VI. ULUSAL HAVACILIK VE UZAY KONFERANSI **UHUK-2016-126**
28-30 Eylül 2016, Kocaeli Üniversitesi, Kocaeli

İKİ BOYUTLU YAPISAL OLMAYAN ÇÖZÜM AĞI ÜRETİLMESİ İÇİN DELAUNAY TEMELLİ BİR ALGORİTMA GELİŞTİRİLMESİ

Semih Akkurt* ve Mehmet Şahin†
İstanbul Teknik Üniversitesi
Uçak ve Uzay Bilimleri Fakültesi, 34469 Maslak, İstanbul

ÖZET

Bu çalışmada özel olarak kanat profilleri üzerinde akış problemlerinin çözümü için yapısal olmayan ağ üretebilen bir algoritma geliştirilmiştir. Bu amaçla literatürde mevcut olan birkaç yöntem geliştirilerek FORTRAN95 ortamında kodlanmıştır. Ek olarak viskoz çözümler için gerekli sınır tabaka çözüm ağı üretimi ayrıca irdelenmiş ve geliştirilen algoritmalarla uyumlu olarak eklenmiştir. Çalışma kapsamında üretilen çözüm ağları iki boyutlu üçgen elemanlar ile sınırlı tutulmuştur. Elde edilen çözüm ağlarının uygunluğu SU^2 çözücüsünde test edilmiştir.

GİRİŞ

Yapısal olmayan ağlar, hesaplama alanlarının hızlı ve verimli bir şekilde sonlu küçük düzenli elemanlara bölünmesinde oldukça yaygın bir şekilde kullanılmaktadır. Yapısal ağlar ile karşılaştırıldığında daha karmaşık bir data yapıları mevcuttur ve bu ağlar üzerinde çalışan algoritmalar daha karmaşık nümerik yöntemler gerektirir. Ancak bu karmaşık yapılarına karşın, yapısal olmayan ağlar bölgesel olarak element boyutlarının kontrol edilebilmesine olanak sağlamaktadır. Ama en önemli avantajı karmaşık geometriler etrafında bu tür ağları oluşturabilmek için bir çok yöntemin mevcut olmasıdır. Literatürde yapısal olmayan çözüm ağı üretme yöntemleri 4 temel başlıkta toplanabilir [Mavriplis, 1995]. Bunlar sırasıyla:

- Dörtlü/Sekizli Ağaç (Quadtree/Octree) Temelli
- İlerleyen Sınır (Advancing Front) Tekniği
- Delaunay Yöntemi
- Hibrit Yaklaşımlar (Delaunay-Advancing Front)

*Lisans Öğr. Uçak Müh. Böl., E-posta: akkurtse@itu.edu.tr
†Doç. Dr., Uzay Müh. Böl., E-posta: msahin@itu.edu.tr

Dörtlü/Sekizli Ağaç Temelli: Bu yöntemde çözüm ağı üretilmesi, çözüm ağı üretilecek bölgenin özyinelemeli (recursive) olarak Kartezyen elemanlara ayrılmasına dayanır. Bu ayrılma 2 boyutta bir dikdörtgenin 4 eş dikdörtgene bölünmesi, 3 boyutta ise bir dikdörtgenler prizmasının 8 eş dikdörtgenler prizmasına bölünmesi olarak karşımıza çıkar. Genellikle özyineleme şartı olarak yeni elde edilen hücrelerin başlangıçta verilen sınır bölgesini içerip içermediği bilgisi kullanılır ve bunun yanında işlemin sonsuza gitmemesi için maksimum özyineleme sayısı tanımlanır. Bir seviyeden diğer seviyeye geçişin çok hızlı olmaması için, genellikle bir seviyeden yan yana en az kaç kademenin olacağı da belirtilerek çözüm ağının daha kaliteli olması sağlanır.

İlerleyen Sınır Tekniği: İlerleyen sınır tekniğinde sınır tabiri 2 boyutta eğri, 3 boyutta ise yüzey olarak karşımıza çıkar. Bu sınır için kendi boyut seviyesinde çözüm ağı üretilmesi ilk olarak gerçekleştirilmelidir. Bu sınırın kendi boyut seviyesindeki çözüm ağının kaliteli olması, sınırlar arasında kalan bölgedeki çözüm ağının kaliteli üretilmesinin ön şartıdır. Sınırın bir kenarı veya yüzeyi kullanılarak üretilen her bir eleman, sınır olarak tanımlanmış bölgenin güncellenmesine yol açar. Her bir sınırdan aynı anda başlayan bu işlemin sınırların karşılaşma noktasında eleman boyutlarında bir süreksizliğe yol açmaması için, her nokta ekleme işleminde arka planda yer alan bir çözüm ağı boyutu fonksiyonu dikkate alınmalıdır.

Delaunay Yöntemi: Delaunay yöntemi simpleksler kullanılarak çözüm ağı üretmeye yarar. Diğer yöntemlere kıyasla çok daha hızlı çözüm ağı üretilmesini sağlar. Matematiksel olarak bakıldığında, iki boyutta bir çözüm ağında bulunan simplekslerin, yani üçgenlerin, çevrel çemberinin düzlemde bulunan başka herhangi bir noktayı içermemesi şartını gerektirir. Düzlemde bir nokta setinin Delaunay üçgenlemesi, bazı özel durumlar haricinde tekildir ve her zaman minimum açının maksimum olmasını sağlar [Lee, Schachter, 1993]. Bu yöntemde her bir nokta ekleme prosedürü, bölgesel olarak o noktayı içeren her elemanın belirlenmesi ve yeni noktanın da kullanılarak o bölgenin yeniden Delaunay şartına uygun düzenlenmesine dayanır.

Hibrit Yaklaşımlar: Hibrit yaklaşımlar, ilerleyen sınır tekniğinin kaliteli çözüm ağı üretmesinden ve Delaunay yöntemlerinin hızından ve matematiksel basitliğinden yararlanır. Delaunay özelliğini her zaman koruyarak, ilerleyen sınır tekniğiyle noktalar eklenmesi bu yaklaşımda genel olarak izlenen yöntemdir.

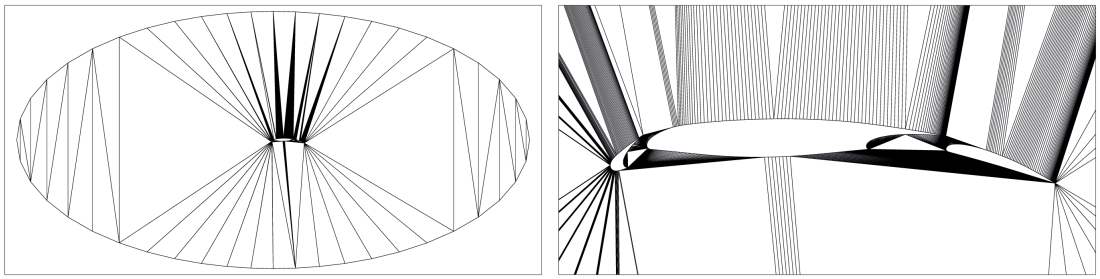
YÖNTEM

Akış problemlerinin nümerik yollarla çözülmesinde kullanılan çözüm ağlarının gerekliliklerini karşılayan bir yazılım geliştirebilmek için literatürde bulunan yöntemler taranmış, bu yöntemlerin iyi yönleri birleştirilerek bir kod geliştirilmiştir. İlk aşamada çalışma 2 boyutlu olarak sınırlandırılmıştır. Geliştirilen kod ile 4 farklı yaklaşımla 2 boyutlu, yapısal olmayan, üçgensel çözüm ağı üretmek mümkündür. Literatürden seçilen 3 adet yöntem geliştirilerek kullanılmış, daha sonra bu çalışma için özel olarak geliştirilen bir yöntem önerilmiş ve bu yöntem de kodlanarak test edilmiştir.

Delaunay yönteminin matematiksel altyapısı çözüm ağı üretimini kolaylaştıran bir etkendir. Bundan dolayı, çalışma kapsamında kullanılan 4 farklı yöntemin her birinde Delaunay şartı her aşamada korunmaktadır. Delaunay şartının her aşamada korunması sonucunda; üçgen eleman boyutlarında ani sıçramaların, üretilen üçgen elemanların domaini tamamiyle kapsayamama veya üst üste binme durumu gibi problemlerin oluşumunun önüne geçilmiştir. Bunun yanısıra, Delaunay şartı, çözüm ağındaki noktalar sabit olmak koşuluyla, üçgen elemanların minimum açılarını maksimize etmektedir. Bu özellikleri sayesinde Delaunay yöntemi akış problemleri için kullanılacak çözüm ağlarının sahip olması gereken özellikleriyle büyük ölçüde örtüşmektedir. Uygulanan algoritmaların her biri çözüm ağı üretmek için gerekli noktaların birer birer eklenmesi esasına dayalıdır. Noktaların çözüm ağına eklenme aşamasında Bowyer-Watson algoritması kullanılmaktadır. Bowyer-Watson algoritması Delaunay şartını sağlayan çözüm ağlarında bu şartı korumayı garanti

ederek eklenen noktayı çözüm ağına dahil eder. Bu algoritma çözüm ağını yalnızca lokal olarak değiştirdiğinden, harcadığı zaman bakımından çözüm ağının büyüklüğünden bağımsızdır.

Çalışma kapsamında uygulanan tüm algoritmalar, yalnızca sınır noktaları kullanılarak oluşturulmuş, ilk üçgenleme olarak adlandırılan çözüm ağını kullanır (Şekil 1). İlk üçgenlemede yer alan

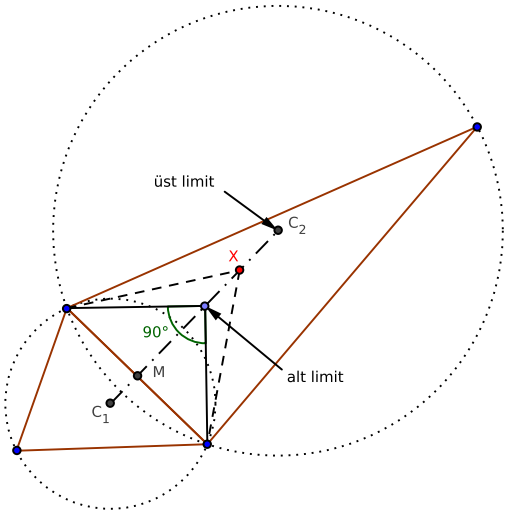


Şekil 1: İlk üçgenleme.

elemanlar uygun bir çözüm ağı tanımlamadığından, sıklaştırılma işlemine tabi tutulmaları gerekmektedir. Bu noktada, üçgensel çözüm ağı üreten algoritmaların birbirinden farklı olduğu iki önemli kriter bulunmaktadır [Baker, 2005; Muller, 1996]. Bu kriterler çözüm ağına eklenecek noktanın konumunun belirlenmesi ve sıklaştırma işlemi için kullanılacak üçgenlerin öncelik sıralamasıdır. Literatürde var olan yöntemlerin hepsi bu iki kriterin farklı kombinasyonlarından oluşmaktadır. Buna ek olarak, bazı algoritmalarda domainin belirli bölgelerinde daha sık elemanlar üretebilmek amacıyla bir boyut fonksiyonu tanımlanmaktadır. Boyut fonksiyonu domainin her noktasında tanımlıdır ve basitçe üçgenler için maksimum çevrel çember çap değeri belirtir. Bir üçgen elemanın boyut fonksiyonunca uygun olup olmadığı, üçgenin çevrel çemberinin merkezindeki boyut fonksiyonu değeri ve üçgenin çevrel çemberinin çapı kıyaslanarak belirlenir. Çevrel çemberinin çapı boyut fonksiyonundan gelen değerden küçük olan üçgenler, uygun boyutlu üçgenler olarak nitelendirilebilir.

Çözüm ağını sıklaştırmak ve uygun hale getirmek için sınır noktaları haricinde domainin içerisine yeterince nokta yerleştirilmesi gerekmektedir. Bu noktaların konumlarının belirlenmesinde iki temel yöntem kullanılmaktadır. İlk yöntem basitçe, aktif üçgenin çevrel çemberinin merkezine yeni bir noktanın eklenmesidir (Şekil 2). Oldukça hızlı olan bu yöntemin dezavantajı üretilen çözüm ağının kalitesinin görece düşük olması ve eleman boyutlarında kısmi sıçramalara yol açmasıdır. İkinci yöntemde ise noktanın konumu, uygun boyuta ulaşmış komşu üçgen ve boyut fonksiyonu yardımıyla hesaplanır. Boyut fonksiyonundan gelen değere göre nokta, alt limit ve üst limit arasına yerleştirilir (Şekil 2). Bu sayede üretilen elemanlar yüksek kaliteye sahip olmaktadır. Fakat uygulanabilmesi için komşu elemanlardan en az birinin uygun boyuta ulaşmış olması gerekmektedir. Bu sebepten ikinci yöntem ilerleyen sınır tekniğiyle çalışan algoritmalarda sıklıkla kullanılmaktadır.

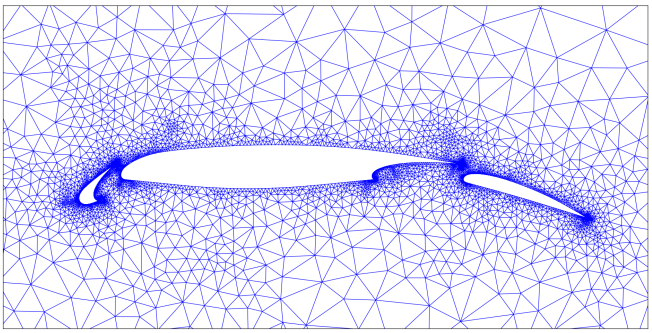
Çözüm ağı sıklaştırma işleminin sağlıklı sonuç verebilmesi için uygun boyuta ulaşmamış, yani kötü üçgenlerin bir listede tutulması, bu listenin sürekli güncellenerek çözüm ağı üretme sürecinin listedeki elemanlar üzerinden devam ettirilmesi gerekmektedir. Çözüm ağı sıklaştırılırken nokta eklemek için kullanılacak üçgenlerin seçilme kriteri algoritmaya bağlı değişmektedir. Tüm algoritmalarda, algoritmaya has öncelik sıralaması kriterine göre belirlenen kötü üçgenler, linkli listeler aracılığıyla tutulmaktadır ve bu sayede listeye eleman ekleme ve çıkarma işlemi oldukça hızlı gerçekleştirilebilmektedir. Bunun yanında, linkli listede tutulan kötü üçgenler, öncelik sıralamasına göre sıralı halde tutulduğu gibi, ilk giren ilk çıkar (İGİÇ) kuyruğu (FIFO queue) olarak da tutulabilmektedir [Shewchuk, 1996]. Bu yöntemde Bowyer-Watson algoritmasıyla birlikte değişen üçgenler listenin sonuna eklenmektedir ve sıklaştırma işlemine daima listenin ilk elemanı



Şekil 2: Nokta konumu belirleme şematiği.

kullanılarak devam edilir. İGİÇ yöntemi kullanıldığında, algoritmaların oldukça hızlandığı sonuç bölümünde Çizelge 1,2,3 ve 4'te karşılaştırılmalı olarak sunulmuştur.

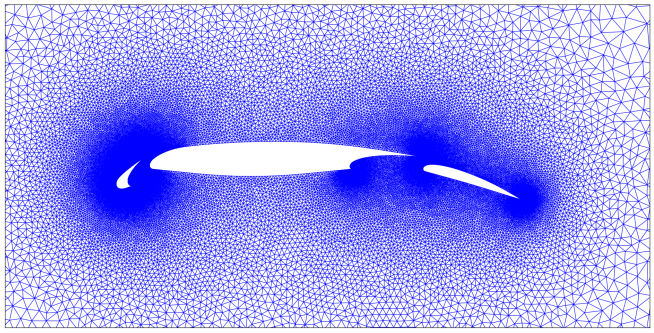
Holmes-Snyder algoritması [Holmes, Snyder, 1988], çözüm ağı üretimi için üçgen kalitesi şartını kullanır. Üçgen kalitesi üçgenin iç teğet çemberinin çapının, dış teğet çemberinin çapına oranı olarak tanımlanmıştır. Eşkenar üçgen için 0.5 olan bu oran, daha makul bir ölçekte değerlendirilebilmesi için normalize edilmiştir. Çözüm ağı üretme süreci üçgen kalitesine göre en kötü üçgenden başlar ve çözüm ağındaki her bir elemanın kalitesi başta kullanıcı tarafından belirtilen kaliteye ulaşana kadar devam eder. Bu süreç boyunca kalitesi minimum değerin altında olan tüm elemanlar kötü üçgenler listesinde tutulmaktadır. Çözüm ağı üretme sürecinde kullanılan noktalar en basit yöntem olan üçgenlerin çevrel çemberlerinin merkezleri olarak seçilmiştir. Holmes-Snyder algoritması diğer algoritmalara göre oldukça hızlıdır fakat çözüm ağı kalitesi fark edilir ölçüde zayıftır. Buna rağmen bu algoritma örneğin Euler denklemleri için çözüm ağı üretiminde oldukça efektif bir şekilde kullanılabilir.



Şekil 3: Holmes-Snyder algoritması.

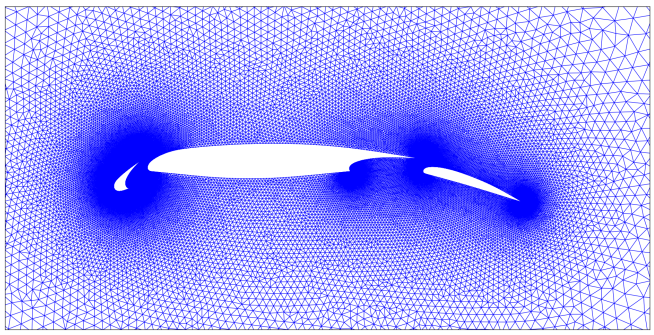
Rebay'ın birinci algoritması [Rebay, 1993], Holmes-Snyder algoritmasına boyut fonksiyonu ekleyerek daha kaliteli çözüm ağları üretilmesini sağlar. Kötü üçgen kriteri ise üçgenin çevrel çemberinin çapının, boyut fonksiyonundan gelen değere oranı olarak belirlenmiştir. Bir önceki algoritmada olduğu gibi, bu algoritmada da çözüm ağına eklenecek yeni noktalar, üçgen elemanların çevrel çemberlerinin merkezi olarak seçilmiştir. Üretilen üçgen elemanlar her adımda

boyut fonksiyonu kullanılarak kontrol edilir ve domaindeki her bir üçgen boyut fonksiyonunun tanımladığından daha küçük bir çevrel çember çapına ulaşana kadar algoritma sıklaştırma işlemine devam eder. Holmes-Snyder algoritmasında olduğu gibi bu algoritmada da, uygunluk şartını sağlayamamış elemanların tamamı, kötü üçgenler listesinde tutulmaktadır.



Şekil 4: Rebay'ın birinci algoritması.

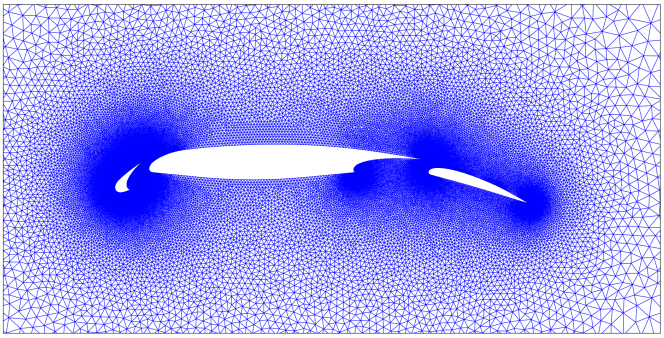
Rebay'ın ikinci algoritması [Rebay, 1993], çözüm ağı üretme sürecinde kullanılacak kötü kaliteli üçgenler listesinin domaindeki tüm kötü üçgenleri tutması hem bellek bakımından hem de listenin güncellenmesi konusunda sıkıntı yaratmaktadır. Rebay'ın ikinci algoritması bu sıkıntının önüne geçebilmek için çözüm ağı üretiminde ilerleyen sınır benzeri bir yaklaşım kullanmaktadır. Çözüm ağı üretilirken kullanılacak üçgen elemanlar yalnızca, en az bir adet uygun boyuta ulaşarak boyut fonksiyonu şartını sağlamış, yani son haline ulaşmış üçgen elemana komşu olanlar arasından seçilir. Bu sayede yalnızca uygun boyuta ulaşmış üçgenlerin komşu üçgenleri üzerinden işlem yapılmakta, domainin geri kalanında bulunup uygun boyuta ulaşmış üçgenlerle komşu olmayan üçgenlerin kötü üçgenler listesinde tutulması gereksiz olduğundan elimine edilebilmektedir. İlerleyen sınır yaklaşımı sayesinde, yeni eklenecek noktaların koordinatları son haline ulaşmış üçgenler kullanılarak hesaplanabilir. Sonuç itibariyle ortaya çıkan çözüm ağının kalitesi arttırılmıştır.



Şekil 5: Rebay'ın ikinci algoritması.

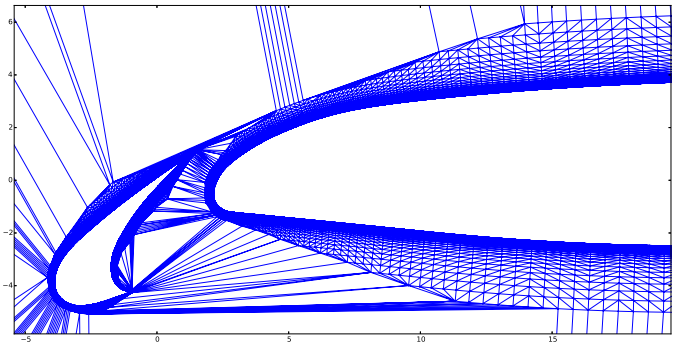
İlerleyen ikili sınır algoritması bu çalışma için özel olarak geliştirilmiştir. Rebay'ın ikinci algoritmasında olduğu gibi kötü üçgenler listesinin olabildiğince kısa olması amaçlanmıştır. Kötü üçgenler listesinin yalnızca bir adet uygun boyuta ulaşmış üçgen elemanlara komşu olanlar üzerinden nokta konumlarının hesaplanması, uygun boyutlu ve uygun boyutlu olmayan üçgen elemanlar arasındaki sınır olarak tabir edilebilecek bölgenin karmaşık olmasına sebep olmaktadır.

Düzgün bir ilerleyen sınır benzeri bir yapı oluşmadığından, olması gerekenden çok daha fazla eleman uygun boyutlu üçgenlere komşu olmakta ve bu durum kötü üçgenler listesinin gereksiz yere daha fazla eleman içermesine sebep olmaktadır. Rebay'ın ikinci algoritmasında olduğu gibi kötü üçgenleri yalnızca bir adet uygun boyutlu üçgene komşu olanlardan seçmek yerine, iki adet komşusu uygun boyuta ulaşmış üçgen elemanlar kullanmak sınır bölgesinin çok daha düzenli olmasını ve üretilen üçgenlerin daha kaliteli olmasını sağlamaktadır. Rebay'ın ikinci algoritmasının aksine çözüm ağında sınırların karşılaştığı yerlerde üçgen kalitesinde düşüş gözlemlenmemektedir. Bu değişiklik Rebay'ın ikinci algoritmasına göre çok küçük olsa da kodlama açısından bazı zorluklar doğurur fakat karşılığında birkaç önemli avantaj sağlar.



Şekil 6: İlerleyen ikili sınır algoritması.

Yüksek Reynolds sayılı akışların çözümü için sınır yakınlarında özel bir geometriye sahip elemanlar gereklidir. Sınır civarında oluşan yüksek gradyan ancak yüksek gradyan yönünde kısa elemanlar kullanılarak yakalanabilir. Bu kriterlere sahip duvar yakınlarındaki çözüm ağları sınır tabaka çözüm ağı olarak adlandırılmaktadır. Çalışma kapsamında sınır tabaka çözüm ağı üretimi için tüm algoritmalarla uyumlu olarak kullanılabilen bir modül geliştirilmiştir. Bu modül çözüm ağı üretme algoritmaları başlamadan önce çağırılarak gerekli sınır tabaka çözüm ağını üretir (Şekil 7). Kullanıcının sınır tabaka çözüm ağı için ilk tabaka kalınlığı artış oranı ve çeşitli sonlandırma kriterleri gibi birkaç parametre belirlemesi gerekmektedir. Sınır tabaka çözüm ağı tabaka tabaka ilerlenerek oluşturulmaktadır. Her katmanda, bir alt katmandan bir kademe daha büyütülerek oluşturulan üçgenlerin eş yönlülüğü kontrol edilir. Eş yönlülükten sapan üçgenler, aktif katmanda oluşturulmaz ve daha sonra bu eleman üzerinden yeni katmanların oluşumunda kullanılmaz. Bu sayede sınır tabaka profilin her yerinde aynı kalınlıkta değildir ve sınır tabakanın oldukça ince olduğu hücum kenarı civarında gereksiz kalınlıkta sınır tabaka elamanlarının oluşumunun önüne geçilmiştir.



Şekil 7: Sınır tabaka çözüm ağı örneği.

Dört çözüm ağı üretimi algoritmasının üçü için alanda boyut fonksiyonu tanımlanmış olmalıdır. Bu çalışmada kullanılan boyut fonksiyonu sınırdaki noktaların birbirlerine olan uzaklığından yola

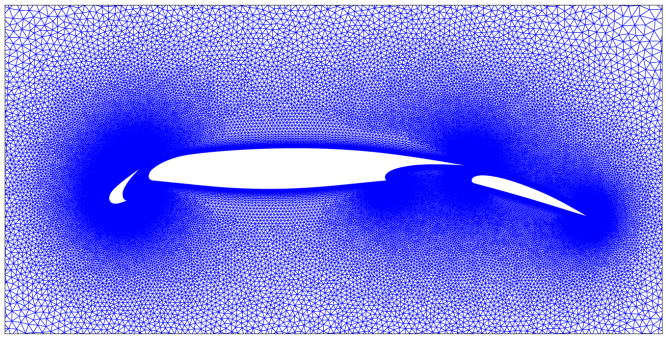
çıkılarak ters kare yöntemiyle hesaplanmıştır. Kullanıcı bunun yerine kendi boyut fonksiyonunu kullanmayı seçebilir.

Veri yapısı olarak dizi (array) tabanlı standart sonlu eleman veri tipi yerine işaretçi (pointer) tabanlı bir veri yapısı kullanılmıştır. Bu veri yapısında her eleman için elamanın 3 komşusuna ve kendisini oluşturan 3 noktasına olmak üzere toplamda 6 adet işaretçi saklanmaktadır. İşaretçi tabanlı bu yöntem dizi tabanlı yönteme göre çok ciddi avantajlar içermektedir ve çözüm ağı üretimini önemli ölçüde hızlandırmıştır.

Üretilen çözüm ağlarının çözücüler tarafından çalıştırılıp çalıştırılamadığı, çalıştırılabiliyorsa ne gibi problemler doğurduğunu bilmek daha kaliteli çözüm ağları üretmek için bir ön koşuldur. Dolayısıyla, üretilen çözüm ağlarının uygunluğu SU^2 [Economon, et al., 2016] çözücüsünde sıklıkla denenmiştir. Sonuç olarak elde edilen çözümlerden çıkarılan geribildirimler sayesinde çözüm ağı üretimi algoritması oldukça geliştirilmiştir.

UYGULAMALAR

Geliştirilen çözüm ağı üretme algoritmalarının testi için standart kanat profillerinden farklı olarak üç elemanlı yüksek taşıma üreten bir profil kullanılmıştır. Profilde var olan konkav, konveks bölgeler ve sivri uçlara rağmen algoritmaların hata vermemesi, çözüm ağı üretiminin karmaşık geometrilerde de uygulanabilir olduğunu göstermektedir. Bu profil üzerinde, ilerleyen ikili sınır algoritmasıyla oluşturulan ve sınır tabaka elemanlarını da içeren çözüm ağı Şekil 8'de verilmiştir.



Şekil 8: İlerleyen ikili sınır algoritması ve sınır tabaka.

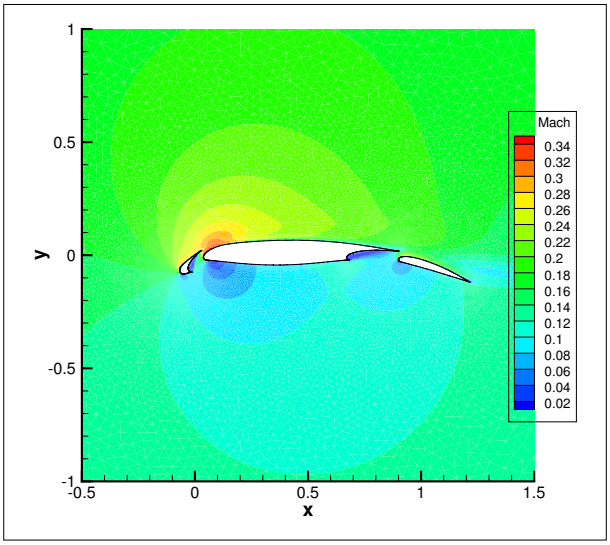
SONUÇ

Bu çalışmada iki boyutlu üçgensel çözüm ağı üretebilen bir yazılım geliştirilmiştir. Ayrıca yüksek Reynolds sayısında akış problemleri için sınır tabaka ağı atmak da mümkün olabilmektedir. Elde edilen üçgensel ağların uygunluğu SU^2 algoritması ile test edilmiştir (Şekil 9).

Ek olarak, algoritmanın FORTRAN95 kodlamasında işaretçiler (pointer) verimli şekilde kullanılarak çözüm ağı üretme zamanı minimize edilmiştir. Ayrıca, çözüm ağı üretimi algoritmaları zaman bakımından karşılaştırılmıştır. Sonuçlar saniye cinsinden Çizelge 1,2,3 ve 4'te verilmiştir. Çizelge 1 ve 2, sıralı liste kullanılarak elde edilen sonuçları; Çizelge 3 ve 4 ise İGİÇ yöntemi kullanılarak elde edilen sonuçları göstermektedir. Sıralı liste ve İGİÇ yöntemi yalnızca, algoritmanın kullandığı kötü

Çizelge 1: 100,000 eleman için CPU zamanı karşılaştırması, sıralı liste.

Algoritma	Toplam Zaman	2B Çözüm Ağı	Boyut Fonksiyonu
Rebay'ın birinci	15.00	12.89	2.11
Rebay'ın ikinci	6.03	2.58	3.45
İlerleyen ikili sınır	5.30	2.07	3.23



Şekil 9: SU^2 ile elde edilmiş çözüm. Mach konturu.

Çizelge 2: 200,000 eleman için CPU zamanı karşılaştırması, sıralı liste.

Algoritma	Toplam Zaman	2B Çözüm Ağı	Boyut Fonksiyonu
Rebay'ın birinci	79.90	70.72	9.18
Rebay'ın ikinci	18.77	12.17	6.60
İlerleyen ikili sınır	14.56	8.38	6.18

üçgenler listesinin oluşturulma biçiminde bir farklılıktır. Sıralı listede elemanlar algoritmanın belirlediği kritere göre küçükten büyüğe sıralı olarak tutulduğundan güncelleme işlemi ciddi zaman kaybına yol açmaktadır. İGİÇ yöntemi kullanıldığında ise her eleman listenin sonuna eklenir, algoritma listenin başından aldığı elemanın uygunluğunu kontrol ederek devam eder. Algoritmalar aynı zamanda açık kaynak kodlu popüler bir çözüm ağı programı olan GMSH [Geuzaine, Remacle, 2009] ile zaman bakımından karşılaştırılmıştır (Çizelge 3,4).

Çizelge 3: 100,000 eleman için CPU zamanı karşılaştırması, İGİÇ listesi.

Algoritma	Toplam Zaman	2B Çözüm Ağı	Boyut Fonksiyonu
Rebay'ın ikinci	3.11	0.19	2.92
İlerleyen ikili sınır	3.71	0.36	3.35
Gmsh	3.10	-	-

Çizelge 4: 200,000 eleman için CPU zamanı karşılaştırması, İGİÇ listesi.

Algoritma	Toplam Zaman	2B Çözüm Ağı	Boyut Fonksiyonu
Rebay'ın ikinci	6.25	0.29	5.96
İlerleyen ikili sınır	7.06	0.50	6.56
Gmsh	6.10	-	-

Son olarak, üretilen çözüm ağlarının kaliteleri normalize edilmiş üçgen kalitesi değeri eleman sayılarına göre karşılaştırmalı olarak verilmiştir (Çizelge 5).

Çizelge 5: Farklı algoritmaların kalite karşılaştırması.

Kalite	Rebay 2 sıralı	Rebay 2 İGİÇ	İlerleyen ikili sıralı	İlerleyen ikili İGİÇ
0.0 - 0.1	0	0	0	0
0.1 - 0.2	0	0	0	0
0.2 - 0.3	0	0	0	0
0.3 - 0.4	0	5	6	7
0.4 - 0.5	10	25	30	31
0.5 - 0.6	57	308	350	341
0.6 - 0.7	291	1,594	1,632	1,575
0.7 - 0.8	2,802	7,821	7,866	7,944
0.8 - 0.9	15,559	40,199	44,812	43,447
0.9 - 1.0	192,688	148,964	150,112	148,542
Total	211,410	198,918	204,808	201,887

KAYNAKLAR

Akkurt, S. (2016). *A Delaunay based algorithm for unstructured mesh generation.* Bitirme tezi. İstanbul Teknik Üniversitesi, Uçak ve Uzay Bilimleri Fakültesi, İstanbul, Türkiye

Baker, T.J. (2005). Mesh generation: Art or science? *Progress in Aerospace Sciences, 41*(1), 26-63.

Economon, T.D., Palacios, F., Copeland, S.R., Lukaczyk, T.W. and Alonso, J.J. (2016). SU2: An Open-Source Suite for Multiphysics Simulation and Design, *AIAA Journal, 54*(3), 828-846.

Geuzaine, C. and Remacle, J.F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, *International Journal for Numerical Methods in Engineering, 79*(11), 1309-1331.

Holmes, D.G. and Snyder, D.D. (1988). The generation of unstructured meshes using Delaunay triangulation, *Numerical Grid Generation in Computational Fluid Mechanics'88,* Pineridge Press, Swansea, United Kingdom.

Lee, D. T., & Schachter, B. J., (1980). Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences, 9*(3), 219-242. doi:10.1007/BF00977785

Mavriplis, D. J., (1995). Unstructured mesh generation and adaptivity. 26th computational fluid dynamics lecture series, rhode-saint-genese, march 1995. *45 Pp,*

Muller, J.D. (1996). *On triangles and flow.* Ph.D. thesis.

Rebay, S., (1993). Efficient unstructured mesh generation by means of delaunay triangulation and bowyer-watson algorithm. *Journal of Computational Physics, 106*(1), 125-138. doi:10.1006/jcph.1993.1097

Shewchuk, J.R., (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, volume1148 of *Lecture Notes in Computer Science, Springer-Verlag*, pp.203-222, from the First ACM Workshop on Applied Computational Geometry