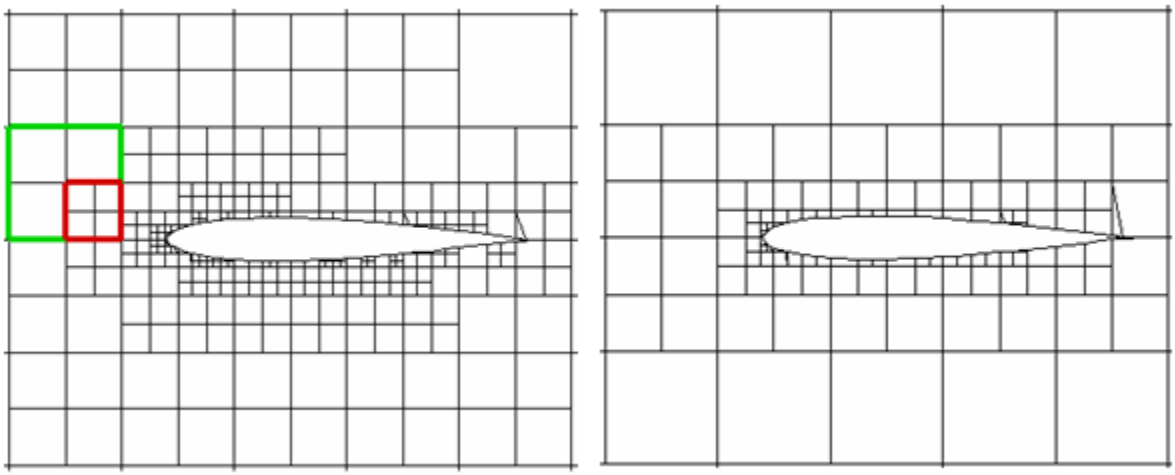
Geliştirilen yazılımda daha az yoğun ağların (2h, 4*h* ve 8*h* seviyeli ağlar) elde edilmesi oldukça karmaşıktır. Bu işlemin iki aşamada özetlenmesi mümkündür.

(i) Öncelikle bütün çocukları yaprak hücre olan ebeveyn hücreler seçilmektedir. Örneğin, Şekil 4.7a'daki *h* seviyeli ağda mavi renkli ebeveyn hücre, bütün çocukları yaprak hücre olduğundan ve bu hücre seyreltildiğinde bir seviye kuralını ihlal etmediğinden (bakınız 2. aşama) 2*h* seviyeli ağda yaprak hücre olarak görülmektedir. Fakat, Şekil 4.7b'deki *h* seviyeli ağda yeşil renkle gösterilen ebeveyn hücrenin

kırmızı renkle gösterilen dördüncü çocuğunun yaprak hücre olmamasından dolayı bu hücrenin seyreltilebilmesi için öncelikle dördüncü çocuğunun yaprak hücre olması gerekmektedir. Bu arada işaretlenen ebeveyn hücreleri belli etmek için önceden bahsedilen *perform* adlı işaretçi kullanılır.



(a)



(b)

Şekil 4.7 *h* ve 2*h* seviyeli çoklu ağlar

(ii) Eğer seçilen ebeveyn hücreler seyreltildiği zaman bir seviye kuralını ihlal etmiyorlarsa, seçilen bu ebeveyn hücreler daha az yoğun ağda yaprak hücre olmaktadır. Örneğin, Şekil 4.8a'daki *h* seviyeli ağda pembe ve yeşil renkle gösterilen ebeveyn hücrelerin bütün çocukları yaprak hücre olduğundan birinci aşamaya göre bu hücreler *perform* işaretçileri 1 olan seçilmiş ebeveyn hücrelerdir. Şekil 4.8b'de de

görüldüğü gibi seyreltildiğinde bir seviye kuralını ihlal etmeyen yeşil ebeveyn hücre 2*h* seviyeli ağda yaprak hücre olarak görülmektedir. Buna karşılık, seyreltildiğinde bir seviye kuralını ihlal eden pembe ebeveyn hücre 2*h* seviyeli ağda yine ebeveyn hücre olarak görülmektedir. Eğer bir seviye kuralı dikkate alınmayarak birinci aşamada seçilen tüm ebeveyn hücrelerin, 2*h* seviyeli ağda yaprak hücre olması durumunda, elde edilen çoklu ağın Şekil 4.8c'de görüldüğü gibi olması gerekmektedir.

İşaretlendikten sonra bir seviye kuralını da ihlal etmediği görülen ebeveyn hücreler seyreltilmiş ağda hesaplama hücresi olarak görünürler. Bu ebeveyn hücreleri hesaplama hücresi yapmak için bu hücrelerin çocuklarını silmek ve sıfır olarak depolamak yerine *compcell* isimli işaretçilerini 1 yapmak yeterli olmaktadır. Böylece daha az yoğun ağlardaki hesaplama hücreleri aranırken sadece hücrenin *compcell* işaretçisine bakmak yeterli olur.

2*h* seviyeli çoklu ağ elde edildikten sonra, aşağıdaki (4.79), (4.80) ve (4.81) numaralı denklemler sırasıyla kullanılarak bu ağı oluşturan yaprak hücrelerin ara çözüm, artakalan ve zorlayıcı fonksiyon değerleri hesaplanmaktadır.

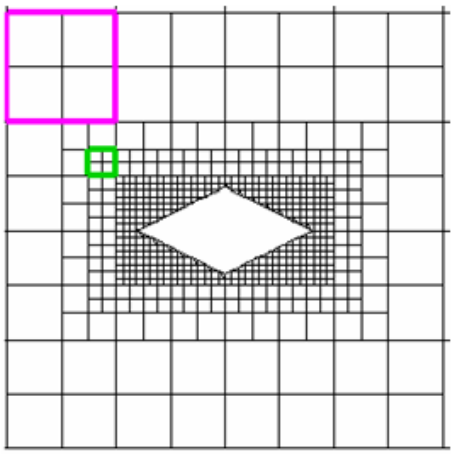
$$\mathbf{u}_0^{2h} = \mathrm{I}_h^{2h}\mathbf{u}_n^h = \begin{cases} \dfrac{\sum_{\text{çocuklar}} (A_{hücre}\,\mathbf{u}_n^h)}{\sum_{\text{çocuklar}} A_{\text{hücre}}} & h \text{ seviyeli agda ebeveyn hücre olanlar} \\ \mathbf{u}_n^h & h \text{ seviyeli agda yaprak hücre olanlar} \end{cases} \tag{4.79}$$

$$\hat{\mathrm{I}}_h^{2h}\left(\mathbf{R}\left(\mathbf{u}_n^h\right)\right) = \begin{cases} \sum_{\text{çocuklar}} \left(\mathbf{R}\left(\mathbf{u}_n^h\right)\right) & h \text{ seviyeli agda ebeveyn hücre olanlar} \\ \mathbf{R}\left(\mathbf{u}_n^h\right) & h \text{ seviyeli agda yaprak hücre olanlar} \end{cases} \tag{4.80}$$

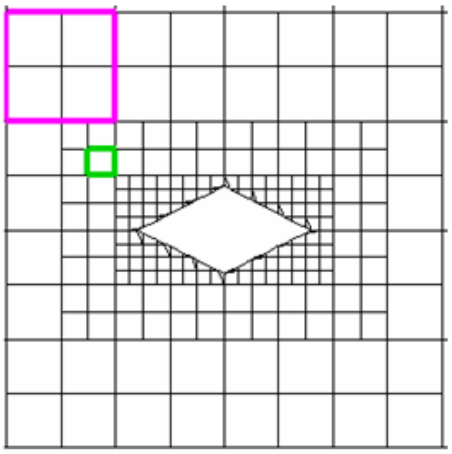
$$\mathbf{FF}^{2h} = \begin{cases} \hat{\mathrm{I}}_h^{2h}\left(\mathbf{R}\left(\mathbf{u}_n^h\right) + \mathbf{FF}^h\right) - \mathbf{R}\left(\mathbf{u}_0^{2h}\right) & h \text{ seviyeli agda ebeveyn hücre olanlar} \\ \left(\mathbf{R}\left(\mathbf{u}_n^h\right) + \mathbf{FF}^h\right) - \mathbf{R}\left(\mathbf{u}_0^{2h}\right) & h \text{ seviyeli agda yaprak hücre olanlar} \end{cases} \tag{4.81}$$

Yukarıdaki denklemlerde kullanılan I_h^{2h} ağırlıklı sınırlama operatörü ve $\hat{\mathrm{I}}_h^{2h}$ artakalan sınırlama operatörüdür.

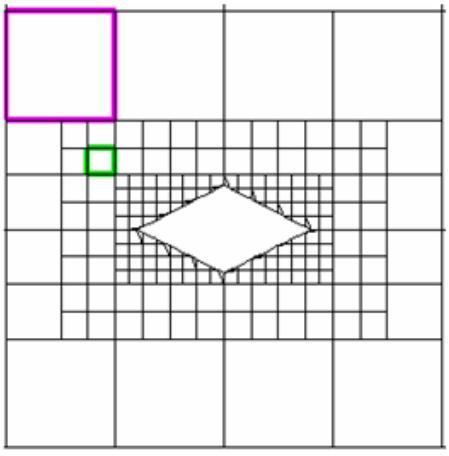
Şekil 4.5'te gösterilen çoklu ağ seti örnek alınacak olursa, sekiz kez ötelemeden sonra 2*h* seviyesindeki ağı oluşturan yaprak hücrelerin ara çözüm değerleri elde edilmektedir.



(a) *h* seviyeli çoklu ağ



(b) 2*h* seviyeli bir seviye kuralı dikkate alınarak yaratılan çoklu ağ



(c) 2*h* seviyeli bir seviye kuralına bakılmadan yaratılan çoklu ağ

Şekil 4.8 Bir çoklu ağın *h* seviyesinden *2h* seviyesine seyreltilmesi

$$\mathbf{u}_1^{2h} = \mathbf{u}_0^{2h} - \beta_1 \frac{\Delta t}{A_{hücre}} (\mathbf{R}(\mathbf{u}_0^{2h}) + \mathbf{FF}^{2h})$$
$$\text{.....} \qquad (4.82)$$
$$\mathbf{u}_n^{2h} = \mathbf{u}_0^{2h} - \beta_{n\text{-}1} \frac{\Delta t}{A_{hücre}} (\mathbf{R}(\mathbf{u}_{n\text{-}1}^{2h}) + \mathbf{FF}^{2h})$$

Son olarak, yapılan ötelemeler sonucunda elde edilen ara çözümler kullanılarak 2*h* seviyeli ağdaki tüm yaprak hücreler için artakalan değerleri, $\mathbf{R}(\mathbf{u}_n^{2h})$, hesaplanmaktadır.

(iii) Çoklu Ağ Yönteminin Üçüncü Aşaması: Uzatma

Sınırlama aşamasında daha az yoğun ağları oluşturan hücreler için $\mathbf{u}_n^{8h}$, $\mathbf{u}_n^{4h}$ ve $\mathbf{u}_n^{2h}$ değerleri bulunduktan sonra, bu değerlerin tekrar ağ aralığı *h* olan yoğun ağa aktarılmaları gerekmektedir. Bu işlem, uzatma operatörü kullanılarak yapılmaktadır. Örneğin, *h* seviyeli ağdaki yaprak hücrelerin yenilenmiş ara değerlerinin, 2*h* seviyeli ağdaki yaprak hücrelerin ara çözüm değerleri kullanılarak nasıl elde edildiği aşağıdaki denklemde gösterilmektedir.

$$\mathbf{u}_n^{h(\text{yeni})} = \mathbf{u}_n^h + \mathrm{I}_{2h}^h \left(\mathbf{u}_n^{2h(\text{yeni})} - \mathrm{I}_h^{2h}\, \mathbf{u}_n^h \right) \qquad (4.83)$$

Yukarıdaki denklemde kullanılan I_{2h}^h uzatma operatörüdür. İki farklı uzatma operatörü (gradyan operatörü ve enjekte operatörü) kullanılarak elde edilmiş çözümler, iki boyutlu Kartezyen ağ üreticisi ile Euler çözücüsünün test edilmesi bölümünde sunulmuştur. Bu operatörler sırasıyla

$$\mathrm{I}_{2h}^h \left(\mathbf{u}^{2h} \right) = \nabla \left(\mathbf{u}^{2h} \right) \bullet \mathrm{d}\mathbf{r} \qquad (4.84)$$

$$\mathrm{I}_{2h}^h \left(\mathbf{u}^{2h} \right) = \mathbf{u}^{2h} \qquad (4.85)$$

verilebilir. Ayrıca, iki farklı döngünün karşılaştırması da iki boyutlu Kartezyen ağ üreticisi ile Euler çözücüsünün test edilmesi bölümünde sunulmuştur. Bunlardan ilki V döngüsü, diğeri ise testere dişi (Saw-tooth cycle) döngüsüdür. Bu iki döngü arasında tek fark, V döngüsünün uzatma aşamasında da istenilen sayı kadar öteleme yapılıp, daha az yoğun ağlar için elde edilen yenilenmiş çözümlerin bir miktar daha düzeltilmesinin sağlanmasıdır. Bu aşamada da V döngüsü için ötelemeler, çoklu ağ yöntemi için adapte edilmiş çok kademeli zaman adımlama yöntemi kullanılarak yapılmaktadır. Bu yöntemde kullanılacak zorlayıcı fonksiyon değerleri, sınırlama aşamasındaki değerlerin aynısıdır. Bu değerlerin uzatma aşamasında da kullanılabilmesi için her ağ seviyesinde kullanılan tüm zorlayıcı fonksiyon değerlerinin depolanması gerekmektedir. Bu durumda, çok fazla verinin depolanması gerektiğinden ve testere dişi döngüsü ile V döngüsü arasında iyileştirmenin (iki boyutlu Kartezyen ağ üreticisi ile Euler çözücüsünün test edilmesi bölümü) birbirine çok yakın olmasından dolayı testere dişi döngüsünün kullanılması daha avantajlı olmaktadır.

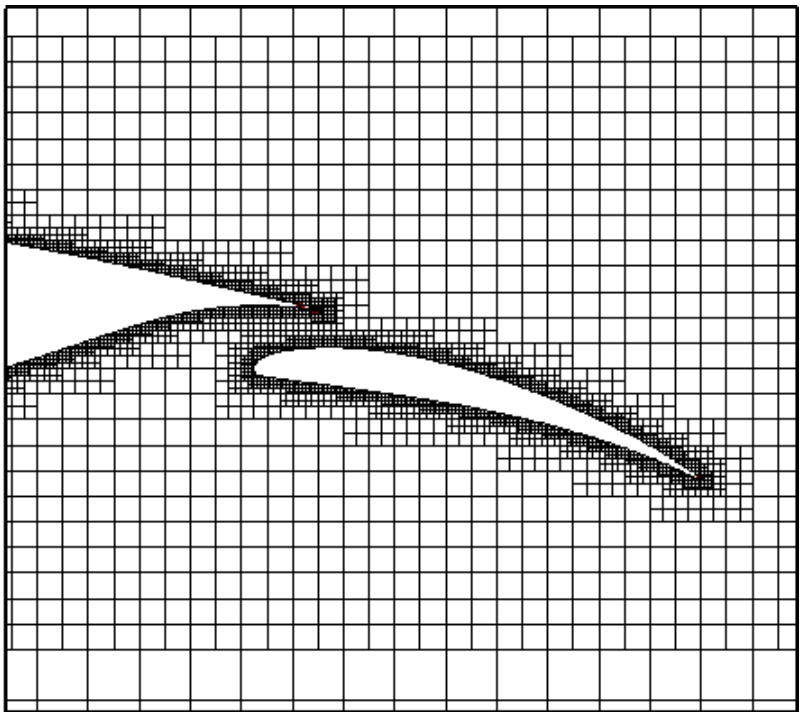
4.5.3 Çoklu Ağ Yönteminde Ayrık Hücrelerin Önemi

Literatürdeki bazı çalışmalarda ayrık hücrelerden kurtulmak için yaratılan ağda ayrık hücreler varsa tekrar bölünmeye gidilmiştir. Ayrık hücrelerden kurtulana kadar da bölünme devam etmiştir. Ama önceden de belrtildiği gibi ayrık hücreler çoklu ağ yöntemi için gereklidir. Bu bölümde niçin gerekli olduğu anlatılacaktır. Çoklu ağ yöntemi, çözümü hızlandırmak için değişik ağ seviyelerinde problemi çözüp kısa ve uzun dalga boyuna sahip hataları en kısa zamanda yok etme ilkesine dayanmaktadır ve bu yüzden yoğun ve seyrek ağlar kullanılmaktadır. Diğer bir deyişle, eğer yoğun ağ yaratma işleminde ayrık hücrelerle karşılaşıyorsa, bu hücrelerden tekrarlı bölünme işlemiyle kurtulmak bir çözüm değildir çünkü seyrek ağlarda bu ayrık hücreler tekrar karşımıza çıkacaktır.

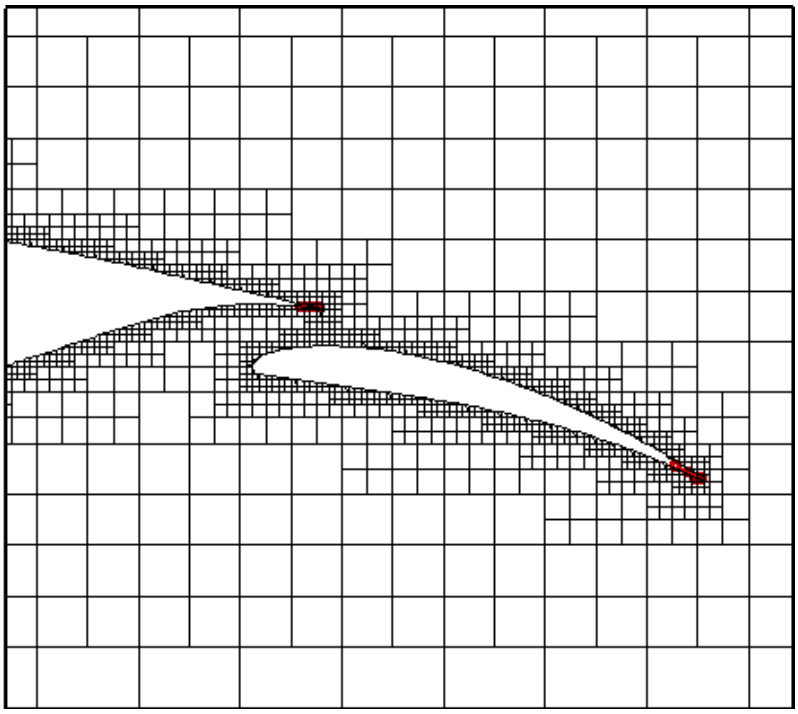
Bu durumu bir örnekle açıklamak gerekirse, Şekil 4.9'te iki elemanlı NLR 7301 kanatçığının çevresinde geometrik adaptasyon kullanılarak yaratılmış ağ görülmektedir ve bu şekilde kırmızı renkle gösterilmiş çok ufak, 5 adet ayrık hücre bulunmaktadır. Bu ayrık hücreler birinci ve ikinci kanatçığın arka kenarlarında (trailing edges) bulunmaktadır. Bu hücrelerden kurtulmak için literatürde bazı yöntemler geliştirilmiştir. Bunlardan biri, kanatçıkları modifiye etmektir. Kanatçıkları modifiye etmek demek, kanatçıkların ayrık hücre oluşan kısımlarını silmek yani kanatçığın şeklinde ufak değişiklik yapmaktır. Bu yöntemin, diğer seyrek ağ seviyelerinde de uygulanması demek, 32*h* seviyeli ağda ikinci kanatçığın tamamen ve birinci kanatçığın da arka kenarının büyük bir kısmının yok olması demektir. Bu durum 2*h*, 4*h*, 8*h*, 16*h* ve 32*h* seviyeli ağlarda sırasıyla Şekil 4.10, 4.11, 4.12, 4.13 ve 4.14'te gösterilmiş olup, hesaplamaya büyük zarar vermektedir. Geliştirilen yazılımda elde edilen 32*h* seviyeli ağ Şekil 4.'15te görülmektedir ve geometrideki kaybın sadece ikinci kanatçığın ön kenarı olduğu görülmektedir ve bu seviyede ve Şekil 4.15'teki seyrek ağ ile elde edilen sonuç çözümü çok fazla etkilememektedir.

4.6 YENİDEN YAPILANDIRMA

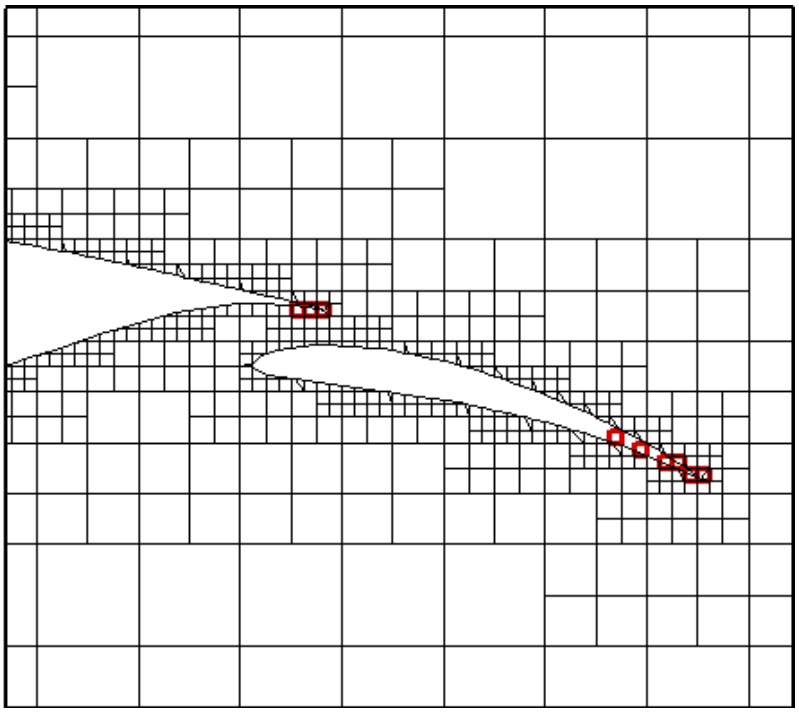
Hücrenin yüzünden geçen akının hesaplanabilmesi için bu yüzün sağ ve sol taraflarındaki durumlar için ilkel akış değişkenlerinin bulunması gerekir. Bunun için en kolay yaklaşım, sol durum için incelenen hücredeki değerlerin, sağ durum için ise komşu hücredeki değerlerin alınmasıdır. Ancak, bu yaklaşım çok da iyi sonuçlar vermemektedir. Bu nedenle hücredeki değişkenlerin değerlerinin yeniden yapılandırılması gerekmektedir. Başka bir deyişle sonlu hacim yönteminde hücredeki akış değişkenleri hücre merkezinde tanımlanmasına karşılık, bu akış değişkenlerinin hücre içerisindeki değişiminin göz önüne alınması istenmektedir. Yeniden yapılandırmada en önemli nokta ilgilenilen bölgede yeni maksimum noktaların ortaya çıkmamasıdır.



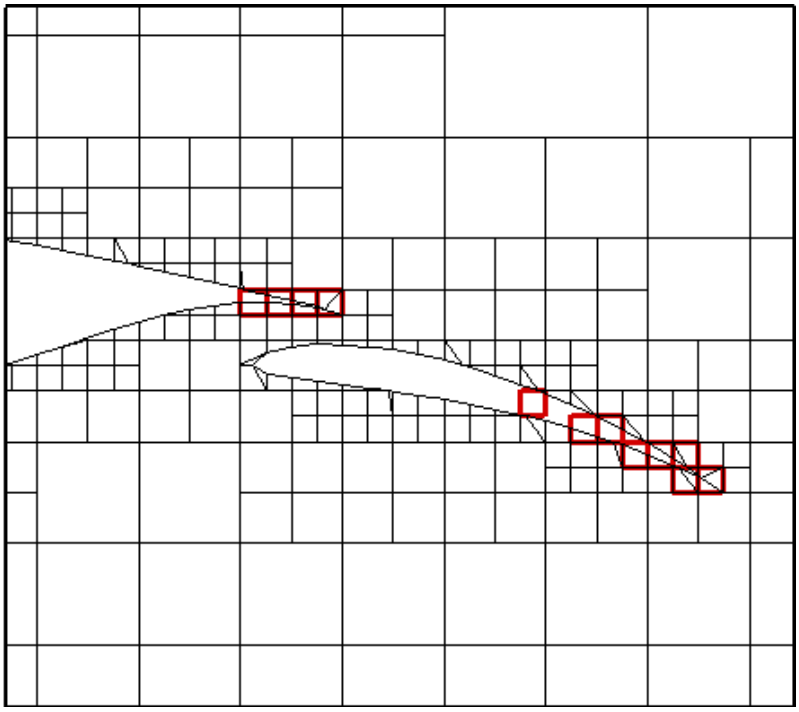
Şekil 4.9 İki elemanlı NLR 7301 kanatçığı çevresindeki *h* seviyeli çoklu ağ



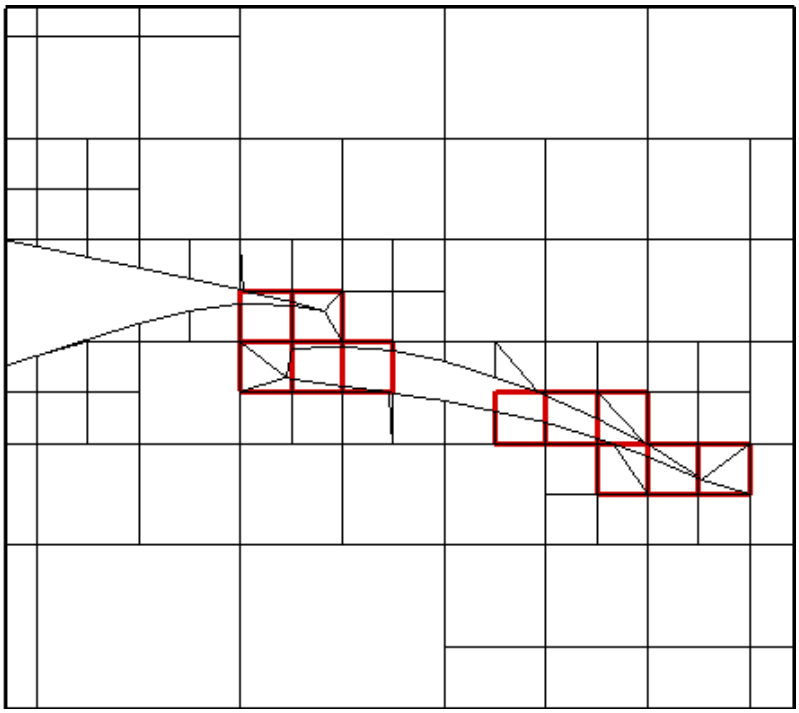
Şekil 4.10 İki elemanlı NLR 7301 kanatçığı çevresindeki 2*h* seviyeli çoklu ağ



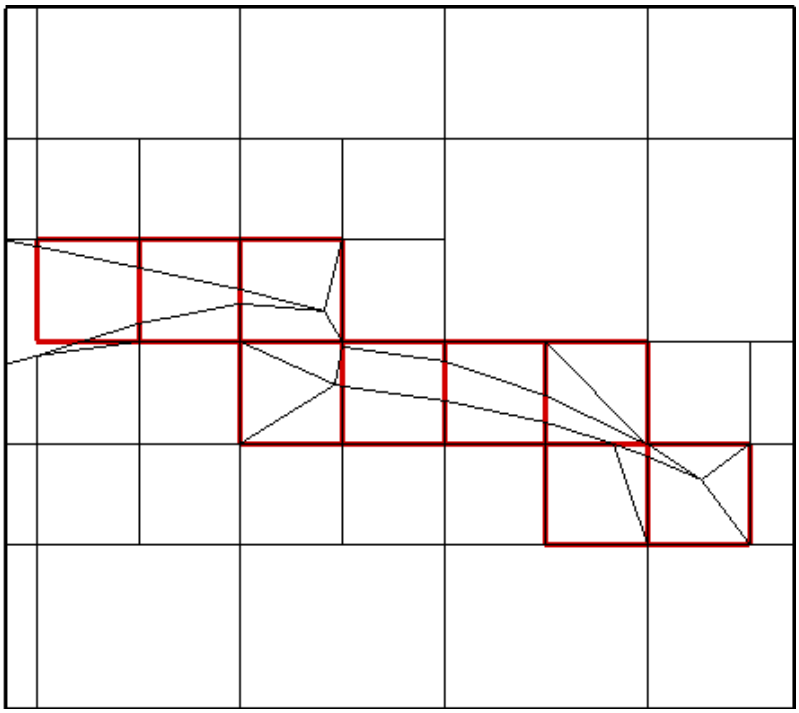
Şekil 4.11 İki elemanlı NLR 7301 kanatçığı çevresindeki 4*h* seviyeli çoklu ağ



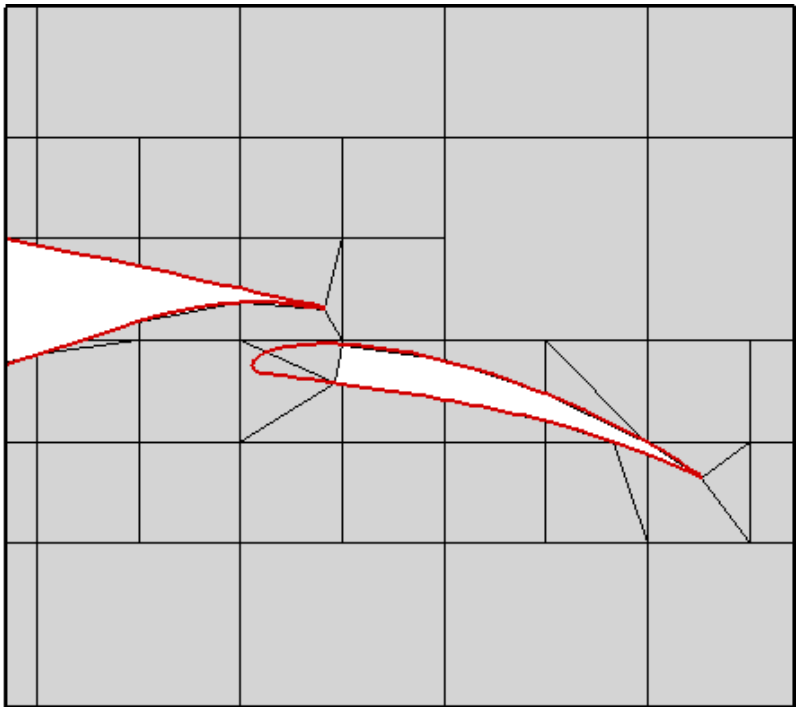
Şekil 4.12 İki elemanlı NLR 7301 kanatçığı çevresindeki 8*h* seviyeli çoklu ağ



Şekil 4.13 İki elemanlı NLR 7301 kanatçığı çevresindeki 16*h* seviyeli çoklu ağ



Şekil 4.14 İki elemanlı NLR 7301 kanatçığı çevresindeki 32*h* seviyeli çoklu ağ



Şekil 4.15 İki elemanlı NLR 7301 kanatçığı çevresindeki geliştirilen kodla elde edilen 32*h* seviyeli çoklu ağ

Literatürde üç değişik yeniden yapılandırma şemasının bulunduğu görülmüştür.. Bunlardan ikisinde akış değişkeninin gradyanı hücre merkezinde bulunmakta, diğerinde ise sonlu farklarla akış değişkeninin değeri hücrenin değişik noktalarında hesaplanmaktadır. Hücre merkezinde gradyanların bulunmasına dayanan şemalardan biri Green-Gauss yeniden yapılandırma yöntemi diğeri ise Ufak Kareler (Least Squares) veya Minimum Enerji yapılandırma yöntemidir. Bu yeniden yapılandırma şemaları için genel tasarım kriterleri Barth (1998) ile Barth ve Jespersen (1990) tarafından incelenmiştir. Doğrusal yeniden yapılandırmanın detayları Barth ve Fredericson (1989), Coirier (1994) ve De Zeeuw (1993) tarafından verilmesine karşılık, *k*'ıncı dereceden doğru şemalar Barth (1993) tarafından incelenmiştir. Sonlu farklara dayanan üçüncü şema ise Popinet (2003) tarafından açıklanmıştır.

4.6.1 Ufak Kareler (Minimum Enerji) Yeniden Yapılandırması

Basitçe ufak kareler yeniden yapılandırması, hücre ile alakalı bir değişken için ortalama fonksiyonun tersidir. Hücre içerisindeki *w* değişkeninin hücrenin herhangi bir noktasındaki değeri aşağıdaki denklem ile tespit edilir.

$$\mathbf{w} = \mathbf{w}_c + \mathbf{w}_{xc}(x - x_c) + \mathbf{w}_{yc}(y - y_c) + \mathbf{w}_{zc}(z - z_c) \tag{4.86}$$

$$\mathbf{w} = \begin{pmatrix} \rho \\ u \\ v \\ w \\ p \end{pmatrix} \tag{4.87}$$

Bu denklemdeki bilinmeyenler sadece hücre merkezindeki *w* değişkeninin gradyan bileşenleri olan w_{xc}, w_{yc} ve w_{zc}'dir. *w* değişkeni için doğrusal polinom elde edildiğinde, w_{xc}, w_{yc} ve w_{zc} bilinmeyenlerini bulmak için destek kümesini oluşturan hücrelerden meydana gelen bölge içerisinde, *w* değişkeninin gerçek değeri ile yapılandırma polinomundan elde edilen değer arasındaki farkın karelerinin toplamı ufaltılır.

$$\mathbf{L_{ij}}(\nabla \mathbf{w_j}) = \mathbf{B_i} \tag{4.88}$$

$$\nabla \mathbf{w} = \begin{bmatrix} \mathbf{w}_{xc} \\ \mathbf{w}_{yc} \\ \mathbf{w}_{zc} \end{bmatrix} \tag{4.89}$$

$$\mathbf{L} = \begin{bmatrix} \sum_{i=1}^{nKomşKomş} (x_{n,c} - x_c)_i (x_{n,c} - x_c)_i & \sum_{i=1}^{nKomşKomş} (x_{n,c} - x_c)_i (y_{n,c} - y_c)_i & \sum_{i=1}^{nKomşKomş} (x_{n,c} - x_c)_i (z_{n,c} - z_c)_i \\ \sum_{i=1}^{nKomşKomş} (x_{n,c} - x_c)_i (y_{n,c} - y_c)_i & \sum_{i=1}^{nKomşKomş} (y_{n,c} - y_c)_i (y_{n,c} - y_c)_i & \sum_{i=1}^{nKomşKomş} (y_{n,c} - y_c)_i (z_{n,c} - z_c)_i \\ \sum_{i=1}^{nKomşKomş} (x_{n,c} - x_c)_i (z_{n,c} - z_c)_i & \sum_{i=1}^{nKomşKomş} (z_{n,c} - z_c)_i (y_{n,c} - y_c)_i & \sum_{i=1}^{nKomşKomş} (z_{n,c} - z_c)_i (z_{n,c} - z_c)_i \end{bmatrix} \tag{4.90}$$

$$\mathbf{B} = \begin{bmatrix} \sum_{i=1}^{nKomşKomş} (\mathbf{w}_{n,c} - \mathbf{w}_c)_i (x_{n,c} - x_c)_i \\ \sum_{i=1}^{nKomşKomş} (\mathbf{w}_{n,c} - \mathbf{w}_c)_i (y_{n,c} - y_c)_i \\ \sum_{i=1}^{nKomşKomş} (\mathbf{w}_{n,c} - \mathbf{w}_c)_i (z_{n,c} - z_c)_i \end{bmatrix} \tag{4.91}$$

(4.91) numaralı denklemdeki matris, hücrenin içerisinde bulunan herhangi bir konumdaki *w* değişkeninin değerinin elde edilebildiği hücrelerin merkezlerindeki gradyanları elde etmek için çözülür. Ufak kareler yöntemi, geliştirilen yazılımda yeniden yapılandırma için kullanılmıştır.

4.6.2 Gradyan Limitleme

Akış değişkenlerinin gradyanı yapılandırılan hücrelerde ve destek hücreleri tarafından oluşturulan bölge içerisinde yeni maksimum noktaların oluşturulması istenmez. Bu yüzden gradyanların, yeni maksimum noktalarının oluşmasını engellemek için, sınırlayıcı ile çarpılması gerekir. Bu durumun uygulanması ile doğrudan yeniden yapılandırma aşağıdaki denklemde gösterildiği gibi olur.

$$\mathbf{w} = \mathbf{w}_c + \Phi\left[\frac{d\mathbf{w}}{dx}(x - x_c) + \frac{d\mathbf{w}}{dy}(y - y_c) + \frac{d\mathbf{w}}{dz}(z - z_c)\right] \tag{4.92}$$

Sınırlayıcının Φ değeri 0 ile 1 arasındadır. Sınırlayıcının tam değerini belirlemek için *w* değişkeninin maksimum ve minimum değerleri hücre merkezinde bulunur.

$$\mathbf{w}^{\max} = \max(\mathbf{w}_c, \mathbf{w}_{ic}) \qquad i = 1,....,N_{sup} \tag{4.93}$$

$$\mathbf{w}^{\min} = \min(\mathbf{w}_c, \mathbf{w}_{ic}) \qquad i = 1,....,N_{sup} \tag{4.94}$$

Daha sonra, yeniden yapılandırılmış polinomun değerleri, hücre köşelerinde hesaplanır. Çünkü bu mevkiler doğrusal yeniden yapılandırma için hücre içerisindeki maksimum ve minimum değerlere sahip olacaklardır. Merkez değerinden en çok sapan *w* değişkeninin değeri w^1_{cor} olarak işaretlenmesi gerekir.

$$\Phi = \begin{cases} \min\left(1, \dfrac{\mathbf{w}^{\max} - \mathbf{w}_c}{\mathbf{w}^1_{cor} - \mathbf{w}_c}\right) & \mathbf{w}^1_{cor} - \mathbf{w}_c > 0 \\ \min\left(1, \dfrac{\mathbf{w}_c - \mathbf{w}^{\min}}{\mathbf{w}_c - \mathbf{w}^1_{cor}}\right) & \mathbf{w}^1_{cor} - \mathbf{w}_c < 0 \\ 1 & \mathbf{w}^1_{cor} - \mathbf{w}_c = 0 \end{cases} \tag{4.95}$$

4.6.3 Çözüm Adaptasyonu

Belirli bir seviyede yakınsama elde edildiğinde, hücrelerin çözüm gradyanına göre adaptasyonları yapılır. Bu adaptasyonun amacı, şoklar yüzünden oluşabilecek devamsızlık bölgelerinde ve çözümde büyük gradyanların oluştuğu bölgelerde (durgunluk noktasının çevresinde) yeterli çözünürlüğü elde etmek ve gereksiz çözünürlüğün olduğu yerleri seyreltmektir.

Kullanılan çözüm adaptasyonu kriteri, Hunt'da (2004) özetlenen kritere dayanmaktadır. Bu yöntem, hücrenin karakteristik uzunluğu ile çarpılan hızın gradyanına, hücrenin karakteristik uzunluğu ile çarpılan hızın dönümüne ve hücrenin karakteristik uzunluğu ile çarpılan entropi dalgasının gücüne dayanır. Bu durumda çözüm adaptasyonu için

$$\tau_{\mathbf{D}} = \left|\nabla \bullet \mathbf{v}\right|\Omega^{0.5} \tag{4.96}$$

$$\tau_{\mathbf{C}} = \left|\nabla \times \mathbf{v}\right|\Omega^{0.5} \tag{4.97}$$

ve

$$\tau_{\mathbf{EW}} = \left|\nabla p - c^2 \nabla \rho\right|\Omega^{0.5} \tag{4.98}$$

tanımlanır. (4.97), (4.98) ve (4.99) numaralı denklemlerle tanımlanan değişkenlerin standart sapmaları hesaplanır.

$$\sigma_\alpha = \sqrt{\frac{\sum_{i=1}^{nHücreler} (\tau_\alpha)_i^2}{nHücreler}} \tag{4.99}$$

Bu standart sapmayı aşan hücrelerin, $(\tau_\alpha)_i > \sigma_\alpha$, yoğunlaştırma adaptasyonları yapılır. Bu standart sapmanın onda birinin altında kalan hücrelerin, $(\tau_\alpha)_i < 0.1\sigma_\alpha$, seyreltme adaptasyonları yapılır.

BÖLÜM 5

SONUÇLAR

5.1 GİRİŞ

Bu bölümde, geliştirilen iki ve üç boyutlu kodlar kullanılarak elde edilen sonuçlar literatürdeki diğer numerik sonuçlarla veya deney sonuçları ile kıyaslanacak ve programın doğruluğu bu şekilde ispatlanmaya çalışılacaktır.

5.2 İKİ BOYUTLU KARTEZYEN AĞ ÜRETİCİSİ İLE EULER ÇÖZÜCÜSÜNÜN TEST EDİLMESİ

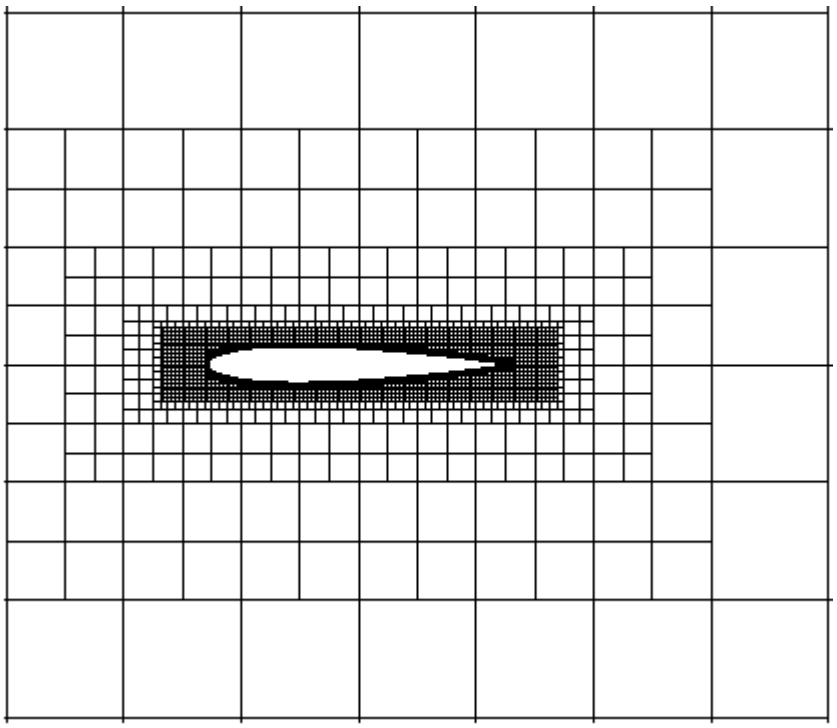
5.1.1 NACA0012 Kanat Profili Çevresinde Ses Civarı Akış

Geliştirilen iki boyutlu Euler çözücüsünün doğrulanmasına NACA 0012 kanadı ile başlanılmıştır. Ses civarı akış için NACA 0012 kanat profili etrafında gerçekleştirilen testlerde uzak alandaki Mach sayısı 0.85, hücum açısı ise 1^0 olarak alınmıştır. Böyle bir akışta, kanadın alt ve üst yüzeylerinde şok oluşması ve üst yüzeydeki şokun alt yüzeydekine göre çok daha kuvvetli olması beklenmektedir. Bu çalışmalardaki en kritik nokta şokun yeridir. Ses civarı akış, ses altı akışa göre çok daha karmaşık bir yapıda olduğu için çözüm adaptasyonlu ve çözüm adaptasyonsuz durumların karşılaştırılması ses civarı akış için gerçekleştirilmiştir. Bu testler için altı seviyeli çoklu ağ kullanılmıştır. Elde edilen sonuçlar aşağıdaki tabloda ve şekillerde görülmektedir. Sayısal sonuçları doğrulayabilmek için AGARD (1986) verileri kullanılmıştır.

Tablo 5.1 NACA 0012 çevresindeki ses civarı akışta (M_∞ = 0.85 ve α = 1) elde edilen sonuçlar

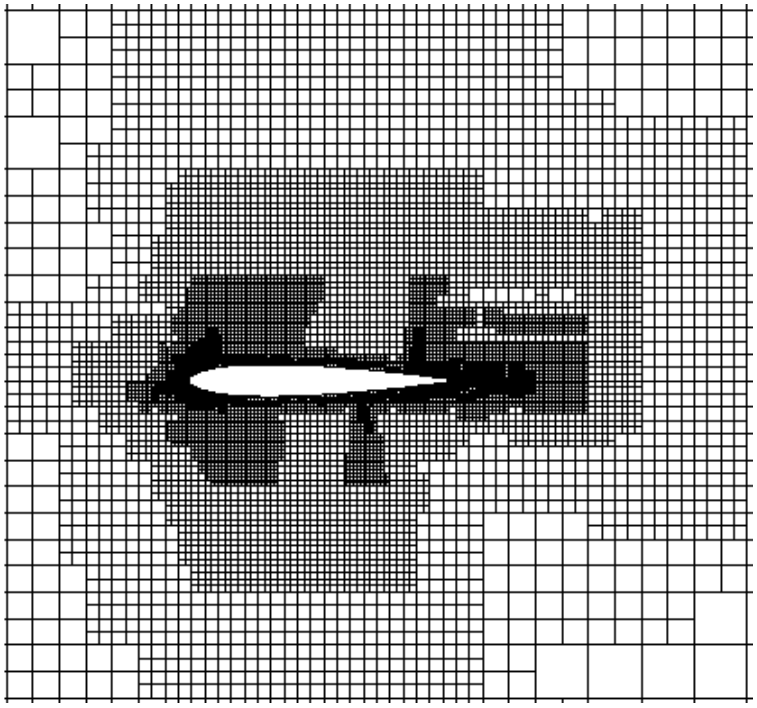
	C_L	C_D	Hücre sayısı	Zaman (s)
AGARD	0.3584	0.058	20480	-
Çözüm adaptasyonlu	0.3219	0.0611	18641	1012
Çözüm adaptasyonsuz	0.2361	0.0763	3538	68

Tablo 5.1'de de görüldüğü gibi, literatüreki referans kaldırma ve sürüklenme katsayılarına çözüm adaptasyonlu ile elde edilen sonuçlar daha yakındır. Çözüm adaptasyonlu durum, kaldırma katsayısını % 10 kadar az, sürüklenme katsayısını da % 5 kadar çok olarak hesaplamıştır. Çözüm adaptasyonunun ağ üzerindeki faydasını görebilmek için çözüm adaptasyonsuz ve adaptasyonlu ağ sırasıyla Şekil 5.1 ve 5.2'te verilmektedir. Şekil 5.1'deki ağa 3 kez çözüm adaptasyonu uygulandığında Şekil 5.2'deki ağ elde edilir.

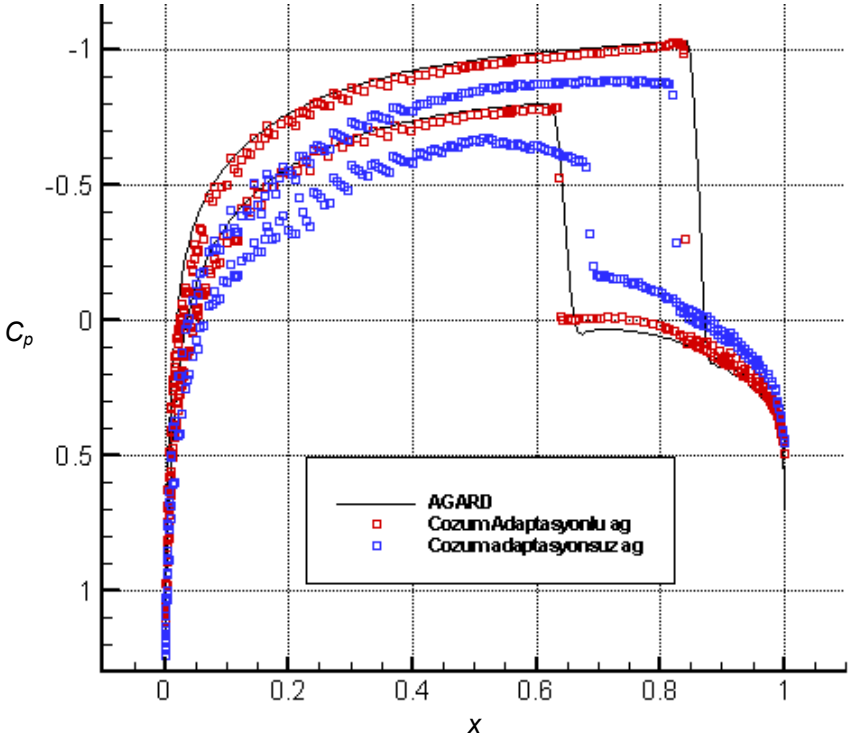


Şekil 5.1 NACA 0012 çevresindeki çözüm adaptasyonsuz ağ

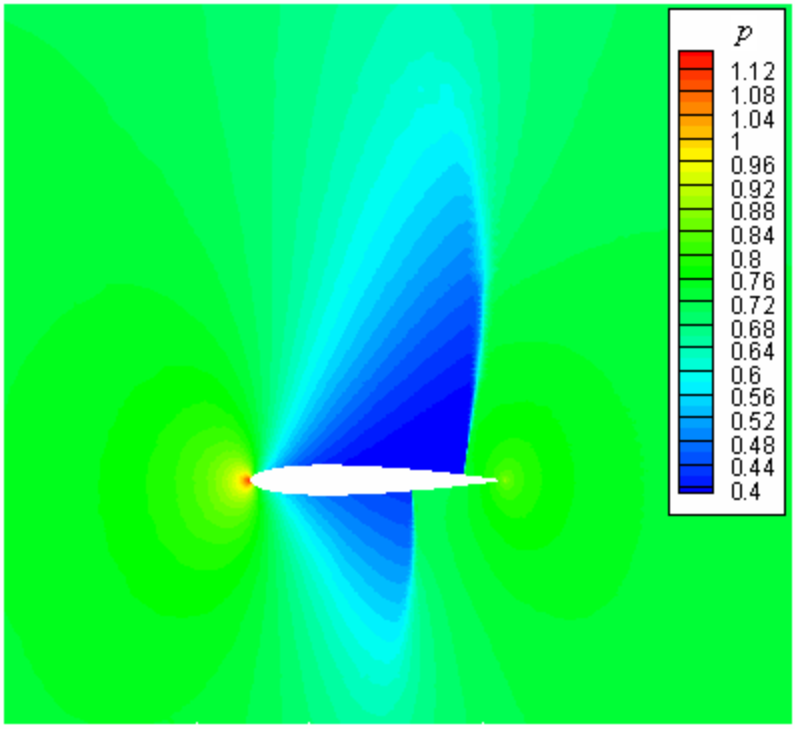
Basınç katsayısının dağılımı Şekil 5.3'te, çözüm adaptasyonlu basınç eş eğrileri Şekil 5.4'te ve çözüm adaptasyonsuz basınç eş eğrileri Şekil 5.5'de görülmektedir. Çözüm adaptasyonlu Mach eş eğrileri Şekil 5.6'da ve AGARD (1986)'dan alınan Mach eş eğrileri ise Şekil 5.7'de görülmektedir. Şekillerden de görüldüğü gibi çözüm adaptasyonlu ağ kullanılarak alt ve üst yüzeylerde oluşan şokların yerleri %1 sapmayla tespit edilmiştir.



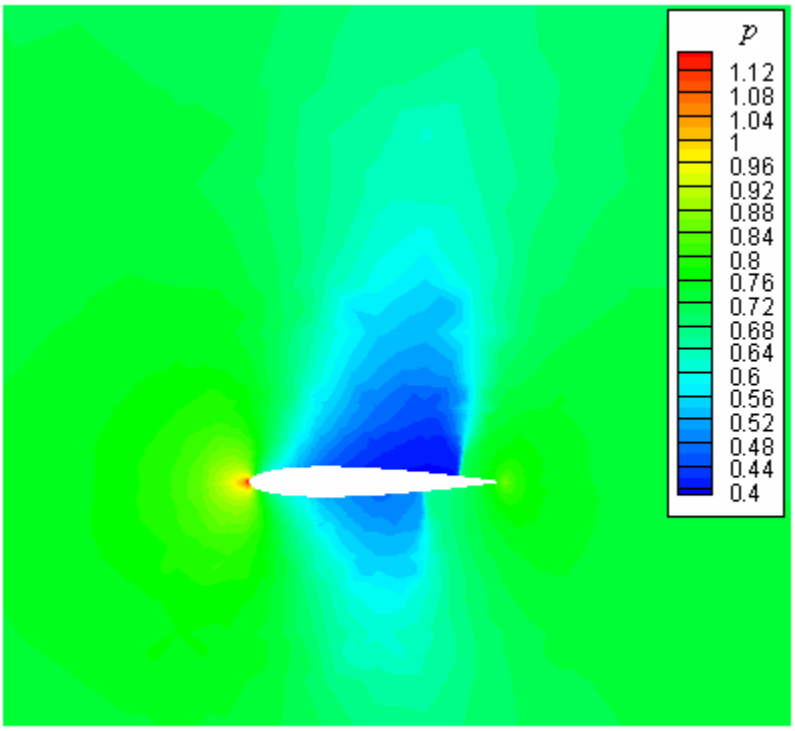
Şekil 5.2 NACA 0012 çevresindeki çözüm adaptasyonlu ağ



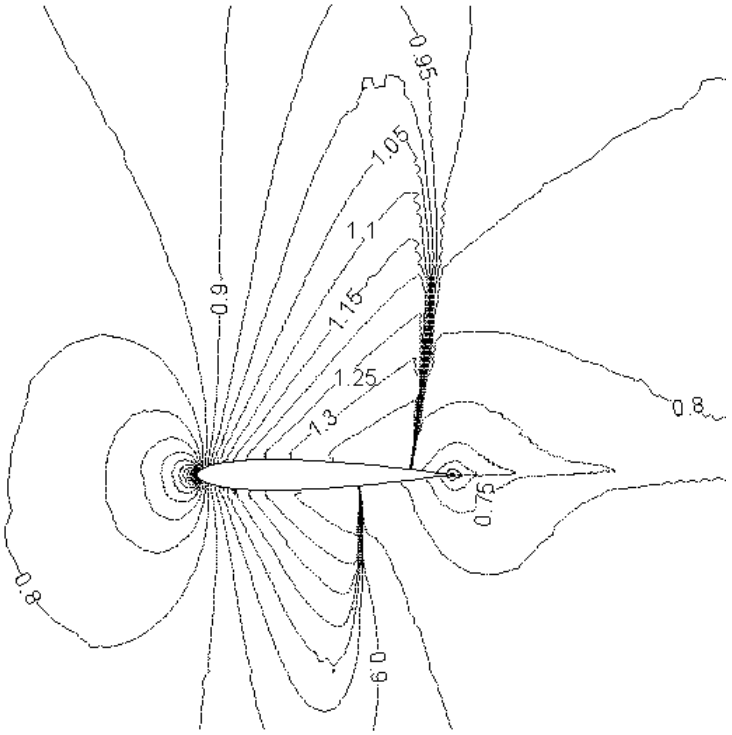
Şekil 5.3 NACA 0012 çevresindeki basınç katsayısı dağılımı



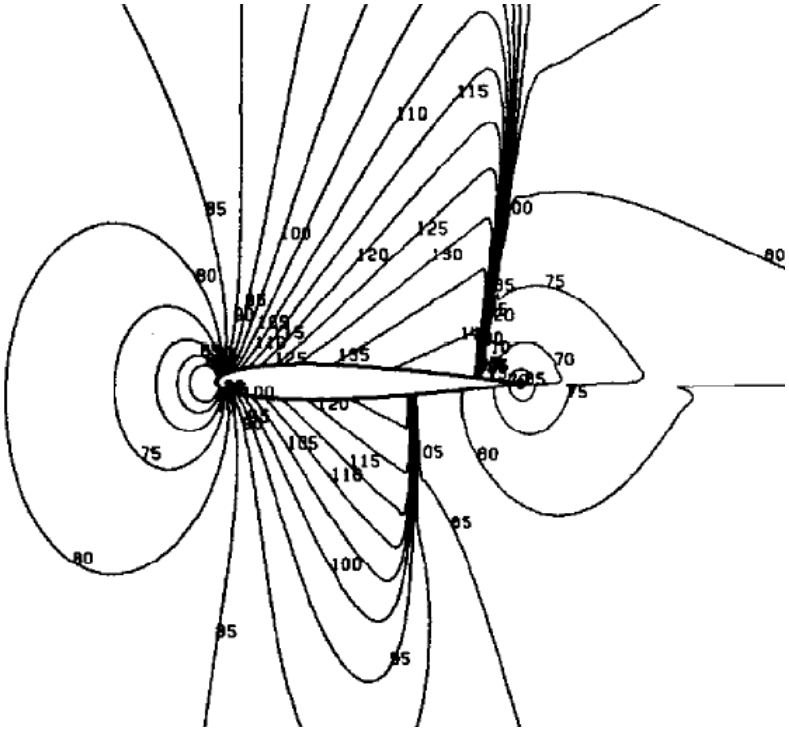
Şekil 5.4 NACA 0012 çevresindeki çözüm adaptasyonlu ağ kullanılarak elde edilen basınç eş eğrileri



Şekil 5.5 NACA 0012 çevresindeki çözüm adaptasyonsuz ağ kullanılarak elde edilen basınç eş eğrileri



Şekil 5.6 NACA 0012 çevresindeki çözüm adaptasyonlu ağ kullanılarak elde edilen Mach eş eğrileri



Şekil 5.7 AGARD (1986)'tan alınan Mach eş eğrileri

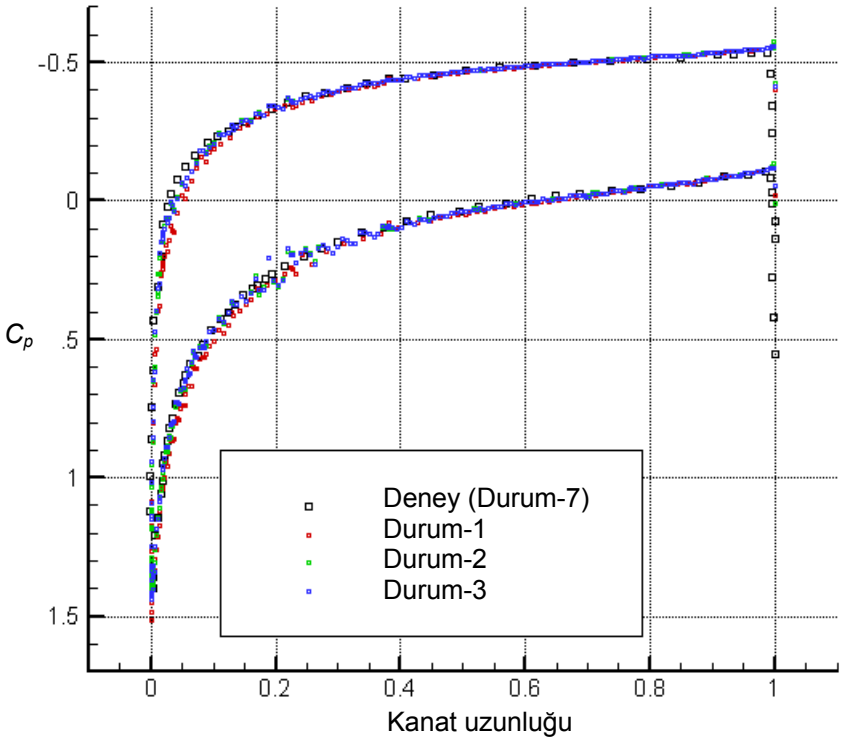
5.2.2 NACA 0012 Kanat Profili Çevresindeki Ses Üstü Akış

İkinci problem ise NACA 0012 kanat profili etrafındaki 1.2 Mach sayısında ve 7^0 hücum açısındaki akıştır. Bu test durumu geliştirilen çözücünün yay ve eğik şokları doğru olarak yakalayıp yakalamadığının anlaşılması için gerçekleştirilmiştir. Akıların hesaplanması için birinci ve ikinci dereceden şemalar kullanılmıştır.

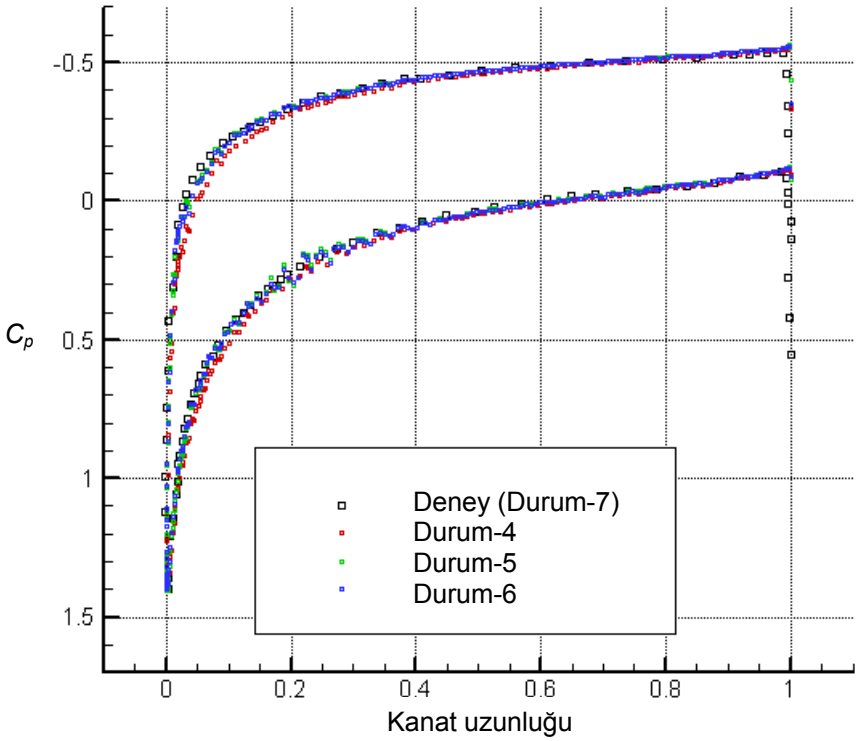
Bu bölümdeki bütün test durumları için uzak alan sınırı yaklaşık olarak kanattan 15 kanat uzunluğu kadar uzakta alınmıştır. Geliştirilen yazılımdaki yinelemelere yoğunluktaki ortalama artık 10^{-15} olana kadar devam edilmiştir. İkinci dereceden şemaların kullanıldığı bazı çözümlerde çoklu ağ yöntemi kullanılmamıştır. Tablo 5.2'de geliştirilen çözücü ile elde edilen ve AGARD (1986)'da verilen kaldırma katsayıları, sürüklenme katsayıları, hücre sayıları ve yakınsama zamanları verilmiştir. Bu tablodan görüldüğü gibi hesaplanan değerler ile AGARD (1986) verilen değerler arasında uyum olduğu görülmektedir. Örnek olarak üçüncü durumda kaldırma katsayısı % 0.4, sürüklenme katsayısı ise % 0.8 kadar fazla bulunmaktadır. Bu durumlar için basınç katsayısı dağılımları Şekil 5.8 ve 5.9'da verilmiştir. Basınç katsayısı dağılımlarının birbirlerine çok yakın olmakla birlikte Şekil 5.10'da verilen basınç eş eğrilerinde gözle görülebilen farklılıklar bulunmaktadır.

Tablo 5.2 NACA 0012 çevresindeki ses civarı akışta (M_∞ = 1.2 ve $\alpha = 7^0$) elde edilen sonuçlar

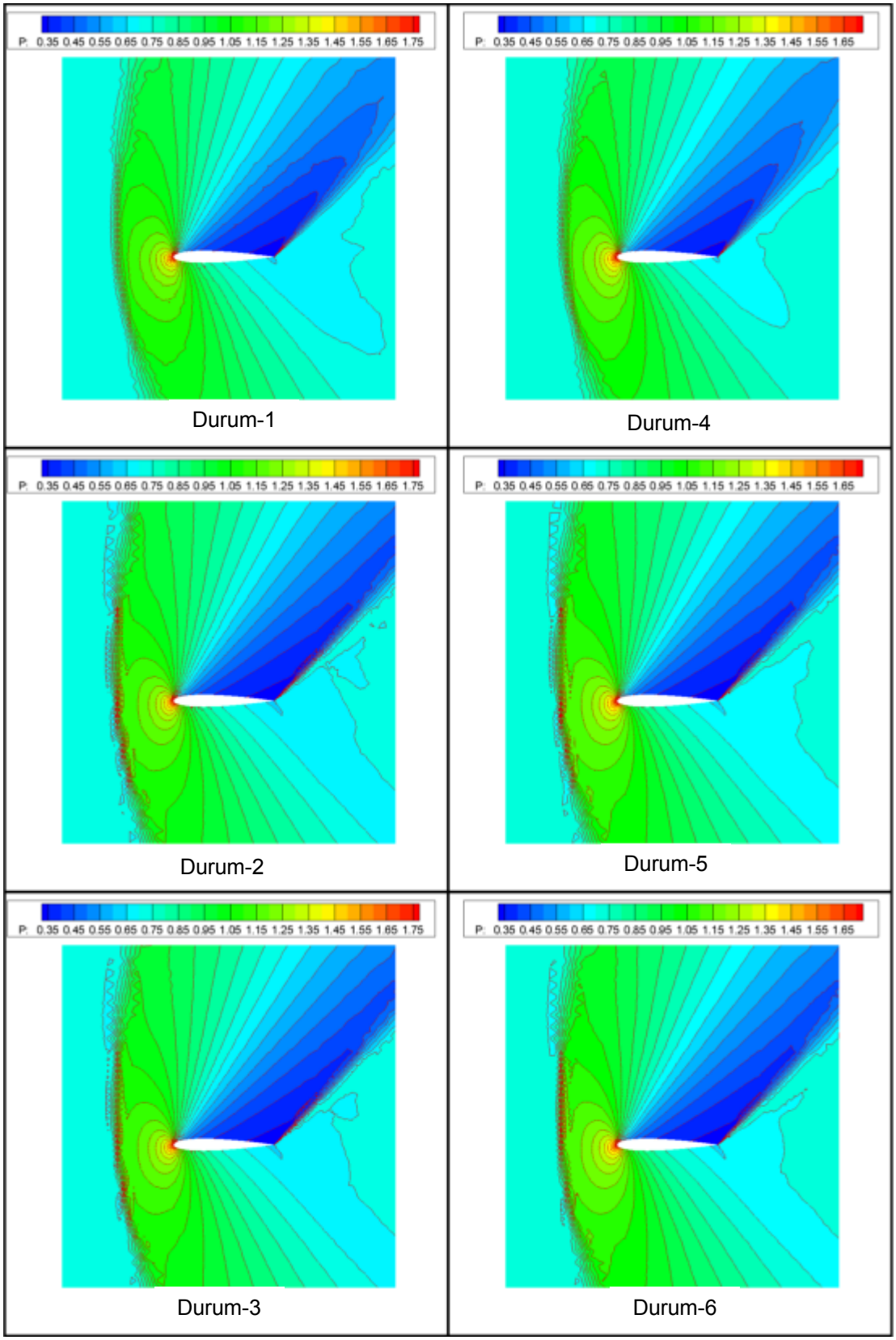
Durum #	**Test durumlarının tanımlanması**	C_L	C_D	**Hücre #**	**Zaman (s)**
Durum 1	AUSM ile birinci derece şema	0.5236	0.1620	11761	887
Durum 2	AUSM ile sınırlayıcısız ikinci derece şema	0.5216	0.1560	13784	23241
Durum 3	AUSM ile sınırlayıcılı ikinci derece şema	0.5217	0.1556	14081	16515
Durum 4	Roe'nun akı farkı yöntemi ile birinci derece şema	0.5201	0.1614	11717	1335
Durum 5	Roe'nun akı farkı yöntemi ile sınırlayıcısız ikinci derece şema	0.5219	0.156	13561	32211
Durum 6	Roe'nun akı farkı yöntemi ile sınırlayıcılı ikinci derece şema	0.5227	0.1555	13796	18733
Durum 7	AGARD (1986)	0.5196	0.1543	10209	-



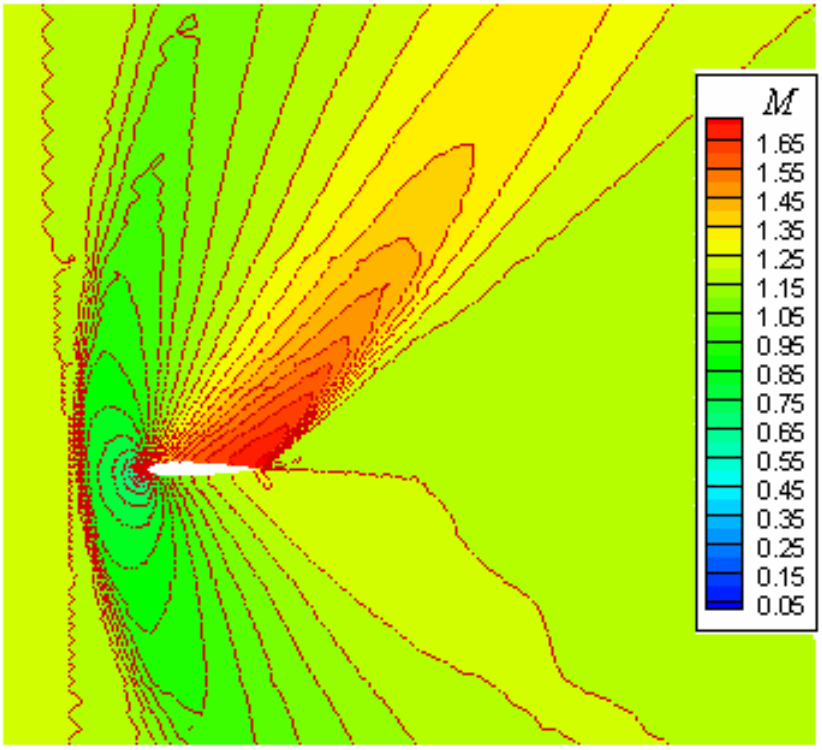
Şekil 5.8 NACA0012 kanat profili çevresinde $M_\infty = 1.2$ and $\alpha = 7°$ şartlarında birinci, ikinci ve üçüncü durumlar için basınç katsayısı dağılımları



Şekil 5.9 NACA0012 kanat profili çevresinde $M_\infty = 1.2$ and $\alpha = 7°$ şartlarında dördüncü, beşinci ve altıncı durumlar için basınç katsayısı dağılımları



Şekil 5.10 NACA0012 kanat profili çevresinde M_∞ = 1.2 and α = 7° şartlarında basınç eş eğrileri



Şekil 5.11 NACA0012 kanat profili çevresinde $M_\infty = 1.2$ and $\alpha = 7°$ şartlarında birinci durum için Mach eş eğrileri

Basınç eş eğrilerindeki farklılıklar, ikinci derece şemelarda ve özellikle de sınırlayıcısız olanlarda kirlilikten kaynaklanmaktadır. Şekşl 5.10 ve 5.11'den görüldüğü gibi, kanadın hücum kenarından önce yay şeklinde kuvvetli bir şok bulunmakta olup, bu şokun şiddeti çözüm alanının uzak bölgelerinde azalmaktadır. İkinci derece şemelar kullanılarak elde edilen çözümlerde şokun yakalanabilmesine karşılık çıban kararsızlığı (carbuncle instability) adı verilen bir kirlilik oluşmaktadır. Bu kararsızlık ikinci ve beşinci durumlarda görüldüğü gibi yay şeklindeki şokun önünde bulunan bölgeyi etkilemektedir. Sınırlayıcılar çözümdeki dalgalanmaları azalttığı için sınırlayıcıların bulunmaması daha fazla kirliliğe neden olmaktadır. Yay şeklindeki şok, kanat hücm kenarından yaklaşık olarak kanat uzunluğunun % 55'i kadar uzakta yer almaktadır,

5.2.3 RAE 2822 Kanat Profili Çevresinde Ses Civarı Akış

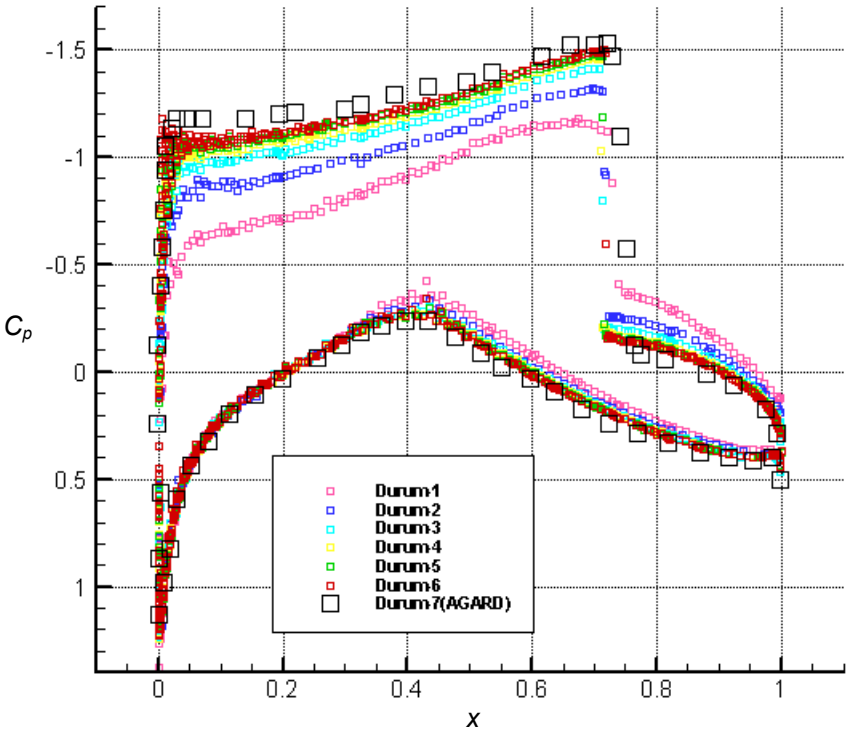
RAE 2822 profili etrafında gerçekleştirilen testlerde uzak alandaki Mach sayısı 0.75, hücum açısı ise 3^0 olarak alınmıştır. Bu testin amacı, simetrik olmayan kanat profilleri üzerindeki akışı incelemekir. Ayrıca, bu akış ile çözüm adaptasyon çevrim sayısının çözümün doğruluğuna sağladığı etkide incelenmektedir. Bu testler için altı seviyeli çoklu ağ kullanılmıştır. Elde edilen sonuçlar aşağıdaki tabloda ve şekillerde görülmektedir. Sayısal sonuçları doğrulayabilmek için AGARD (1986) verileri kullanılmıştır.

Tablo 5.3 RAE2822 çevresindeki ses civarı akışta (M_∞ = 0.75 ve α = 3) elde edilen sonuçlar

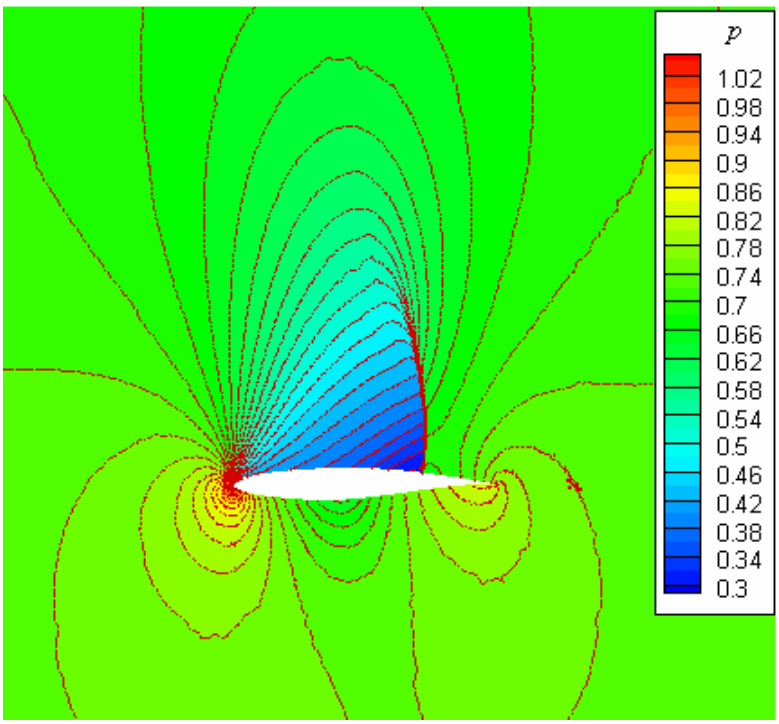
Durum #	Test durumlarının tanımlanması	C_L	C_D	Hücre #	Zaman (s)
Durum 1	Çözüm adaptasyonsuz ağ	0.7446	0.0705	2308	36
Durum 2	1 çevrimli Çözüm adaptasyonlu ağ	0.8573	0.0546	4848	52
Durum 3	2 çevrimli Çözüm adaptasyonlu ağ	0.9214	0.0476	10029	163
Durum 4	3 çevrimli Çözüm adaptasyonlu ağ	0.9538	0.046	18492	428
Durum 5	4 çevrimli Çözüm adaptasyonlu ağ	0.9731	0.0446	32268	1477
Durum 6	5 çevrimli Çözüm adaptasyonlu ağ	0.9881	0.0444	54254	3478
Durum 7	AGARD	1.1044	0.0448	20480	-

Tablo 5.3'de de görüldüğü gibi, elde edilen katsayıların, literatüreki referans kaldırma ve sürüklenme katsayılarına yakınsaması çevrim sayısı ile doğru orantılıdır. En iyi sonuç 5 çevrimli çözüm adaptasyonundan elde edilmiştir. Bunun nedeni çevrim sayısı arttıkça kritik olan bölgelerde ağın giderek daha yoğun hale gelmesidir. Fakat, yapılan testlerde görülmüştür ki, çevrim sayısı beşi geçtiğinde referans çözüme yakınsama azalmakta iken çözümün yakınsama süresi çok fazla artmaktadır. Bu yüzden, çevrim sayısı için optimum değer bu test için beştir. Basınç katsayısının dağılımı Şekil 5.12'de görülmektedir.

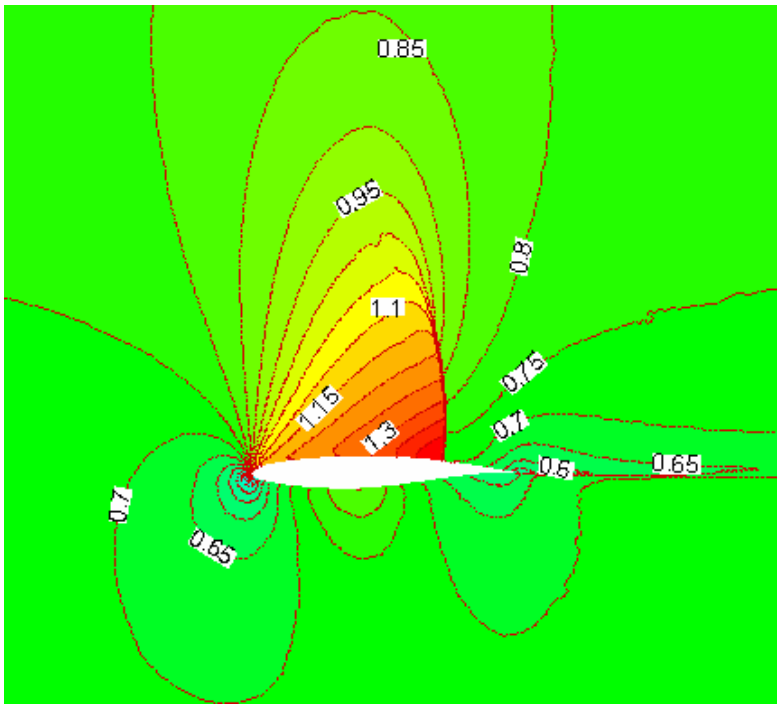
5 çevrimli çözüm adaptasyonunun basınç ve Mach eş eğrileri sırasıyla Şekil 5.13 ve 5.14'de görülmektedir. Şekil 5.14'de görüldüğü gibi kanatçığın üst yüzeyinde şoktan hemen önce elde edilen Mach sayısı 1.4'tür. Ayrıca üst yüzeydeki şokun yeride doğru olarak tespit edilmiştir. 5 çevrimli çözüm adaptasyonundan sonra elde edilen ağ Şekil 5.15'de görülmektedir.



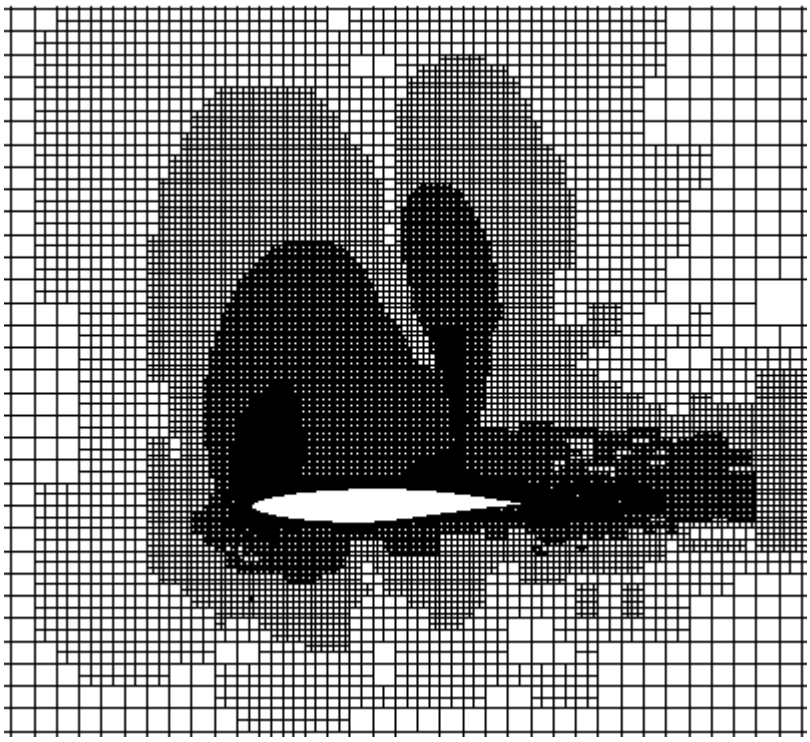
Şekil 5.12 RAE 2822 çevresindeki basınç katsayısı dağılımı



Şekil 5.13 RAE 2822 çevresindeki 5 çevrimli çözüm adaptasyonu kullanılarak elde edilen basınç eş eğrileri



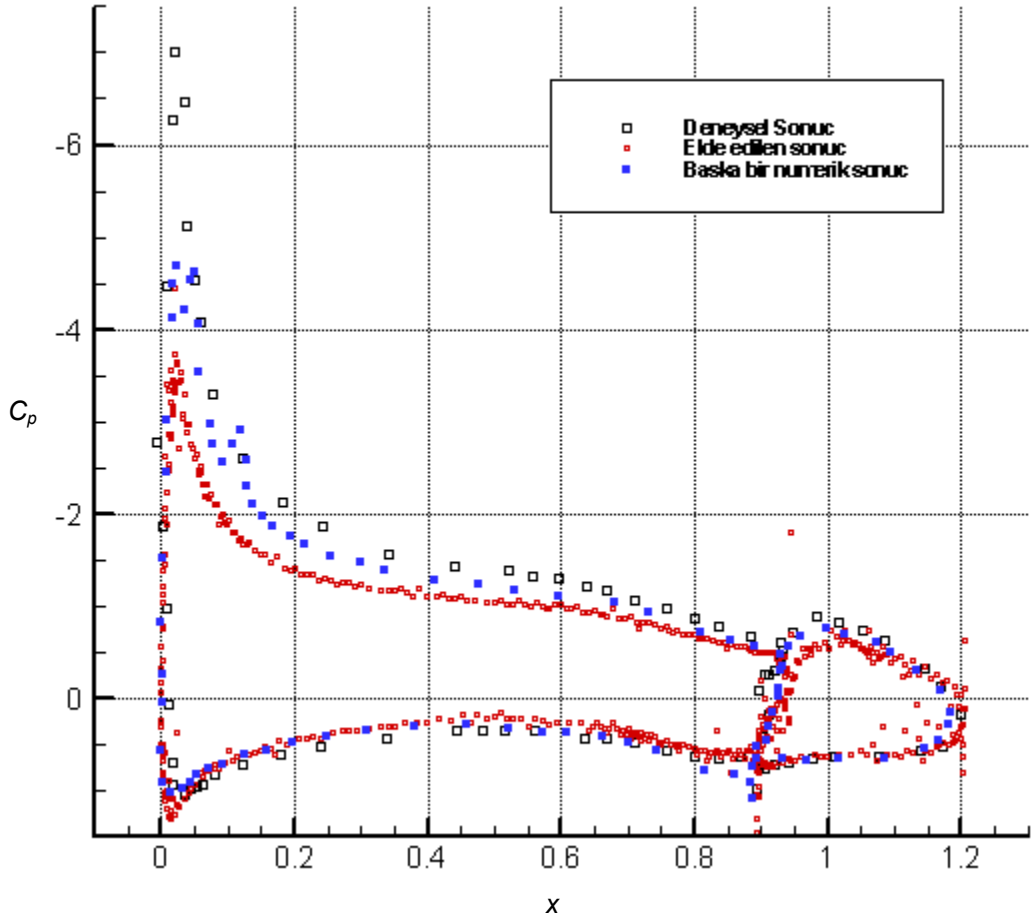
Şekil 5.14 RAE 2822 çevresindeki 5 çevrimli çözüm adaptasyonu ile elde edilen Mach eş eğrileri



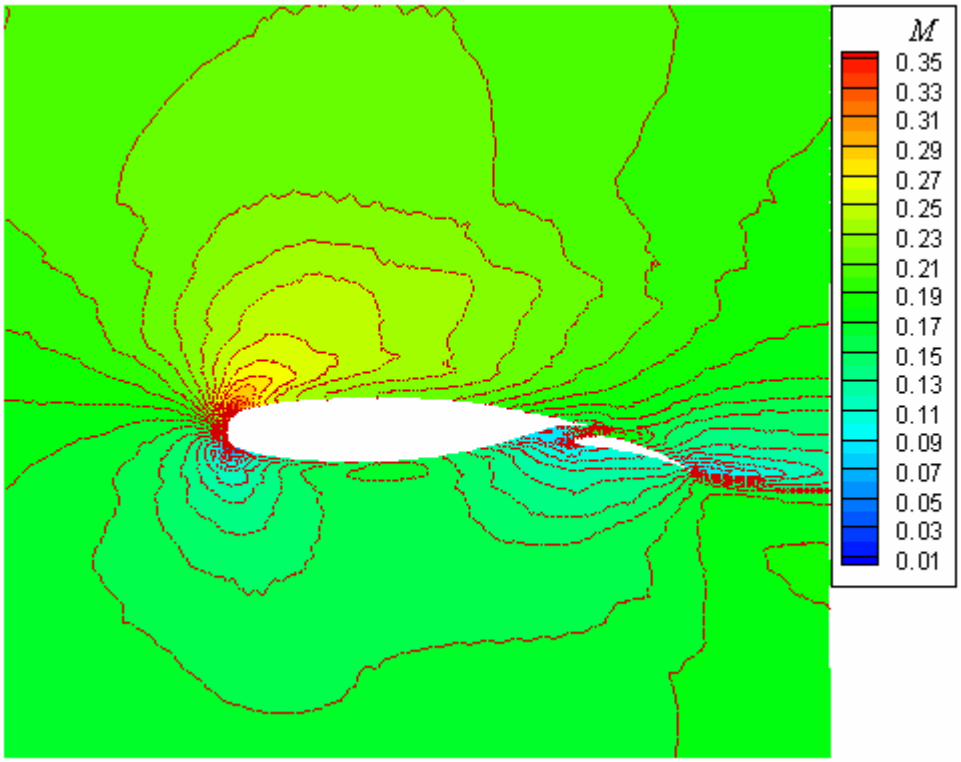
Şekil 5.15 RAE2822 çevresindeki 5 çevrimli çözüm adaptasyonlu ağ

5.2.4 İki Elemanlı Kanatçık Profili Çevresinde Ses Altı Akış

İki elemanlı, NLR7301 ve flap, kanatçık profili etrafında gerçekleştirilen testlerde uzak alandaki Mach sayısı 0.185, hücum açısı ise 6^0 olarak alınmıştır. Bu testin amacı, çok elemanlı, karmaşık geometriler çevresindeki akışı incelemekir. Bu testler için altı seviyeli çoklu ağ kullanılmıştır. Elde edilen kaldırma katsayısı 1.49 ve sürükleme katsayısı 0.1481 olarak bulnmuştur. Elde edilen çözümler hem deneysel hemde başka bir numerik çözümle (Gönç, 2005) kıyaslanmıştır ve sonuçlar Şekil 5.16'da görülmektedir. Şekil 5.16'dan da kolayca farkedildiği gibi kanatçığın üst yüzeyinde oluşan basınç yükselmesi doğru olarak elde edilememiştir. Daha düşük elde edilmiştir ve bunun sebebi incelenen akış rejiminin geliştirilen koda uygun olmamasıdır. Görüldüğü gibi hem geliştirilen kodun çözümünde hemde diğer numerik çözümde basınç katsayısının ulaştığı tepe noktası doğru olarak hesaplanamamıştır. Son olarak, elde edilen Mach eş eğrileri Şekil 5.17'de gösterilmektedir.



Şekil 5.16 İki elemanlı kanatçık profili çevresindeki basınç katsayısı dağılımı



Şekil 5.17 İki elemanlı kanatçık profili çevresindeki Mach eş eğrileri

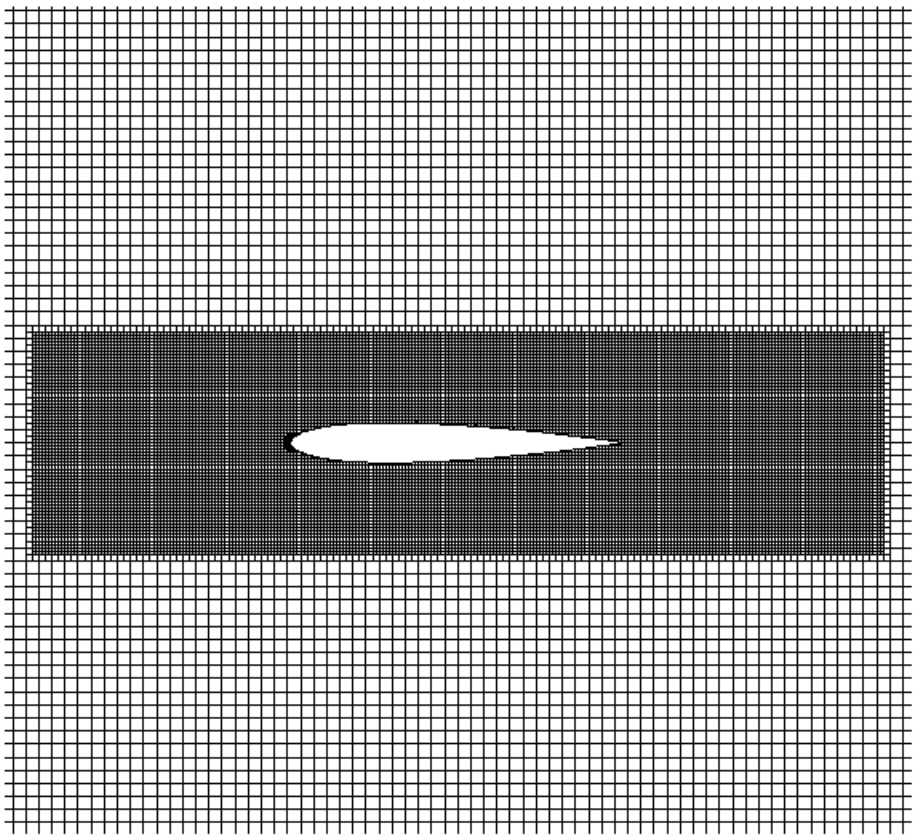
5.3 ÇOKLU AĞ YÖNTEMİNİN TEST EDİLMESİ

Bu bölümde yapılan tüm testler NACA0012 profili etrafında uzak alandaki Mach sayısı 0,85 ve hücum açısı 1^0 olan durum için yapılmıştır. Çözüm adaptasyonlu ve çözüm adaptasyonsuz durumlar içiin kullanılan hesaplama ağları sırasıyla Şekil 5.18 ve 5.19'da verilmiştir.

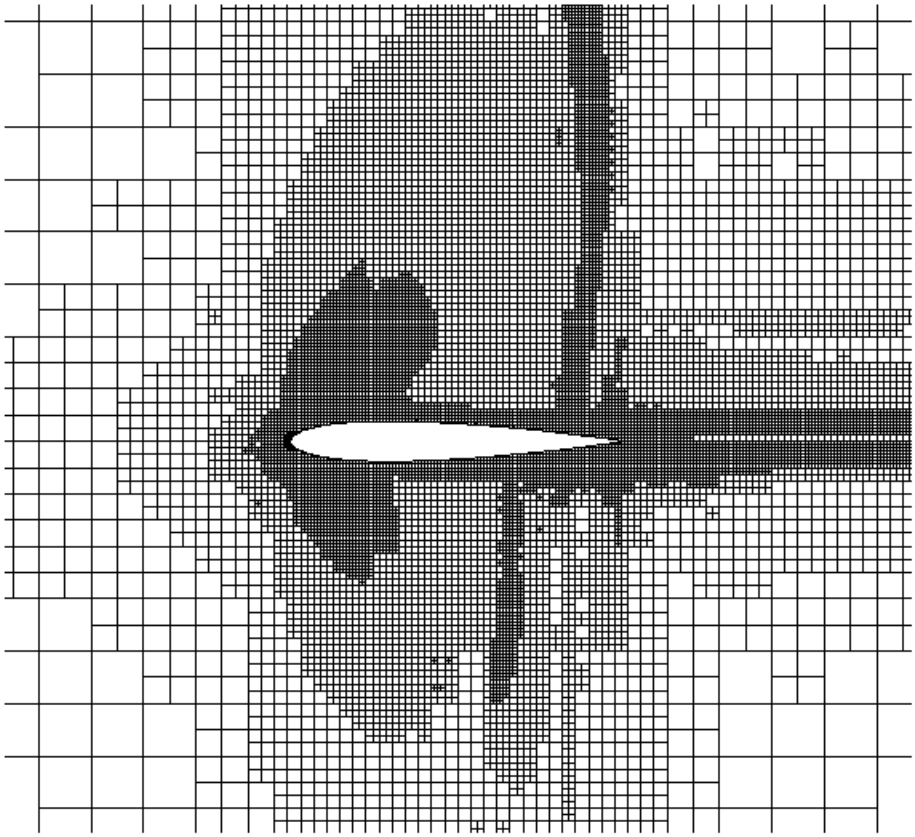
Öncelikle kullanılacak çoklu ağ döngüsüne karar verilebilmesi için V-döngüsü ve testere dişi döngüsü kullanılarak elde edilen çözümlerin ne kadar çevrimde, yinelemede ve zamanda yakınsadığı karşılaştırılmıştır. Şekil 5.20 ve 5.21, 6 seviyeli V döngüsü, 6 seviyeli testere dişi döngüsü ve çoklu ağ yöntemi kullanılmayan durumlar için sırasıyla yineleme sayısını, zamanını ve çevrim sayısını göstermektedir. Şekil 5.20, çözüm adaptasyonu kullanılmadan alınan çözümleri ve yakınsama hızını, Şekil 5.21 ise çözüm adaptasyonu kullanıldığında alınan çözümleri ve yakınsama hızını göstermektedir. Bu çalışmalar için yakınsama tarihçesi sırasıyla Tablo 5.4 ve 5.5'te verilmiştir. Öncelikle belirtilmelidir ki, çözüm adaptasyonu kullanılmayan durumlar için çoklu ağ yönteminin yakınsamaya sağladığı katkılar çok daha fazladır. Şekil 5.20'de gösterilen değerler ele alındığında, çoklu ağ yöntemi kullanıldığında yakınsama zamanı çoklu ağ yöntemi kullanılmadan elde edilen yakınsama zamanının yaklaşık % 7'si kadardır. Şekil 5.21'de ise bu oran %10'a yükselmektedir. Doğal olarak, farklı ağlar ve durumlar için yakınsama hızı oranları değişiklik göstermektedir. Ancak, çoklu ağ yönteminin yakınsama hızına sağladığı

etkinin hiçbir durumda göz ardı edilmesi mümkün değildir. Şekil 5.20 ve 5.21'de görüldüğü gibi, V döngüsünün daha az çevrim sayısı ile yakınsamasına karşılık, yineleme sayısı testere dişi döngüsüne göre çok daha fazladır. Bu durum da, V döngüsünde hesaplama yükünün testere dişi döngüsüne göre daha fazla olduğunu açıkça göstermektedir. Hesaplama zamanına bakıldığında ise V döngüsü Şekil 5.20'de daha avantajlı görülmesine karşılık, Şekil 5.21'de durum bunun tam tersidir. Her durum için testere dişi döngüsü V döngüsüne göre daha hızlı yakınsamasa da, hafıza kullanımı V döngüsünde daha az olduğu için yapılan çoklu ağ seviyesi ve yineleme sayısı belirleme testlerinde testere dişi döngüsü kullanılmıştır.

Testere dişi ve V döngüsünde kullanılacak uzatma operatörlerine karar verebilmek için enjekte ve gradyan operatörleri ile yapılan çözümler ve yakınsama hızları Şekil 5.22 ve 5.23'de görülmektedir. Bu çalışmalar için yakınsama tarihçesi sırasıyla Tablo 5.6 ve 5.7'de verilmiştir. Testere dişi döngüsünde enjekte operatörü kullanıldığında hem çözüm adaptasyonu kullanılmayan hem de kullanılan durumlarda gradyan operatörüne göre az da olsa daha iyi yakınsama hızı elde edilmiştir. Bu yüzden bundan sonraki yapılacak çoklu ağ seviyesi ve yineleme sayısı belirleme testlerinde enjekte operatörü ile testere dişi döngüsü kullanılacaktır.



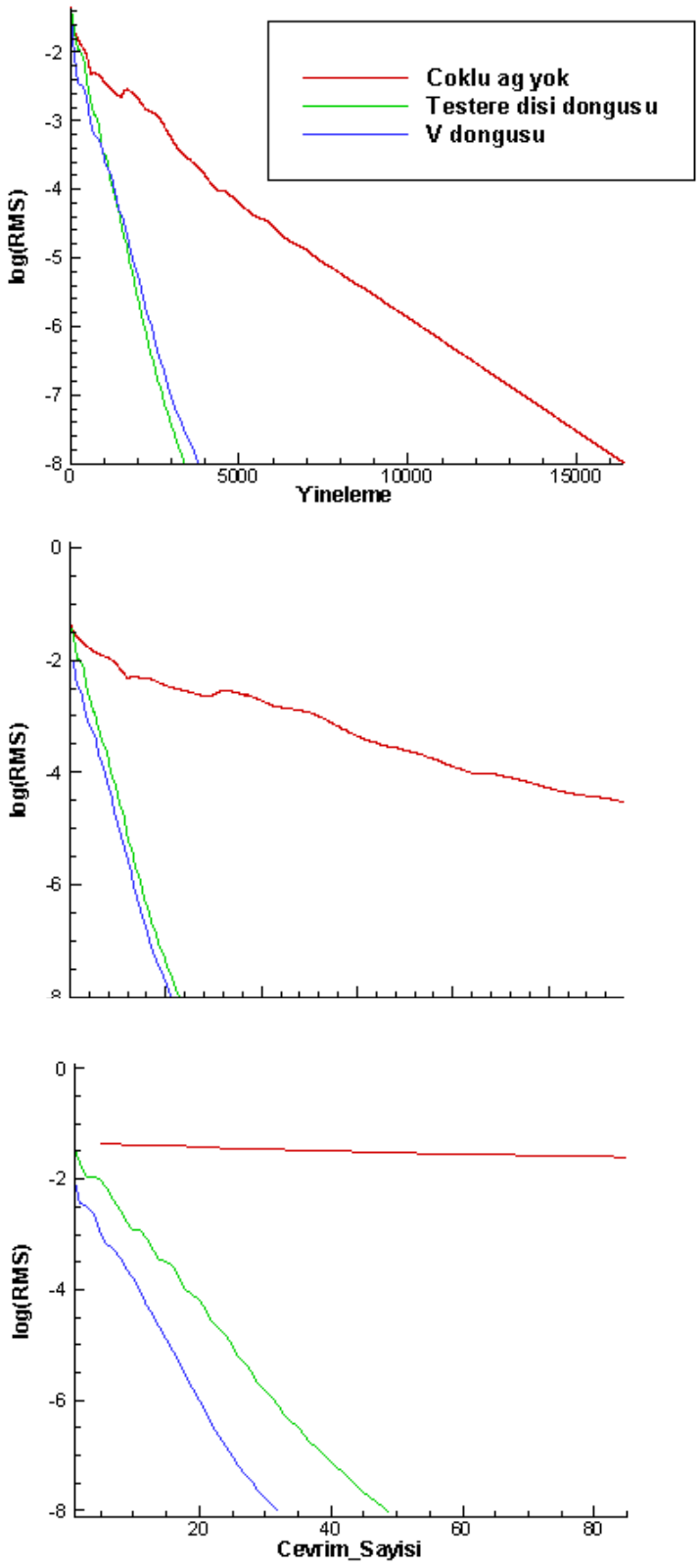
Şekil 5.18 Çözüm adaptasyonsuz durum için kullanılan ağ



Şekil 5.19 Çözüm adaptasyonlu durum için kullanılan ağ

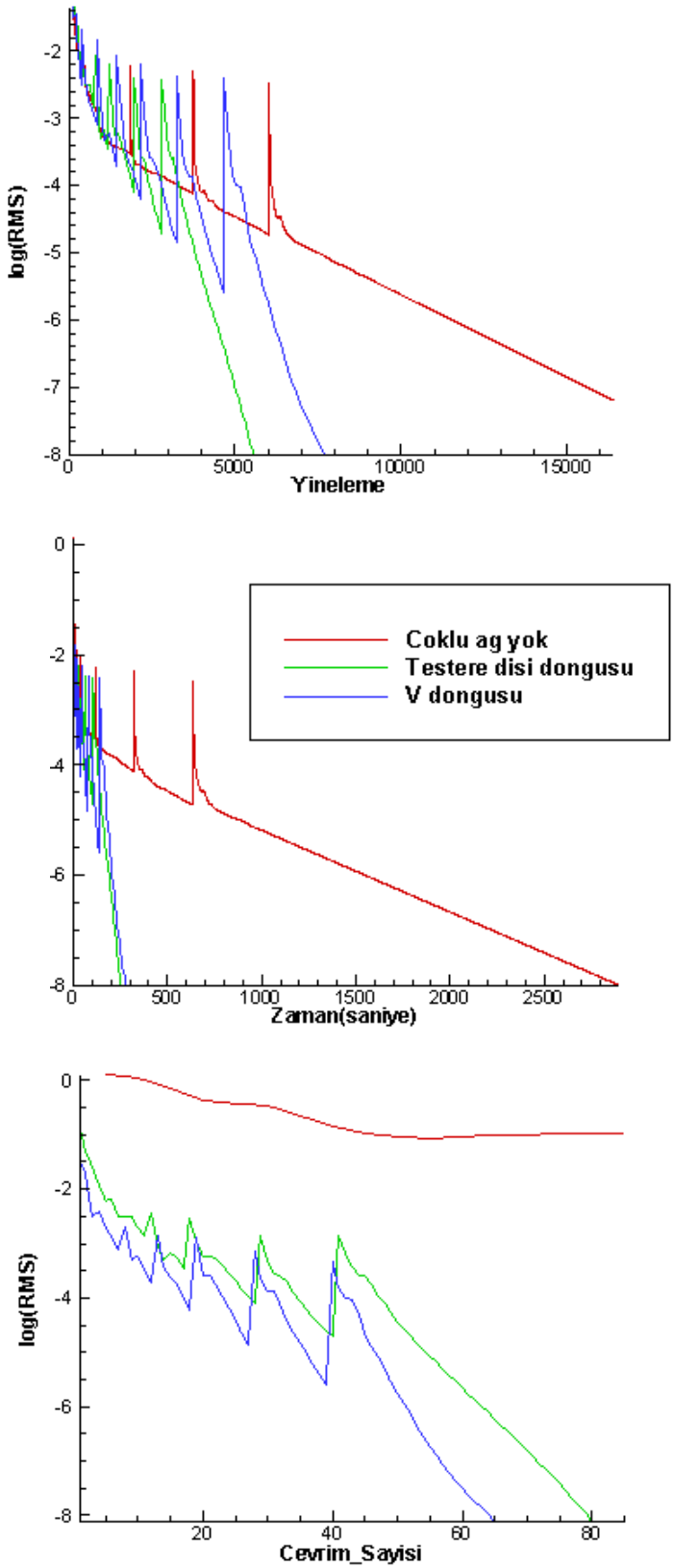
Tablo 5.4 Uygun döngüyü belirlemek için yakınsama tarihçesi
(Çözüm adaptasyonu kullanılmayan durum)

	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
Çoklu ağ yok	16.405	16.405	7.907
Çoklu ağ- Testere dişi döngüsü	49	3.430	579
Çoklu ağ- V döngüsü	32	3.840	535



Şekil 5.20 Uygun döngüyü belirlemek için yakınsama tarihçesi

(Çözüm aaptasyonu kullanılmayan durum)



Şekil 5.21 Uygun döngüyü belirlemek için yakınsama tarihçesi
(Çözüm adaptasyonu kullanılan durum)

Tablo 5.5 Uygun döngüyü belirlemek için yakınsama tarihçesi
(Çözüm adaptasyonu kullanılan durum)

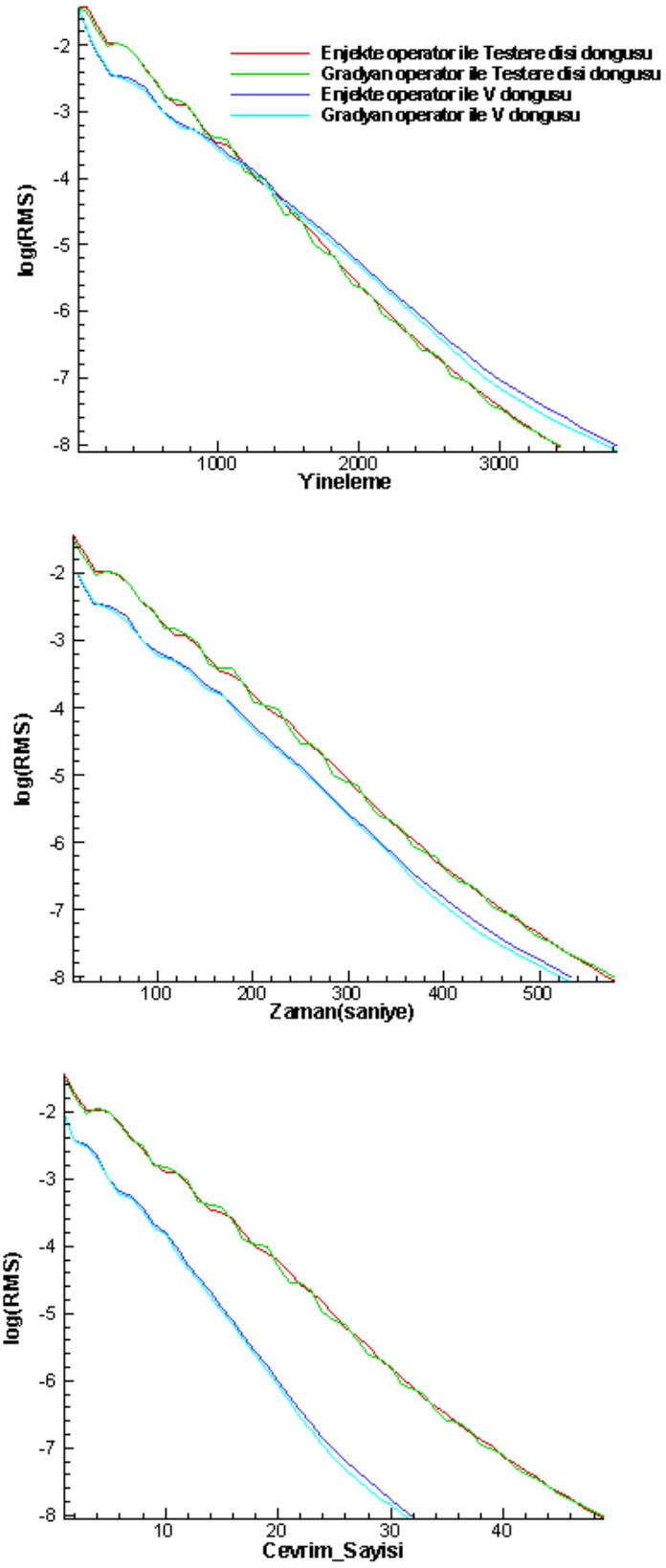
	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
Çoklu ağ yok	19.660	19.660	2.894
Çoklu ağ- Testere dişi döngüsü	80	5.600	253
Çoklu ağ- V döngüsü	65	7.800	284

Tablo 5.6 Uygun uzatma operatörünü belirlemek için yakınsama tarihçesi
(Çözüm adaptasyonsuz)

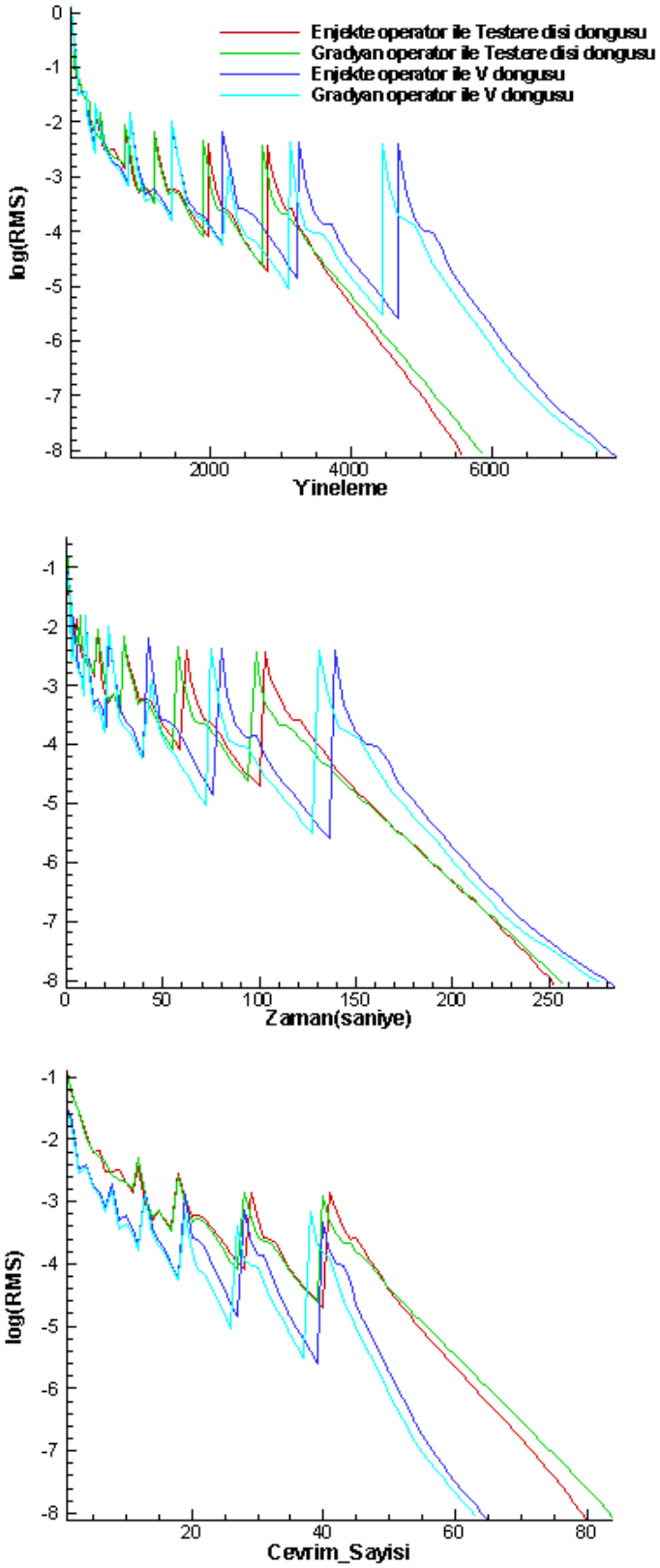
	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
Enjekte operatör ile testere dişi döngüsü	49	3.430	579
Gradyan operatör ile testere dişi döngüsü	49	3.430	582
Enjekte operatör ile V döngüsü	32	3.840	535
Gradyan operatör ile V döngüsü	32	3.840	537

Tablo 5.7 Uygun uzatma operatörünü belirlemek için yakınsama tarihçesi
(Çözüm adaptasyonlu)

	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
Enjekte operatör ile testere dişi döngüsü	80	5.600	253
Gradyan operatör ile testere dişi döngüsü	84	5.880	257
Enjekte operatör ile V döngüsü	65	7.800	284
Gradyan operatör ile V döngüsü	63	7.560	276



Şekil 5.22 Uygun uzatma operatörünü belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)



Şekil 5.23 Uygun uzatma operatörünü belirlemek için yakınsama tarihçesi (Çözüm adaptasyonlu)

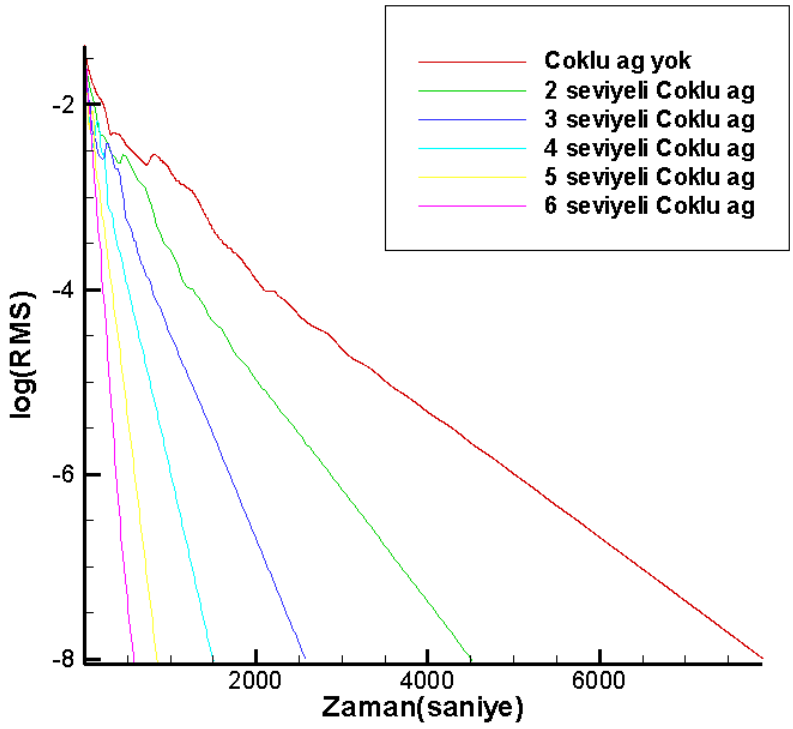
Çoklu ağ yönteminde yakınsama hızını etkileyen diğer bir önemli faktör ise çoklu ağ seviyesidir. Başlangıçta yoğun olan bir ağ için herhangi bir çoklu ağ seviyesi kullanılarak yakınsama hızında iyileştirme elde edilebilir. Fakat başlangıçta az yoğun ağlar için 4 veya 5 seviyeli çoklu ağlardan daha fazla seviye kullanılmaması tavsiye edilir. Ele alınan çözüm adaptasyonsuz ve adaptasyonlu başlangıçta yoğun ağlar kullanılan durumlar için yapılan testlerin yakınsama hızlarını gösteren grafikler Şekil 5.24, 5.25, 5.26 ve 5.27'de verilmiştir. Bu çalışmalar için yakınsama tarihçesi sırasıyla Tablo 5.8 ve 5.9'da verilmiştir. Ele alınan ağlar başlangıçta yoğun olduğu için 6 veya 7 seviyeli çoklu ağlar en iyi yakınsama hızını vermektedir. Çünkü, düşük frekanslı hataları yoğun ağlar kullanılan durumlarda yok edebilmek için normal ağlara göre daha fazla seyreltme işlemi uygulanmaktadır.

Tablo 5.8 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)

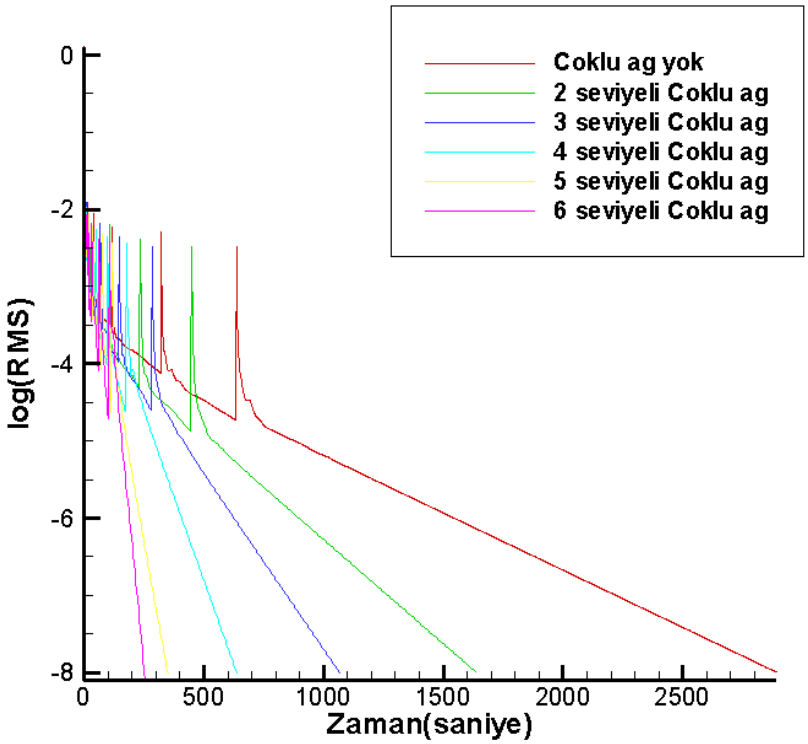
	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
5 seviyeli çoklu ağ	72	4.320	849
6 seviyeli çoklu ağ	49	3.430	579
7 seviyeli çoklu ağ	49	3.920	580
8 seviyeli çoklu ağ	49	4.410	581

Tablo 5.9 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonlu)

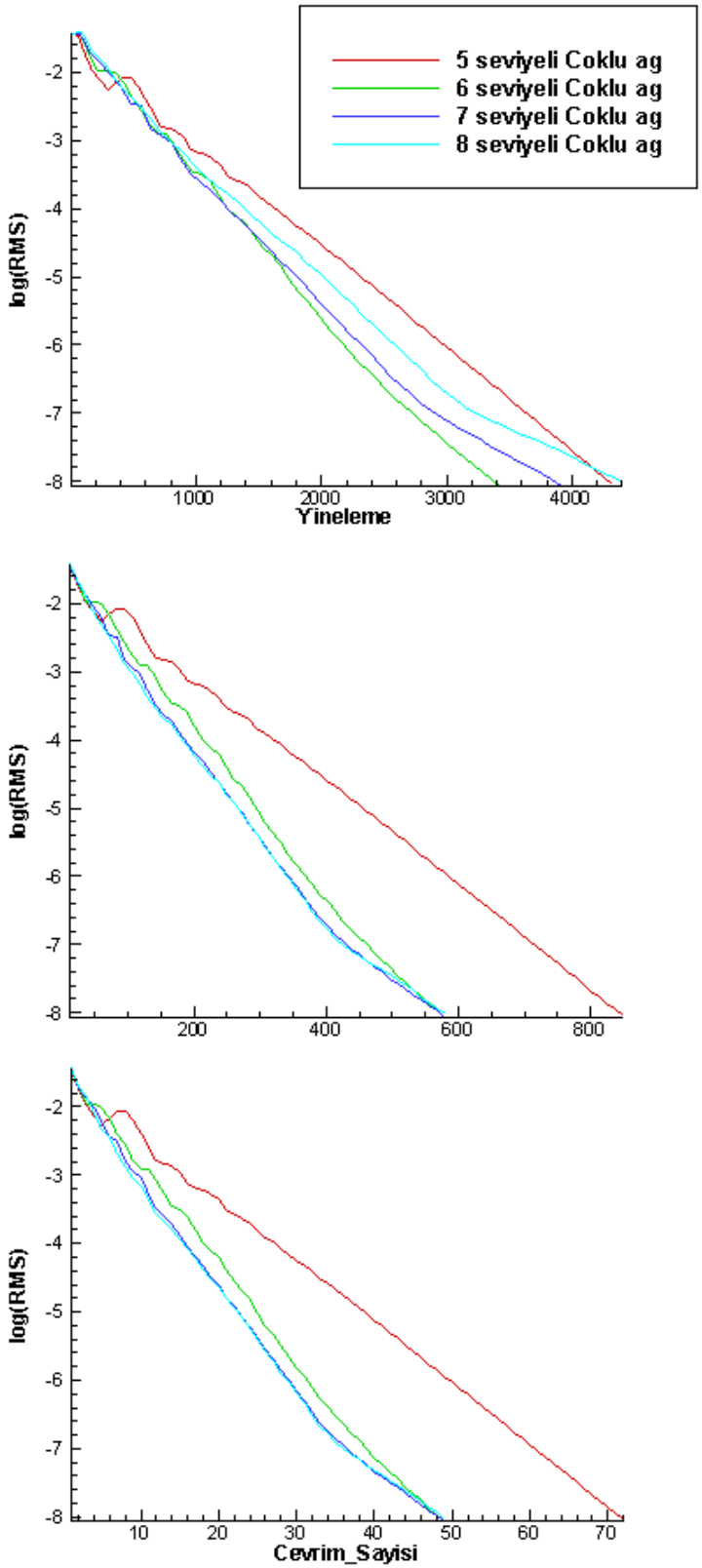
	Çevrim sayısı	**Yineleme sayısı**	**Hesaplama zamanı (s)**
5 seviyeli çoklu ağ	112	6.720	350
6 seviyeli çoklu ağ	80	5.600	253
7 seviyeli çoklu ağ	72	5.760	230
8 seviyeli çoklu ağ	74	6.660	250



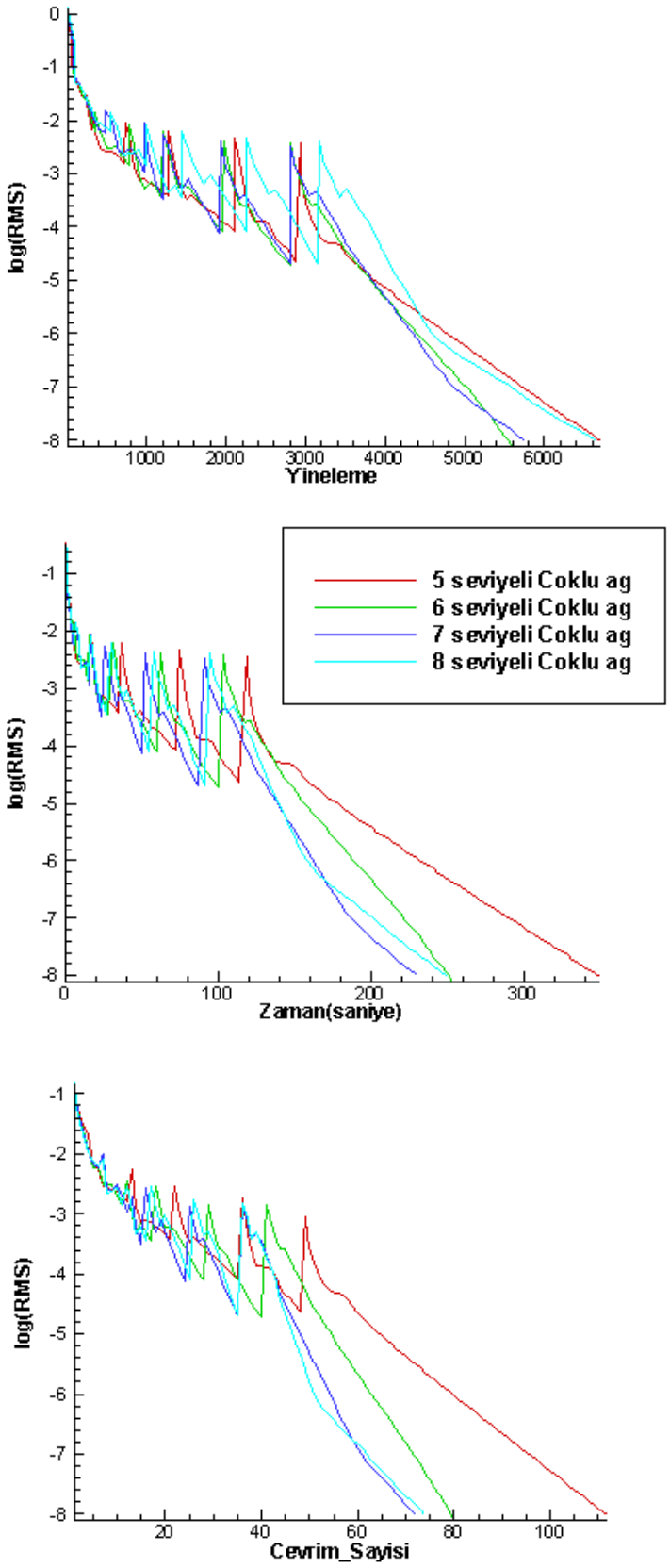
Şekil 5.24 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)



Şekil 5.25 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonlu)



Şekil 5.26 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)



Şekil 5.27 Uygun çoklu ağ seviyesini belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)

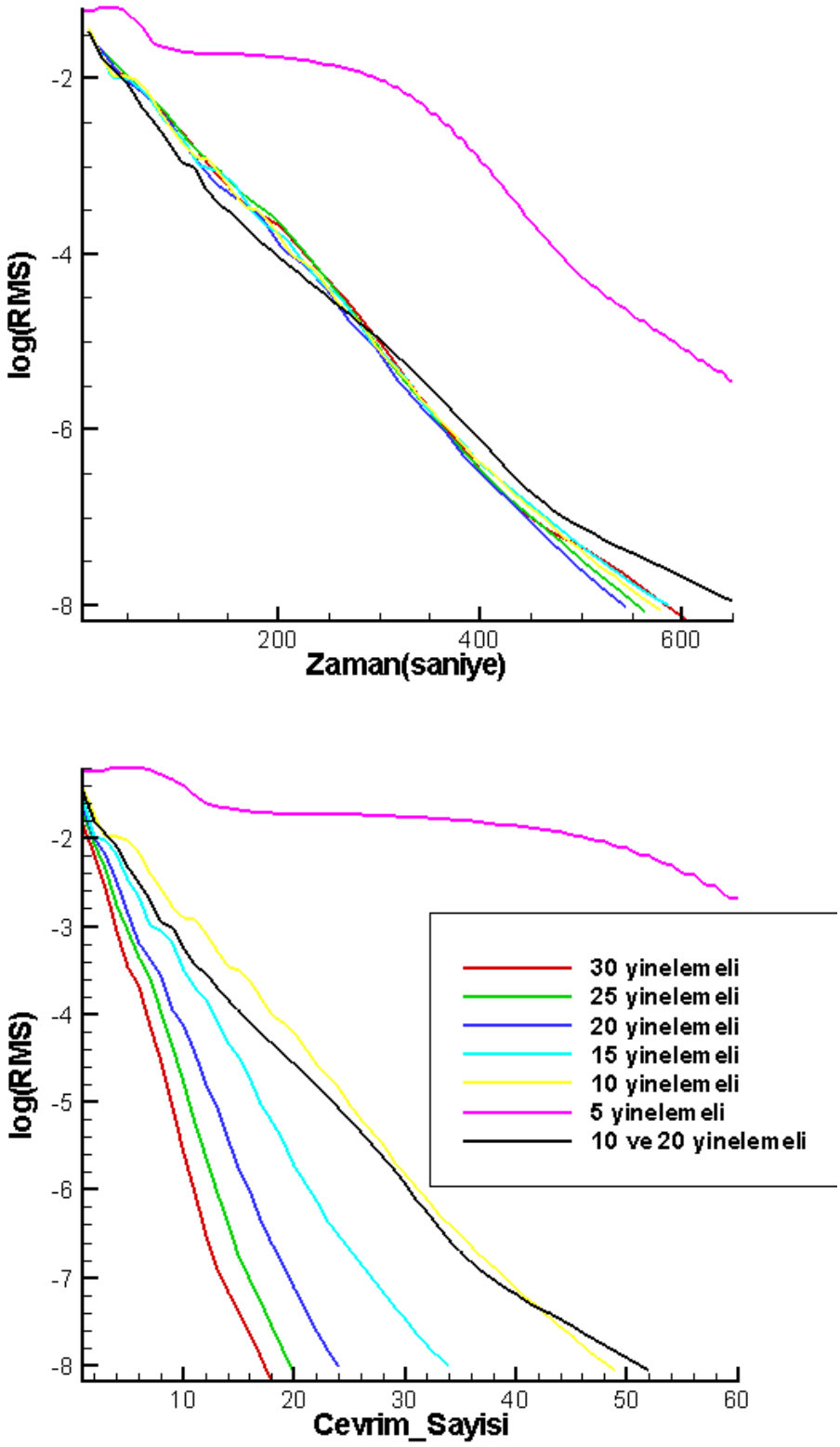
Son olarak, en iyi yakınsama hızını elde edebilmek için her bir çoklu ağ seviyesinde yapılacak yineleme sayısı belirlenmiştir. Bu aşamada, çok çeşitli varyasyonların yapılması mümkündür. Burada sadece en basit hali ile denemeler yapılarak uygun bir yineleme sayısı bulunmaya çalışılmıştır. 6 seviyeli testere dişi döngüsü için kullanılan yineleme sayıları ve yakınsama hızları Şekil 5.28 ve 5.29'da gösterilmiştir. Bu çalışmalar için yakınsama tarihçesi sırasıyla Tablo 5.10 ve 5.11'de verilmiştir. Görüldüğü gibi yineleme sayılarının 10'dan az, 20'den çok olması iyi yakınsama hızları vermemektedir.

Tablo 5.10 Her çoklu ağ seviyesinde yineleme sayısını belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)

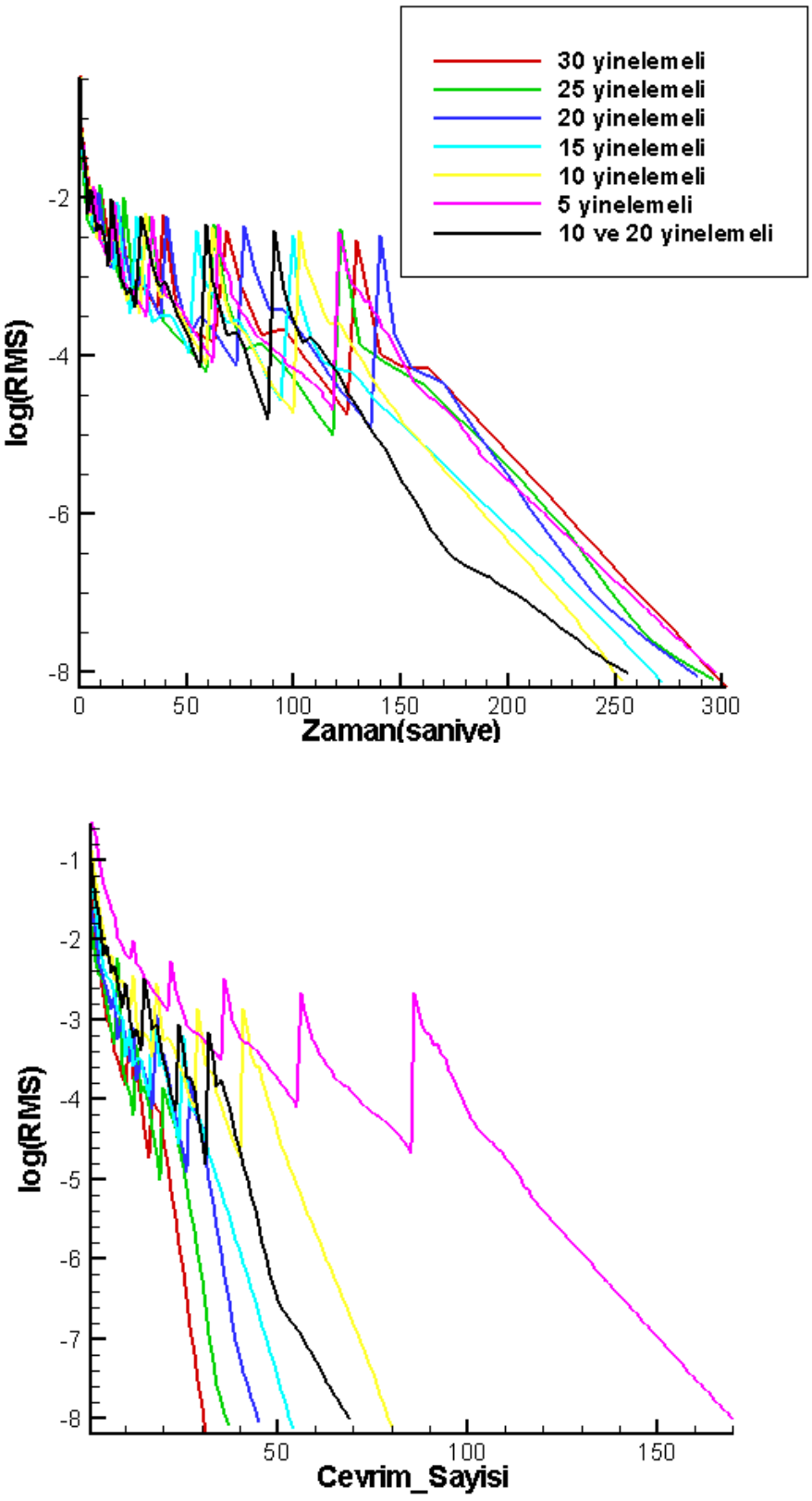
	Çevrim sayısı	**Çevrim sayısı**	**Hesaplama zamanı (s)**
Her seviyede 30 yinelemeli çoklu ağ	18	3.780	606
Her seviyede 25 yinelemeli çoklu ağ	20	3.500	564
Her seviyede 20 yinelemeli çoklu ağ	24	3.360	545
Her seviyede 15 yinelemeli çoklu ağ	34	3.570	587
Her seviyede 10 yinelemeli çoklu ağ	49	3.430	579
Her seviyede 5 yinelemeli çoklu ağ	134	4.690	858
Her seviyede sırasıyla 10 ve 20 yinelemeli çoklu ağ	52	5.200	667

Tablo 5.11 Her çoklu ağ seviyesinde yineleme sayısını belirlemek için yakınsama tarihçesi (Çözüm adaptasyonlu)

	Çevrim sayısı	**Çevrim sayısı**	**Hesaplama zamanı (s)**
Her seviyede 30 yinelemeli çoklu ağ	31	6.510	302
Her seviyede 25 yinelemeli çoklu ağ	37	6.475	296
Her seviyede 20 yinelemeli çoklu ağ	45	6.300	289
Her seviyede 15 yinelemeli çoklu ağ	54	5.670	272
Her seviyede 10 yinelemeli çoklu ağ	80	5.600	253
Her seviyede 5 yinelemeli çoklu ağ	170	5.950	297
Her seviyede sırasıyla 10 ve 20 yinelemeli çoklu ağ	69	6.900	256



Şekil 5.28 Her çoklu ağ seviyesinde yineleme sayısını belirlemek için yakınsama tarihçesi (Çözüm adaptasyonsuz)

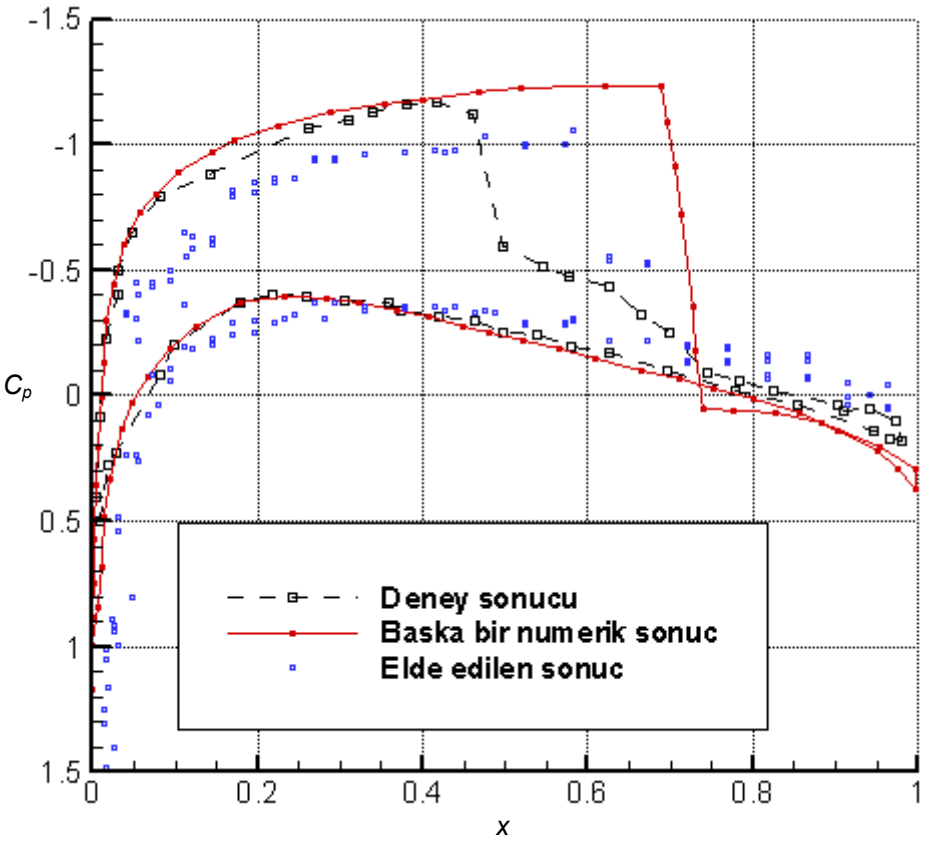


Şekil 5.29 Her çoklu ağ seviyesinde yineleme sayısını belirlemek için yakınsama tarihçesi (Çözüm adaptasyonlu)

5.3 ÜÇ BOYUTLU KARTEZYEN AĞ ÜRETİCİSİ İLE EULER ÇÖZÜCÜSÜNÜN TEST EDİLMESİ

5.3.1 Kanat Çevresinde Ses Civarı Akış

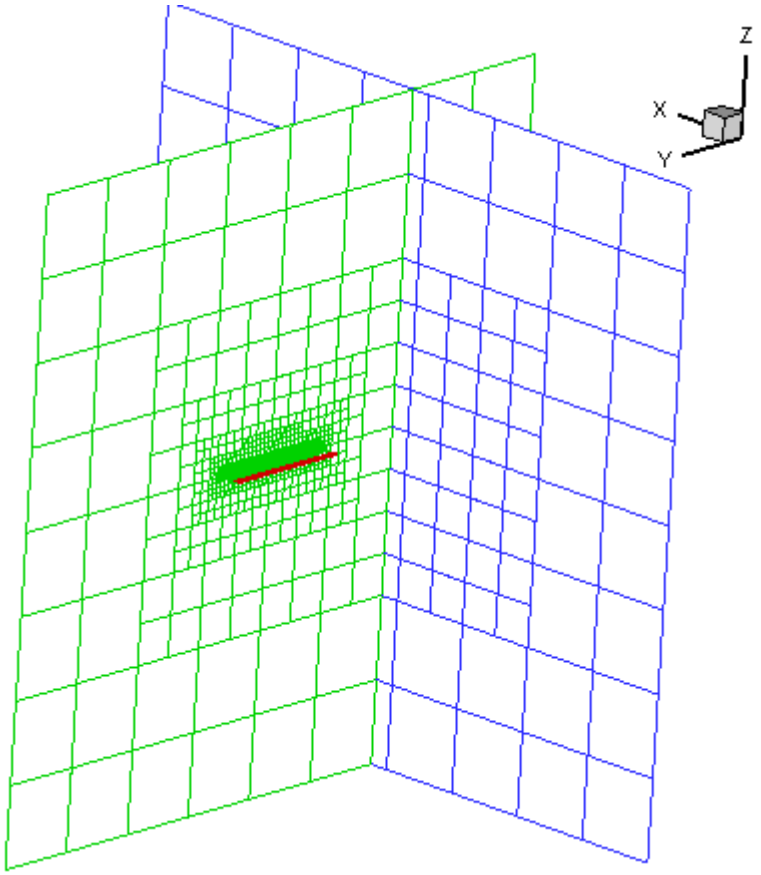
Geliştirilen üç boyutlu Euler çözücüsünün doğrulanmasına NACA0012 kanatçık profili kullanılarak elde edilen kanat ile başlanılmıştır. Ses civarı akış için kanat etrafında gerçekleştirilen testlerde uzak alandaki Mach sayısı 0.799, hücum açısı ise 2.26^0 olarak alınmıştır. Kanadın en-boy oranı (aspect ratio) 20 olarak alınmıştır. En-boy oranının bu kadar büyük alınmasının nedeni, kanat uçlarında oluşan vortekslerin sonuça kattığı üç boyutlu etkiyi minimuma indirerek kanadın ortasında elde edilen sonuçları, literatürde bulunan iki boyutlu NACA0012 deneysel sonuçları ile kıyaslamaktır. Fakat en-boy oranının bu kadar büyük alınması kök hücrenin bir kenar uzunluğunu çok büyütmektedir. Bu durumda kritik yerlerde yoğun ağ elde etme işlemini hücre sayısı çok arttığından dolayı zorlaştırmaktadır. Bu yüzden elde edilen sonuçlar deneysel verilere çok yakınsayamamıştır. Bu durum, üç boyutlu ağ yaratıı ve çözücüsünün yanlış çalıştığını değil, seçilen test durumuna uygun doğru ağın yaratılamamış olduğunu göstermektedir. Üç boyutlu kod ile yapılan diğer test sonuçları deneysel verilere yakın elde edilmiştir.



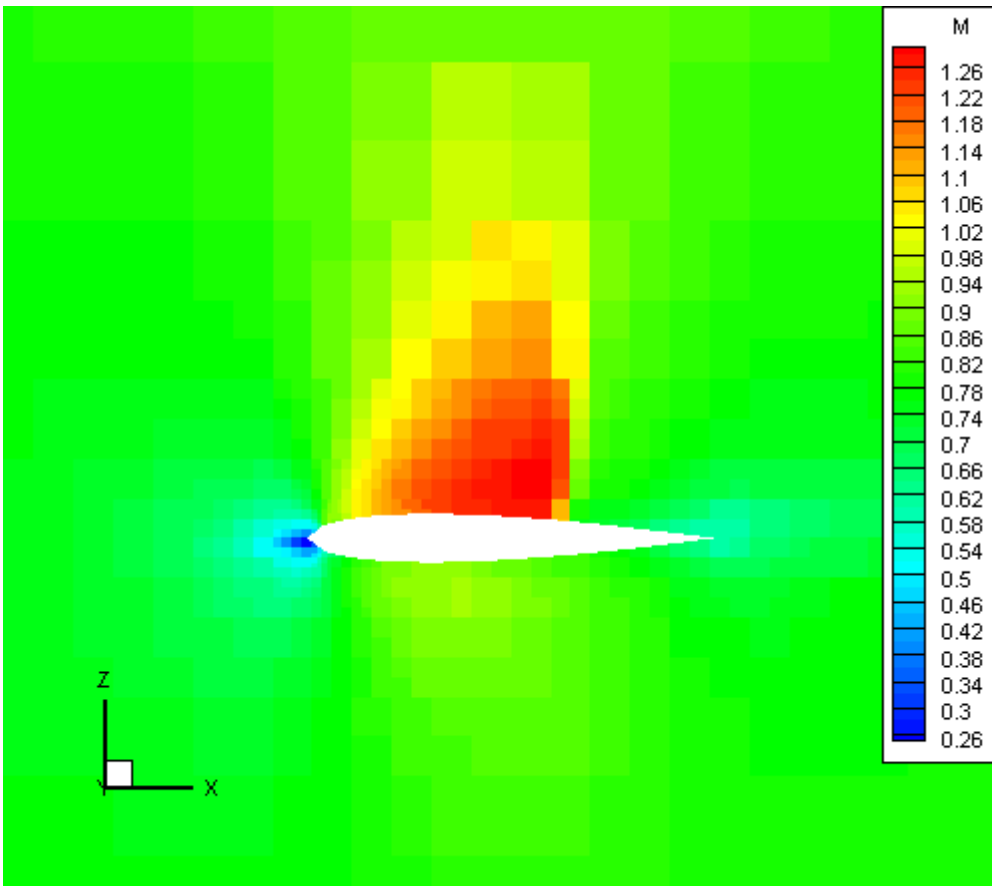
Şekil 5.30 Kanat çevresindeki basınç katsayısı dağılımı

Böyle bir akışta, kanadın üst yüzeyinde şok oluşması beklenmektedir. Bu çalışmalardaki en kritik nokta şokun yeri ve kuvvetidir. Bu test için altı seviyeli çoklu ağ ve 1 çevrimli ağ adaptasyonu kullanılmıştır. Elde edilen sonuçlar, hem literatürdeki deneysel verilerle hemde başka bir sürtünmesiz akış çözücü kullanılarak elde edilmiş basınç katsayılarının sonuçları ile Şekil 5.30'da kıyaslanmıştır.

Çözüm adaptasyonlu ağdan herhangi bir kesit ve Mach eş eğrileri sırasıyla Şekil 5.31 ve 5.32'de görülmektedir. Öncedende bahsedildiği gibi kanadın üstünde oluşan şokun yeri ve kuvveti doğru hesaplanamamıştır. Bu durumun iki sebebi vardır. Birinci sebebi, en-boy oranı çok büyük seçildiğinden ve parallel işlemciler kullanılmadığından tek bilgisayarla fazla çözüm adaptasyonunun uygulanamamış olmasıdır. İkincisi sebep ise, karşılaştırılan deneysel veriler iki boyutlu NACA0012 profili olmaktadır ve üç boyutlu çözümlerle bu deneysel verilerin birebir bağdaşması zaten beklenemez.



Şekil 5.31 Kanat çevresinde oluşturulan bir çevrimli çözüm adaptasyonlu ağdan bir kesit

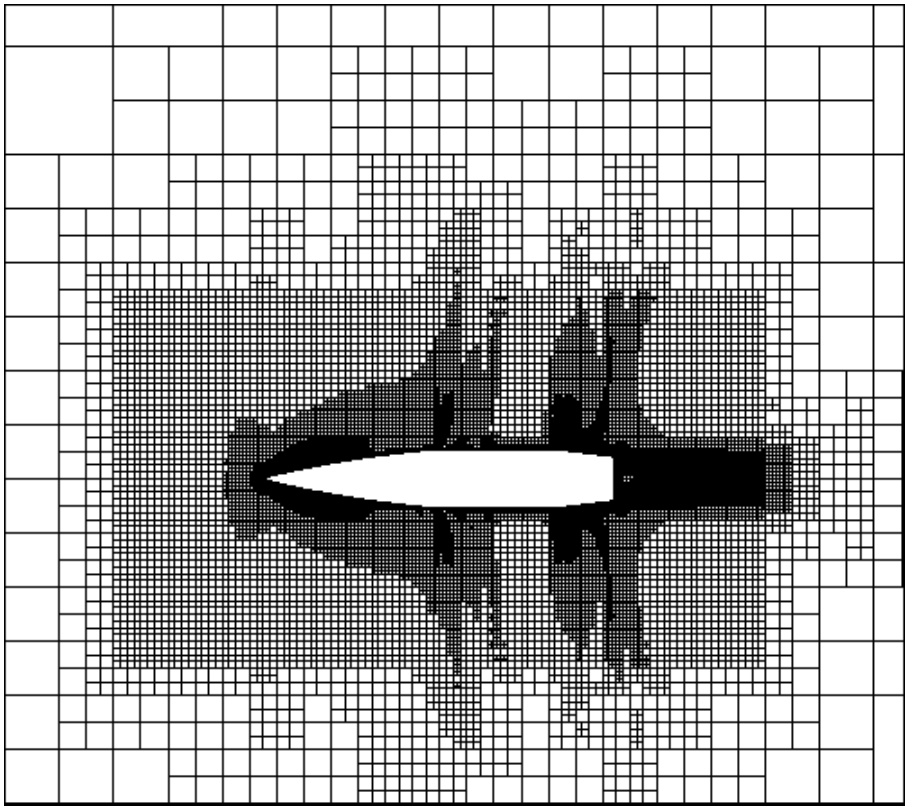


Şekil 5.32 Kanadın tam ortasından alınan kesitteki Mach eş eğrileri (y = 0 iken xz düzlemi)

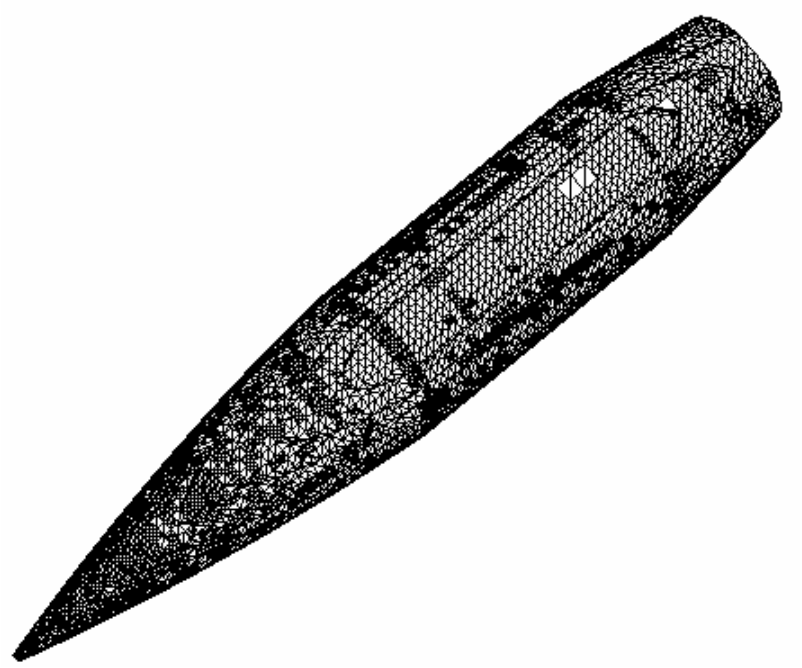
5.3.2 Kurşun Çevresinde Ses Civarı Akış

Ses civarı akış için sekant-ogiv-silindir-konik kuyruklu, kuyruk açısı (boattail angle) 7^0 olan mermi (secant-ogive-cylinder-boat tail projectile (SOCBT)) etrafında gerçekleştirilen testlerde uzak alandaki Mach sayısı 0.95, hücum açısı ise 0^0 olarak alınmıştır. Elde edilen basınç katsayıları deneysel verilerle ve mach konturleri literatürdeki bir numerik çözüm ile kıyaslanmıştır. Bu test için 1 çevrimli ağ adaptasyonu kullanılmıştır. Elde edilen çözüm adaptasyonlu ağın y = 0 iken xz düzlemindeki kesiti ve yüzeyi oluşturan üçgen yüzey ağı sırasıyla Şekil 5.33 ve 5.34'te görülmektedir.

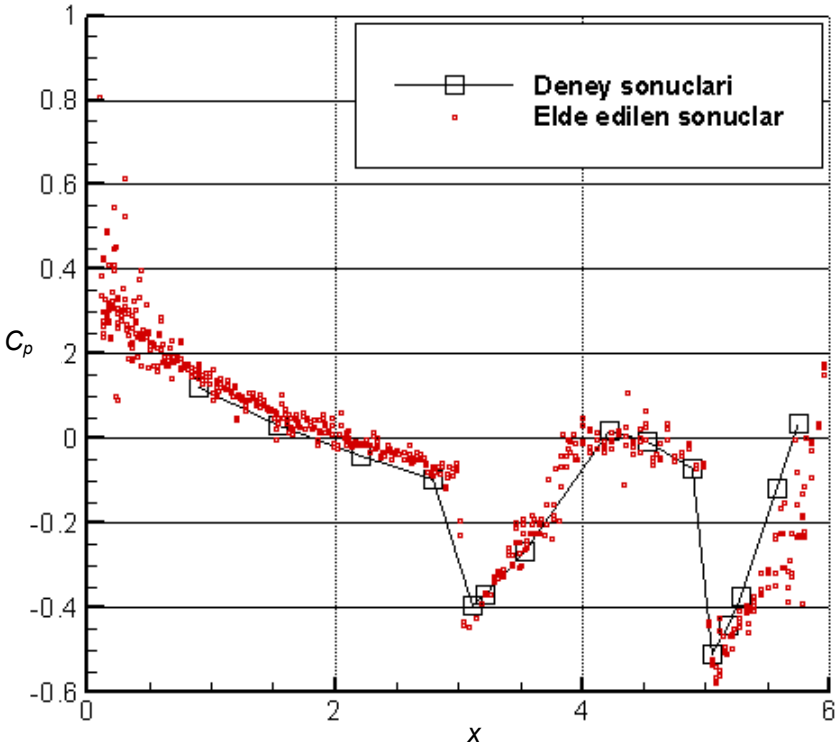
Basınç katsayısının dağılımı Şekil 5.35'te gösterilmektedir. Çözüm adaptasyonlu Mach eş eğrileri konturu Şekil 5.36'da ve literatürdeki bir çalışmadan alınan Mach eş eğrileri Şekil 5.37'de görülmektedir. Şekillerden de görüldüğü gibi mermi üzerinde iki farklı yerde şok vardır. Bunlardan biri merminin ortasında, diğeri konik kuyruğun başladığı yerdedir. Şokların hem yerleri hemde kuvvetleri doğru tespit edilmiştir. Ayrıca, Mach eş eğrileri arasındaki benzerlikte üç boyutlu geliştirilen kodun doğruluğunu kanıtlamaktadır. Son olarak, basınç katsayılarındaki titreşmeyi (oscillation) engellemek için daha fazla çözüm adaptasyonu uygulanabilir.



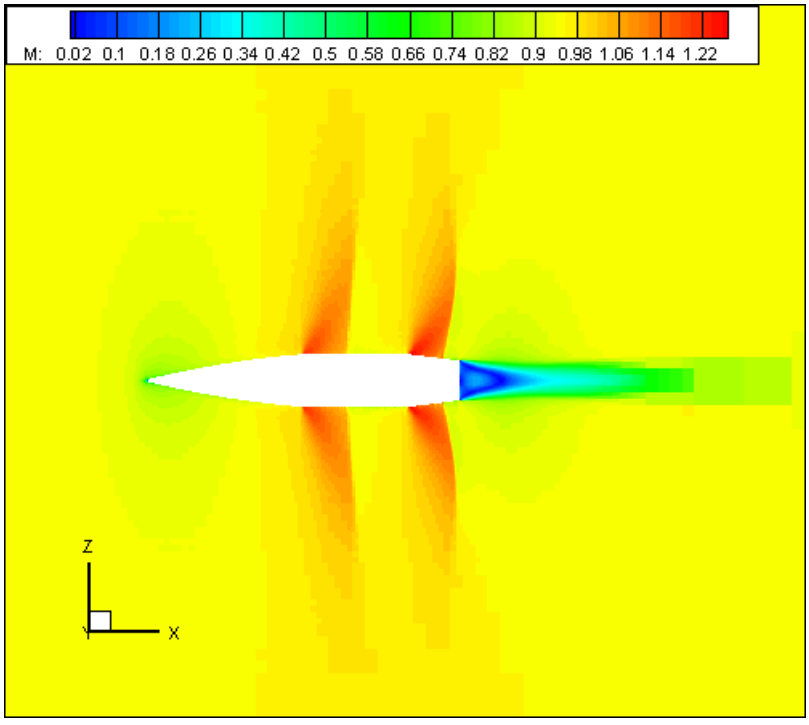
Şekil 5.33 SOCBT çevresinde oluşturulan bir çevrimli çözüm adaptasyonlu ağdan bir kesit



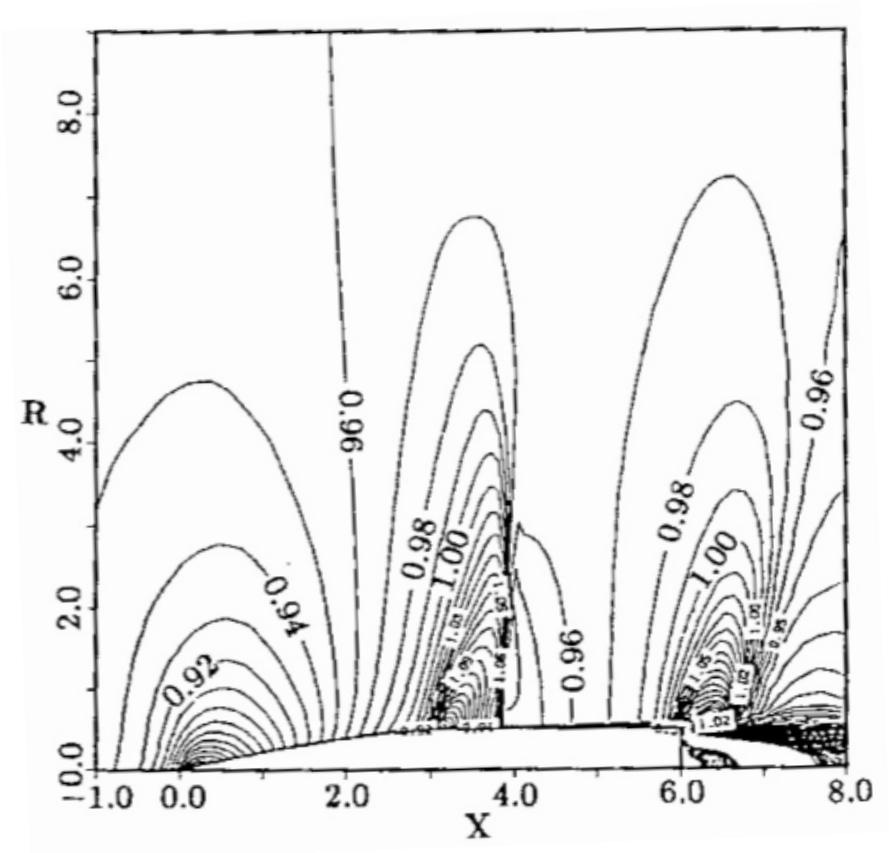
Şekil 5.34 SOCBT çevresinde oluşan yüzey ağı



Şekil 5.35 SOCBT çevresindeki basınç katsayısı dağılımı



Şekil 5.36 SOCBT çevresindeki elde edilen mach konturleri



Şekil 5.37 Literatürdeki bir sayısal çözümünden alınan SOCBT çevresindeki Mach eş eğrileri

BÖLÜM 6

DEĞERLENDİRME

Bu çalışmada iki boyutllu Euler çözücüsünü doğrulmak için dört değişik test çalışması yapılmıştır. İlk test durumunda, çözücünün şok yakalama yeteneği NACA 0012 kanadı üzerindeki ses civarı akışta incelenmiştir. Çözüm adaptasyonlu test durumunun performansı tatminkar bulunmuştur. Şokun yeri ve basınç katsayısının en yüksek değeri iyi bir şekilde belirlenmiştir. Başka bir deyişle, çözüm adamtasyonunun önemi bu test durumu için ispatlanmıştır.

Birinci ve ikinci dereceden akı hesaplama şemaları ikinci test durumu olan NACA 0012 kanadı etrafındaki 7^0 hücum açılı ses üstü akışa uygulanmıştır. İkinci derecedn şemalar, yay şeklindeki şok dalgasından önce karasızlıklar oluşmuştur. Sınırlayıcılar gradyanları düzenleyerek bu kararsızlıkları azaltmasına karşılık, tamamen ortadan kaldıramamıştır.

Diğer bir test durumu RAE 2822 kanadı için optmum çözüm adaptasyon çevrim sayısını belirlemek için gerçekleştirilmiştir. Çevrim sayısının çok fazla artttırılması yakınsama hızının azalmasına neden olmuştur. Bu test durumu için çevrim sayısı beş veya altı olduğu zaman en doğru çözümler ve tatminkar bir yakınsama hızı elde edilmiştir.

Dördüncü test durumunda, Kartezyen hesaplama ağlarının otomatik ağ yaratma yetenekleri gösterilmiştir. Bu test durumları için elde edilen sonuçlarla literatürde mevcut sonuçlar arasında belirgin farklılıklar bulunmaktadır. Buna ek olarak, kanat yüzeyi üzerinde elde edilen çözümlerin düzgün olmadığıı gözlenmiştir. Özellikle üç parçalı kanadın slat ve hücum kenarında dalgalanmalar görülmüştür. Bu durum özellikle basınç katsayılarının verildiği şekillerde rahatlıkla görülmektedir. Bu dalgalanmaların nedenlerinden bir tanesi gövde üzerinde bulunan hücrelerin boyutlarında önemli farklılıklar bulunmasıdır. Örnek olarak, boyutları dış hücrelere göre 10^{-4} oranında daha küçük kesik küçük hücreler bulmak mümkündür.

Çoklu ağ yönteminin yakınsama hızına önemli bir etkisi olduğu görülmektedir. Bütün test durumları yakınsama hızının arttığı görülmektedir.

KAYNAKLAR

AFTOSMIS, M. J., 3D Applications of a Cartesian Grid Euler Method, AIAA Paper 95-0853, (1995).

AFTOSMIS, M. J., Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry, AIAA Paper 97-0196, (1997).

AFTOSMIS, M. J., MELTON, J. E. and BERGER, M. J., Adaptation and Surface Modeling for Cartesian Mesh Methods, AIAA Paper 95-1725-CP, (1995).

AGARD Subcommittee C., Test Cases for Inviscid Flow Field Methods. AGARD Advisory Report 211, (1986).

BARTH, T. J., Recent Developments in High Order K-Exact Reconstruction on Unstructured Meshes, AIAA Paper 93-0668, 31st Aerospace Sciences Meeting, Reno, Nevada, (1993).

BARTH, T. J., *Simplified Numarical Methods for Gasdynamic Systems on Triangulated Domains* (Ph. D. Thesis). Stanford University, Stanford, (1998).

BARTH, T. J. and JESPERSEN D: C., The design and Application of Upwind Schemes on Unstructured Meshes, AIAA Paper 89-0366, (1989).

BARTH T. J. and FREDERICSON, P. O., Higher Order Solution of the Euler Equations, AIAA Paper 90-0013, (1990).

BERGER, M. J. and LeVEQUE, R., Cartesian Meshes and Adaptive Mesh Refinement for Hyperbolic partial Differential Equations, Proc. 3rd Intl. Conf. Hyp. Problems, Upsala, Sweeden, (1990)

BERGER, M. J. and MELTON, J. E., An Acccuracy Test of a Cartesian Grid Method for Steady Flow in Complex Geometries, Proc. 8th. Intl. Conf Hyp. Problems, Upsala, Sweeden, (1995).

BRANDT, A., Multi-Level Adaptive Solutions to Boundary Value Problems, *Mathematics for Computation*, 31, 333-390, (1977).

BLAZEK, J., *Computational Fluid Dynamics: Principles and Applications*, Elsevier, St. Augustin, (2001).

CLARKE, D. K., SALAS, D. and HASSAN, H. A., Euler Calculations for Multielement Airfoils Using Cartesian Grids, *AIAA Journal*, 24, 353-358, (1986).

COIRIER, W. J., An Adaptively Refined, Cartesian, Cell-Based Scheme for the Euler and Navier Stokes Equations (PhD Thesis), The University of Michigan, Michingan, (1994).

COIRIER, W. J. and POWELL, K. G., An Accuracy Assessment of Cartesian-Mesh Approaches for the Euler Equations, *Journal of Computational Physics*, 117, 121-131, (1993).

De ZEEUW, D. L. and POWELL, K. G., An adaptively Refined Cartesian Mesh Solver for the Euler Equations, AIAA Paper 91-1542, (1991).

De ZEEUW, D. L., A Quad-Tree Based Adaptively-Refined Cartesian-Grid Algorithm for the Solution of The Euler Equations (PhD Thesis), The University of Michigan, Michigan, (1993).

DADONE, A. And GROSSMAN, B., Ghost-Cell Method for Inviscid Two-Dimensional Flows on Cartesian Grids, *AIAA Journal*, 42, 2499-2507, (2004).

EPSTEIN, B., LUNTZ, A. and NACHSON, A., Cartesian Euler Method For Arbitrary Aircraft Configurations, AIAA Journal, 30, 679-687, (1992).

FEDORENKO, R. P., A Relaxation Method for Solving Elliptic Difference Equations, *USSR Computational Math. and Math. Phys.*, Vol. 1, (1962).

FEDORENKO, R. P., The Rate of Convergence of an Iterative Process, *USSR Computational Math. and Math. Phys.*, Vol. 4, (1964).

FORRER, H., Boundary Treatment for a Cartesian Grid Method, Seminar für Angewante Mathmatic, ETH Zürich, ETH Research Report No. 96-04, (1996).

FORRER, H. and JELTSCH, R., A Higher-Order Boundary Treatment for Cartesian-Grid Methods, *Journal of Computational Physics*, 140, 259-277, (1998).

FRENCH, R. D., Solution of the Euler Equations on Cartesian Grids", *Applied Numerical Mathematics*, 49, 367-379, (2004).

GAFFNEY, R. HASSAN, H. And SALAS M., Euler Calculations for Wings Using Cartesian Grids, AIAA Paper 87-0356, (1987).

GÖNÇ O. L., *Computation of External Flow Around Rotating Bodies* (Ph. D. Thesis), Middle East Technical University, Ankara, Turkey, (2005).

GROSMAN, B. And WHITAKER, D., Supersonic Flow Computations using a Rectangular Coordinate Finite Volume Method, AIAA Paper 86-0442, (1986).

HUNT, J., An Adaptive 3D Cartesian Approach for the Parallel Computation of Inviscid Flow about Static and Dynamic Configurations (Ph. D. Thesis), University of Michigan, Michigan, (2004).

LI, K. and WU, Z. N., Nonet-Cartesian Grid Method for Shock Flow Computations, *Journal of Scientific Computing*, 20, 303-329, (2004).

KARMAN, S. L., Jr., Splitflow: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries, AIAA Paper 95-0343, (1995).

Karman ve Splitflow (1995)

KEATS, W. A. and LİEN, F. S., Two-Dimensional Anisotropic Cartesian Mesh Adaptation for the Compressible Euler Equations, *International Journal for Numerical Methods in Fluids*, 46, 1099-1125, (2004).

KHOKLOV, A. M., Fully-Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations, (1998).

LAMBERT, J. D., *Computational Methods in Ordinary Differential Methods*, Wiley, London, (1973).

LEDNICER, D., TIDD, D. and BIRCH, N., Analysis of a Closed Coupled Nacelle Installation using a Panel Method (VSAERO) and a Multigrid Euler Method (MGAERO), ICAS-94-2.2.1, (1994).

LEVY, D., WARINER, D. and NELSON, E., Validation of Computational Euler Solutions for a High Speed Business Jet, AIAA Paper 94-1843, (1994).

MELTON, J. E., BERGER, M. J., AFTOSMIS, M. J. and WONG, M. D., 3D Applications of a Cartesian Grid Euler Method, AIAA Paper 95-0853, (1995).

MELTON, J. E., Automated Three-Dimensional Cartesian Grid Generation and Euler Flow Equations for Arbitrary Geometries (Ph. D. Thesis), University of California, Davis, California, (1996)

MELTON, J. E., ENOOTO, E. Y. and BERGER, M. J., 3D Automatic Cartesian Grid Generation for Euler Flows, AIAA Paper 93-3386-CP, (1993).

MITCHEL, R. A., SALAS, M. D. and HASSAN, H. A., Grid Embedding Technique Using Cartesian Grids for Euler Solutions, *AIAA Journal*, 26, 754-756, (1988).

MORINISHI, K., A Finite-Difference Solution of the Euler Equations on Non-Body-Fitted Cartesian Grids, *Computers & Fluids*, 21, 331-344, (1992).

PEMBER, R. B., BELL, J. B., COLELLA, P. CRUTCHFİELD, W. Y. and WELCOME, M. L., An Adaptive Cartesian Grid Method for Unsteady Compressible Flow in Irregular Regions, *Journal of Computational Physics*, 120, 278-304, (1995).

POPINET, S., GERRIS: A Tree-Based Adaptive Solver for teh INcompressible Euler Equations in Complex Geometries, *Journal of Computational Physics,* 190, 572-600, (2003).

PURVIS, J. and BURKHALTER, J., Prediction of Critical Mach Number for Store Configurations, *AIAA Jol*. 17,11, (1979).

QIAN, L., CAUSON, D. M., INGRAM, D. M. and MINGHAM, C. G., Cartesian Cut Cell Two-Fluid Solver for Hydraulic Flow Problems, *Journal of Hydraulic Engineering*, 129, 688-696, (2003).

QUIRK, J. J., An Alternative to Unstructured Grids for Computing Gas-Dynamic Flows Around Arbitrarily Complex 2-Dimensional Bodies", *Computers & Fluids*, 23, 125-142, (1992).

TIDD, D. M., STRASH, D. J., EPSTEİN, B., LUNTZ, A., NACHSHON, A. and RUBIN, T., Multigrid Euler Calculations over Complete Aircraft, *Journal of Aircraft*, 29, 1080-1085, (1992).

WERTERLEN, T. J. and KARMAN, S. L., Rapid Assessment of F-16 Store Trajectories Using Unstructured CFD, AIAA Paper 95-0354, (1995).

WU, Z. N. and LI, K., Anisotropic Cartesian Grid Method for Steady Inviscid Shocked Flow Computation, *International Journal for Numerical Methods in Fluids*, 41, 1053-1084, (2003).

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje No: 106M471

Proje Başlığı: Kartezyen Hesaplama Ağları için Üç Boyutlu Euler Çözücüsü Geliştirilmesi

Proje Yürütücüsü ve Araştırmacılar: Prof. Dr. M. Halûk AKSEL, Y. Doç. Dr. Cüneyt SERT, Makina Yük. Müh. Mehtap ÇAKMAK, Makina Yük. Müh. Bercan SİYAHHAN

Projenin Yürütüldüğü Kuruluş ve Adresi: Makina Mühendisliği Bölümü, Orta Doğu Teknik Üniversitesi, 06531 Ankara

Destekleyen Kuruluş(ların) Adı ve Adresi:

Projenin Başlangıç ve Bitiş Tarihleri: 1 Şubat 2007 – 1 Ağustos 2009

Öz (en çok 70 kelime)

Bu proje çerçevesinde Kartezyen hesaplama ağları için zamana bağlı olmayan üç boyutlu bir Euler çözücüsü geliştirilmiştir. Bu yöntemle geometrik karmaşıklıklarla ilgili zorluklar ile yapısal ve yapısal olmayan hesaplama ağlarında karşılaşılan akışa ve geometriye yönelik adaptasyon problemleri ortadan kaldırılmıştır. Buna ek olarak, geliştirilmiş olan yazılım tam otomatik olup, ağ üretimi ile çözüm aşamaları arasında kullanıcı müdahalesi gerektirmemektedir. Geliştirilecek yazılım literatürde mevcut deneysel sonuçlarla karşılaştırılarak doğrulanmıştır.

Anahtar Kelimeler: Kartezyen Hesaplama, Euler Akış Çözücüsü, Geometrik Adaptasyon, Çözüm Adaptasyonu, Çoklu Ağ Yöntemi

Projeden Yapılan Yayınlar: Bu proje çerçevesinde gerçekleştirilen çalışmaları içeren makale Science Citation Index tarafından taranan bir dergiye yayınlanmak üzere gönderilme aşamasındadır.