İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

T.C. YÜKSEKÖĞRETİM KURULU DOKÜMANTASYON MERKEZİ

# HESAPLAMALI AKIŞKANLAR MEKANİĞİNDE PARALEL İŞLEM

100898

**YÜKSEK LİSANS TEZİ**
**Uzay Müh. H.Siyami ERSOY**
**(511971102)**

100898

**Tezin Enstitüye Verildiği Tarih : Ekim 2000**
**Tezin Savunulduğu Tarih : Kasım 2000**

**Tez Danışmanı : Prof.Dr. A. Rüstem ASLAN**
**Diğer Jüri Üyeleri Prof.Dr. Kadir KIRKKÖPRÜ**
**Doç.Dr. Fırat O. EDİS**

**KASIM 2000**

# ÖNSÖZ

Bu çalışmada sonlu elemanlar yöntemine dayalı bir parçalı adım şeması ile sıkıştırılamaz akış analizleri için daha önce geliştirilmiş bir bilgisayar programına daha verimli şekilde paralel çözümleme yeteneği kazandırılmıştır.

Değişik ve zevkli bir konuda bana tez yazma olanağı sağlayan, çalışmalarım sırasında yardımlarını hiçbir zaman benden esirgemeyen değerli hocam Prof. Dr. A. Rüstem ASLAN'a, bana her konuda hoşgörü ile yaklaşan Açıkhava Reklamcılık' ın değerli çalışanları M.Gökhan YAVUZ, Tolga YILMAZ, Oral BOLKAN ve Arzu MOROVA'ya, maddi ve manevi desteklerinden ötürü Amcam Tuncer ERSOY'a teşekkürü bir borç bilirim.

Ve tabii ki benim bu günlere gelmemi sağlayan, ilgi, sevgi ve alakalarını benden hiçbir zaman esirgemeyen, özgüvenimi sağlayan biricik aileme sonsuz teşekkür ederim.

Aileme.....

Kasım 2000, İSTANBUL

H.Siyami ERSOY

# İÇİNDEKİLER

# TABLO LİSTESİ

# ŞEKİL LİSTESİ

# SEMBOL LİSTESİ

**e** : Eleman
**[M]** : Toplanmış kütle Matrisi
**[D]** : Taşınım Matrisi
**[A]** : Katılık Matrisi
**[C]** : Basınç Katsayısı Matrisi
**[B]** : Sınır Şartları Katkı Matrisi
**[E]** : Sıkıştırılamazlık Matrisi
**n** : İterasyon Seviyesi
**i** : Bölge Sayısı
**j** : Arakesit Sayısı
**ξ,ψ,ζ** : Eğrisel Koordinat Bileşenleri
**x,y,z** : Kartezyen Koordinatları
**U** : Karakteristik Hız
**l** : Karakteristik Boy
**ν** : Kinematik Viskozite
**Re** : Reynolds Sayısı
**u** : Hız Vektörü
**p** : Basınç Değeri
**t** : Zaman Değişkeni

# ÖZET


Günümüzde sayısal çözüm, yüksek kapasiteli problemlerin çözümünde mühendisler için büyük bir şanstır. Bilgisayarların tek başına hesaplama kapasiteleri göz önüne alınırsa paralel işlem en iyi ve ucuz çözüm yöntemi olmaktadır. Düşük maliyetli iş istasyonları ile yüksek kapasite isteyen problem çözümleri yapılmaktadır.

Bu çalışmada daha önce sıkıştırılamaz akış analizleri için geliştirilmiş ve denenmiş bir yazılım[1-6] 2 ve 3 boyutta paralel işlem yapacak hale getirilmiştir. Açık şema ile çözülen hareket denklemlerinde bu işlem kolayca belirlenen yüzey ve düğümler arasında her zaman adımında birer kere yapılırken Poisson denkleminin tekrarlı çözümünde her tekrarda haberleşme gereksinimi hesaplama zamanını aşırı arttırmaktadır. Bu nedenle basınç denklemi, hız denklemlerinden farklı olarak, bölgelere ayırma (domain decomposition) yöntemi ile çözülmektedir. Paralel hesaplama işlemi PVM [9-10] protokolünü kullanan iş istasyonları ile yapılmaktadır. Bölgeler birbirinin içine girmeyecek ve düğümleri uyuşacak (matching-nonoverlapping) şekilde seçilmektedir.

Geliştirilen yazılım Q1Q1 ve pQ2Q1 eleman tanımlamaları ile sonlu elemanlar yöntemine dayalı bir parçalı adım şeması ile sıkıştırılamaz akışı yöneten denklemleri uzay ve zamanda ikinci mertebe hassasiyet ile çözmektedir. Bu çözüm şemasında hareket denklemleri iki defa açık bir şema ile hız alanını bulmak üzere çözülmektedir. Hız alanını süreklilik denklemini sağlayacak şekilde düzeltmekte ve basınç alanını hesaplamakta kullanılan Poisson denklemi ise kapalı bir şema ile çözülmektedir[7-8].

Bölgelere ayırmada kullanılan metot sırasıyla başlangıç aşaması, birim problem aşaması ve sonuç aşamasından oluşmaktadır.

Başlangıç aşaması : Basınç denklemi, $\partial\Omega_i$ sınırı ve $S_j$ arakesiti ile sınırlandırılmış her bir $\Omega_i$ bölgesi için bölgeler arası arakesitte sıfır akı şartı olacak şekilde çözülür.

$$
\begin{aligned}
-\Delta y_i &= f_i \quad in \quad \Omega_i \\
y_i &= g_i \quad on \quad \partial\Omega_i \\
\frac{\partial y_i}{\partial n_i} &= 0 \quad on \quad S_j
\end{aligned}
\qquad\qquad
\begin{aligned}
\mu^0 &: keyfi \\
g^0 &= \mu^0 - (y_2 - y_1)_{S_j} \\
w^0 &= g^0
\end{aligned}
$$

burada $y_i$ p yerine geçmektedir.

Birim problem: Asıl problemden kaynaklanan arakesit akı şartını çıkarmak için bir birim problem aşağıdaki şekilde yaratılır.

$$\begin{aligned} -\Delta x_i^n &= 0 && in \quad \Omega_i \\ x_i^n &= 0 && on \quad \partial\Omega_i \\ \frac{\partial x_i^n}{\partial n_i} &= (-1)^{i-1} w^n && on \quad S_j \end{aligned}$$

En hızlı iniş (Steepest Descent): Ara yüzeydeki akı değerini hesaplamak üzere iteratif şema kullanılmıştır.

$$aw^n = \left(x_1^n - x_2^n\right)_{S_j} \qquad g^{n+1} = g^n - \beta^n aw^n$$

$$g^0 = \left(y_1^0 - y_2^0\right)_{S_j} \qquad s^n = \frac{\sum_j \left\| g^{n+1} \right\|^2}{\sum_j \left\| g^n \right\|^2}$$

$$\mu^{n+1} = \mu^n - \beta^n w^n \qquad w^{n+1} = g^{n+1} + s^n w^n$$

$$\beta^n = \frac{\sum_j \int_{S_j} \left| g^n \right|^2 ds}{\sum_j \int_{S_j} (aw^n) w^n ds}$$

Yakınsma Kriteri $\left| \frac{\mu^{n+1} - \mu^n}{\mu^1} \right| \le \varepsilon$

Sonuçlandırma: Yakınsamış ara kesit akı şartlarını bulduktan sonra çözüm son olarak aşağıdaki şekilde yapılır.

$$\begin{aligned} -\Delta y_i &= f_i && in \quad \Omega_i \\ y_i &= g_i && on \quad \partial\Omega_i \\ \frac{\partial y_i}{\partial n_i} &= (-1)^{i-1} \mu && on \quad S_j \end{aligned}$$

Bu bölümde *i* ve *j* alt indisleri, sırasıyla, bölge ve arakesitleri *n* üst indisi de iterasyon seviyesini göstermektedir.

Mevcut olan bilgisayar programına, daha genel paralel hesaplama yeteneği kazandırabilmek için aşağıdaki uygulamalar yapılmıştır.

1) Alt bölgelerin keyfi doğrultularda ve keyfi şekillerde yapılabilmesi için bilgi alış veriş sistemi genelleştirilmiştir.

2) İstenildiği kadar işlemci tanımlayabilmek üzere her bir CPU'da birden fazla işlemin tanımlanabilmesi sağlanmıştır.

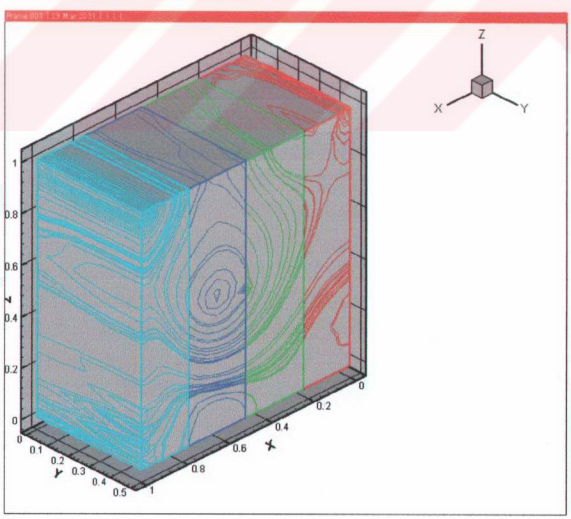
3) Poisson denkleminin Neumann sınır şartı ile çözümünde verilen sabitin bölgeler arası uyumu sağlayacak düzenleme programa eklenmiştir.

Uygulama olarak, daha önce tek ve paralel olarak çözümü bulunan kübik kavite problemi seçilmiştir. Akış Reynolds sayısı diğer uygulamalarda olduğu gibi 1000 olarak seçilmiştir.

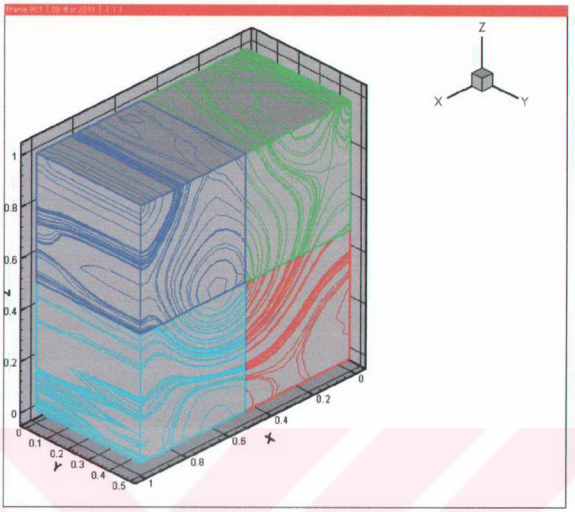
Dört bölgeli ve iki farklı alt bölgelere ayrılmış sayısal ağ ve çözümler Şekil 1 ve 2'de verilmiştir.

En son olarak, kübik kavite probleminin 16 alt bölge için giriş bilgileri genelleştirilmiştir, Şekil 3.

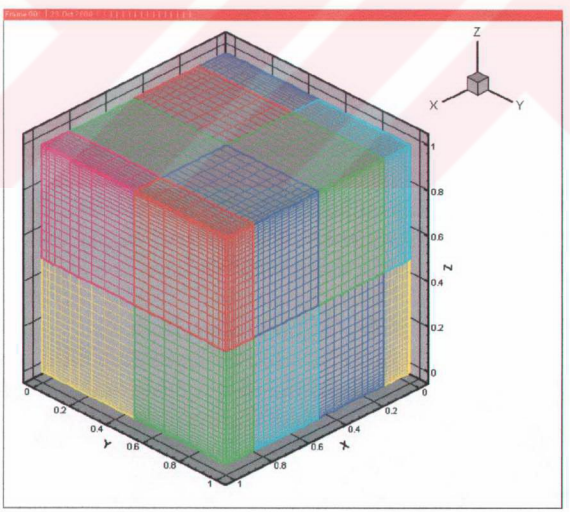
Bundan sonraki uygulamalarda, halihazırda kullanılan Master/Slave uygulamasında iş yükü dağılımını optimize etmek ve işlemciler arası haberleşmeye ve çalışmadan beklemeye giden zamanı kısaltmak açısından işlerin olabildiğince slave'ler tarafından yapılması, Master tarafından olabildiğince işlem yapılmaması yönünde bir düzenlemenin programa eklenmesi de faydalı olacaktır.




Şekil 1. Bir Yönde Bölmeli, Eş Basınç Eğrileri Re=1000




Şekil 2. '+' Bölmeli, Eş Basınç Eğrileri, Re=1000




Şekil 3. 16'lı Bölge Ayrıklaştırması ve Sayısal Ağ

# PARALLEL PROCESS IN COMPUTATIONAL FLUID DYNAMICS

## SUMMARY


Numerical solution of the large-scale engineering problems is one of the biggest challenges of today's research world. Considering the limits of currently available computer power, it became apparent that the only means to achieve this goal is to compute in parallel. Clusters of the low cost work stations or PC's are particularly attractive in constructing such a powerful parallel computing means.

In this study, a previously developed computer code [1-6] which solves the incompressible Navier-Stokes equations numerically sequentially or in parallel to analyze complex flow fields occurring in practical applications is further developed for general parallel computations. In this code, a fractional step algorithm together with Galerkin finite element formulation is employed as the solution procedure. Parallel solutions are obtained in cluster of workstations operating in PVM [9-10] environment. The momentum equation is solved with two-step explicit time marching and the pressure is obtained via solving the Poisson's equation. For solving these two equations in parallel computation, the solution domain is subdivided into sub domains. Both equations are solved in matching/non-overlapping grids.

While the parallel solution of the momentum equations are carried out in a straightforward manner, the Domain Decomposition technique together with the element-by-element (EBE) iteration technique is employed for solution of the Poisson equation for pressure. In addition to use of the Q1Q1 finite elements pQ2Q1 elements are employed for reducing the size of the pressure solution and enhancing the stability of the overall solution [7-8].

Domain Decomposition Method (DDM) employed for pressure formulations uses a sequence of initialization-iteration for a unit problem and finalization procedure.

Initialization: Potential equation is solved in each domain $\Omega_i$ with boundary of $\partial\Omega_i$ and an interface $S_j$, with vanishing Neumann boundary condition on the domain interfaces.

$$
\begin{aligned}
-\Delta y_i &= f_i \quad in \quad \Omega_i \\
y_i &= g_i \quad on \quad \partial\Omega_i \\
\frac{\partial y_i}{\partial n_i} &= 0 \quad on \quad S_j
\end{aligned}
\qquad\qquad
\begin{aligned}
\mu^0 &: \ arbitrarily\ chosen \\
g^0 &= \mu^0 - (y_2 - y_1)_{S_j} \\
w^0 &= g^0
\end{aligned}
$$

where $y_i$ stands for $p$.

Unit Problem: In this DDM a unit problem is set up from the actual problem of solving the Poisson's equation on a subdivided domain as

$$\begin{aligned} -\Delta x_i^n &= 0 && in \quad \Omega_i \\ x_i^n &= 0 && on \quad \partial\Omega_i \\ \frac{\partial x_i^n}{\partial n_i} &= (-1)^{i-1} w^n && on \quad S_j \end{aligned}$$

Steepest Descent

$$aw^n = \left(x_1^n - x_2^n\right)_{S_j} \qquad g^{n+1} = g^n - \beta^n a w^n$$

$$g^0 = \left(y_1^0 - y_2^0\right)_{S_j} \qquad s^n = \frac{\sum_j \left\| g^{n+1} \right\|^2}{\sum_j \left\| g^n \right\|^2}$$

$$\mu^{n+1} = \mu^n - \beta^n w^n \qquad w^{n+1} = g^{n+1} + s^n w^n$$

$$\beta^n = \frac{\sum_j \int_{S_j} \left| g^n \right|^2 ds}{\sum_j \int_{S_j} (aw^n) w^n ds}$$

convergence check: $\left| \mu^{k+1} - \mu^k \right| \le \varepsilon$

Finalization: Having obtained the correct Neumann boundary condition for each interface, the original problem is solved for each domain.

$$\begin{aligned} -\Delta y_i &= f_i && in \quad \Omega_i \\ y_i &= g_i && on \quad \partial\Omega_i \\ \frac{\partial y_i}{\partial n_i} &= (-1)^{i-1} \mu^{n+1} && on \quad S_j \end{aligned}$$

In this section, subscript *i* and *j* indicate the domain and the interface respectively, superscript *n* denotes iteration level.

In this study, the aforementioned computer program is generalized for arbitrary parallel computations as follows:

1) The data relation system was improved and generalized in order to provide the sub domains in arbitrary directions and types.

2) In order to provide the custom quantity of processors, more than one process per CPU was maid possible.

3) A supplementary feature was added especially for the registration of constant, which provides the conformity among sub domains, through the solution of Poisson equation with Neumann boundary conditions.

The classical cubic cavity problem, which was formerly solved sequentially or parallel, was chosen as an example application. Flow Reynolds number was chosen as 1000 as similar to previous applications.

First 4 domain solutions of two different sub domain divisions are obtained. The grid and the solutions obtained are given in Figures 1 and 2 respectively.

Finally a 16 domain arbitrary partition of the cubic cavity along with the required input data is generated, Figure 3.

For future work, obtaining the solution with large number of sub domains and improving the master/slave system for better load balancing and for reducing the communication cost and idle times of the slaves are proposed.

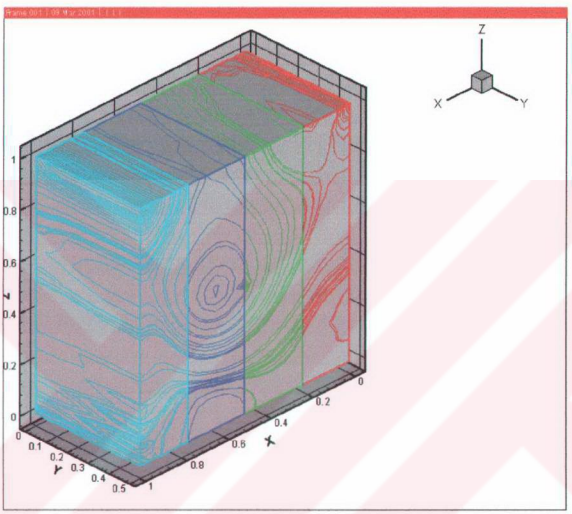



Figure1. One Directional 4 Domains Solution, Pressure Isolines at Outer Boundaries at the Cavitiy, Re=1000

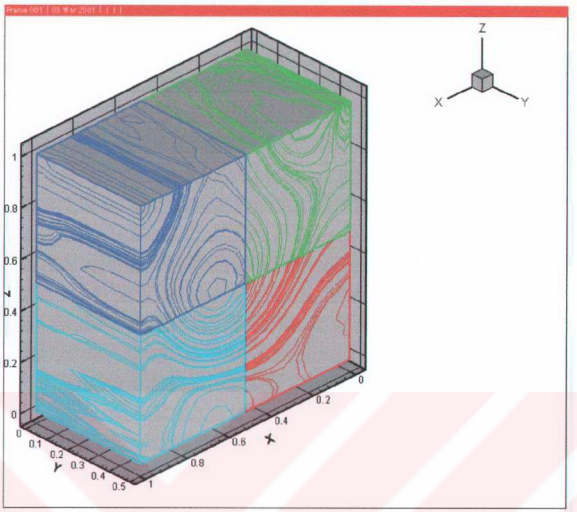



Figure 2. '+' Type 4 Domains Solution, Pressure Isolines at Outer Boundaries at the Cavitiy, Re=1000

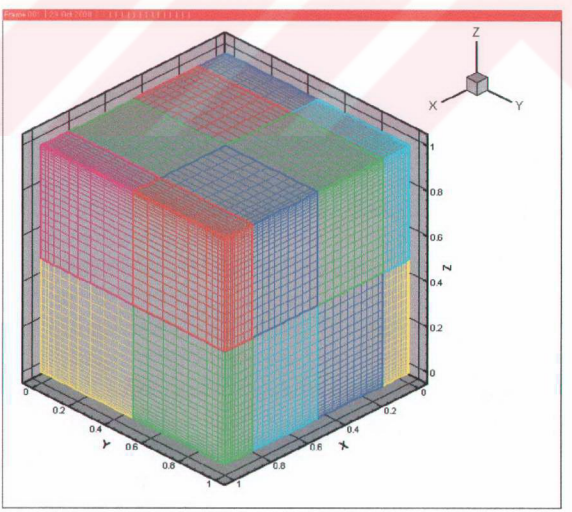



Figure 3. Grid and The 16 Domain Partition at Cubic Cavity

# 1. GİRİŞ

Uçak ve Uzay Bilimleri Fakültesi Uzay Mühendisliği Bölümünde 1996 yılından bu yana dünyadaki eğilimlere benzer olarak yüksek hesaplama gücü gerektiren karmaşık akış problemlerin çözümüne yönelik 'paralel hesaplama' çalışmaları yürütülmektedir.

Bu çalışmalar çerçevesinde daha önce sıkıştırılamaz akış analizleri için geliştirilmiş ve denenmiş bir yazılım[1-2] 2 ve 3 boyutta paralel işlem yapacak hale getirilmiştir[3-6]. Geliştirilen yazılım Q1Q1 ve pQ2Q1 eleman tanımlamaları [6-7-8] ile sonlu elemanlar yöntemine dayalı bir parçalı adım şeması ile sıkıştırılamaz akışı yöneten denklemleri uzay ve zamanda ikinci mertebe hassasiyet ile çözmektedir. Bu çözüm şemasında hareket denklemleri iki defa açık bir şema ile hız alanını bulmak üzere çözülmektedir. Hız alanını süreklilik denklemini sağlayacak şekilde düzeltmekte ve basınç alanını hesaplamakta kullanılan Poisson denklemi ise kapalı bir şema ile çözülmektedir. Bu denklemlerin paralel çözümünde hesaplama bölgesi seçilmiş bir doğrultuda istenen sayıda (kullanılan sistemde en fazla 6) alt bölgeye ayrılmaktadır.Yön seçiminde paralel hesaplamayı kolaylaştırması açısından belirli bir doğrultu seçilmiştir.Çözümü her bir bölgede bütünün parçası olacak şekilde ayrı ayrı yapıldığından, çözümün bütünlüğünü sağlamak üzere bölgeler arası haberleşme ile eksik bilgilerin diğer bölgelere aktarılması ve diğer bölgelerden alınması gerekmektedir. Açık şema ile çözülen hareket denklemlerinde bu işlem kolayca belirlenen yüzey ve düğümler arasında her zaman adımında birer kere yapılırken Poisson denkleminin tekrarlı çözümünde her tekrarda haberleşme gereksinimi hesaplama zamanını aşırı arttırmaktadır. Bu nedenle basınç denklemi, hız denklemlerinden farklı olarak, bölgelere ayırma (domain decomposition) yöntemi ile çözülmektedir. Bu yöntemde her bir bölge bağımsız olarak ele alınırken bölgeler arası sınır şartları yine tekrarlı bir şekilde ancak daha az haberleşme ile verimli bir şekilde çözülmektedir. Bölgeler birbirinin içine girmeyecek ve düğümleri uyuşacak (matching-nonoverlapping) şekilde seçilmektedir.

Bu çalışmada yukarıda belirtilen kısıtlamaları kaldırarak keyfi şekilde paralel işlem yapılmasını sağlayacak düzeltmeler ile bahsedilen program genelleştirilmiştir. Bunlar:

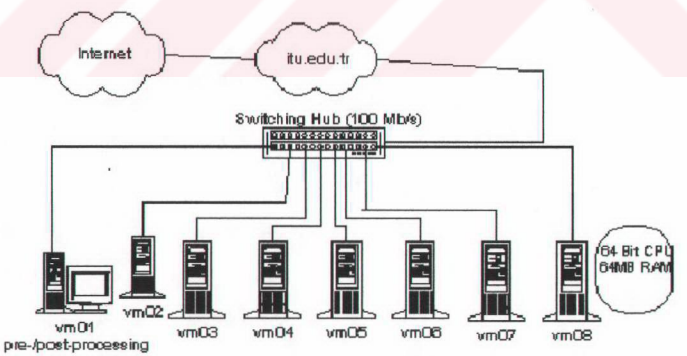
i. İstenildiği kadar işlemci tanımlayabilmek üzere her bir CPU' da birden fazla işlemin tanımlanabilmesi sağlanmıştır.

ii. Alt bölgelerin keyfi doğrultularda ve keyfi şekillerde yapılabilmesi için bilgi alış veriş sistemi genişletilmiş ve genelleştirilmiştir.

iii. ii nolu maddenin uygulanması için özellikle Poisson denkleminin Neuman sınır şartları ile çözümünde atanan sabitin bölgeler arası uyumunu sağlayacak düzenleme programa eklenmiştir.

Uygulama olarak, daha önce tek ve paralel olarak çözülmüş bulunan kübik kavite problemi seçilmiştir. Akış Reynolds sayısı diğer uygulamalarda olduğu gibi 1000 olarak seçilmiştir[3-8].

Paralel hesaplama işlemi İTÜ Araştırma Fonu ve TÜBİTAK COST F1 projeleri ile alınan ve şekil 1.1'de şematik olarak gösterilen 8 iş istasyonundan oluşmuş ve 100 Mbit/s'lik bir hızlı 'switch' ile haberleşen hesaplama sistemi ile yapılmaktadır. Bu iş istasyonları PVM 3.3 (Parallel Virtual Machines)[9] ile haberleşmeyi gerçekleştirirken TCP/IP protokolünü kullanmaktadır.




Şekil 1.1. İş İstasyonu Kümesi

Tezin ikinci bölümde teorik formülasyon kısaca anlatılmıştır. Üçüncü bölümde kullanılan paralel hesaplama teknikleri ayrıntılı anlatılmış, dördüncü bölümde ise elde edilen sonuçlar sunulmuş ve tartışılmıştır. Son bölümde ise değerlendirme ve önerilere yer verilmiştir.

# 2. FORMÜLASYON

## 2.1 Navier-Stokes Denklemleri

Sıkıştırılamaz viskoz akışkanların zamana bağlı akışını karakterize eden denklemler, süreklilik;

$$\nabla.\mathbf{u} = 0 \tag{2.1}$$

ve hareket (Navier-Stokes) denklemleri;

$$\frac{D\mathbf{u}}{Dt} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u} \tag{2.2}$$

olarak verilmektedir. Burada denklemler vektör formunda yazılı olup koyu harfler vektör veya matris büyüklükleri göstermektedir. Değişkenler boyutsuzlaştırılmış olup boyutsuzlaştırma sırasında U karakteristik hız, $l$ karakteristik boy ve $\nu$ kinematik viskozite olarak alındığında Reynolds sayısı, $\mathrm{Re}=Ul/\nu$, boyutsuz parametre olarak yerini denklemlerde almaktadır. Hız vektörü, basınç ve zaman değişkenleri sırasıyla **u**, **p** ve **t** ile gösterilmiştir. Akış denklemleri aynı olmakla birlikte değişik tür akışların varlığında esas rolü sınır ve başlangıç şartları oynamaktadır. Yukarda vektör formunda verilmiş olan diferansiyel denklemlerin çözümleri için sınır ve başlangıç şartları en genel haliyle iç ve dış akış problemlerine uyarlanmıştır[1-2].

## 2.2 Sonlu Eleman Formülasyonu

Sıkıştırılamaz akışı yöneten denklemler, uzay ve zamanda 2. mertebe hassasiyetle çözülmektedir. Hız alanı açık şema ile hesaplanırken yardımcı potansiyel fonksiyonunu ya da basınç alanını hesaplamakta kullanılan Poisson denklemi kapalı bir şema ile çözülmektedir. Çözümün, bir zaman adımı olan $\Delta t$ kadar ilerletilmesi için, $n$ den $n+1$ 'e, çözülmesi gereken matris denklemler sırasıyla,

$$\mathbf{M}\mathbf{u}_\alpha^{n+1/2} = \mathbf{M}\mathbf{u}_\alpha^n + \left[\mathbf{B}_\alpha + p_e\mathbf{C}_\alpha - \left(\frac{\mathbf{A}}{\mathrm{Re}} + \mathbf{D}\right)\mathbf{u}_\alpha\right]^n \frac{\Delta t}{2} \tag{2.3}$$

$$\mathbf{M}\mathbf{u}_\alpha^* = \mathbf{M}\mathbf{u}_\alpha^n + p_e^n\mathbf{C}_\alpha + \left[\mathbf{B}_\alpha - \left(\frac{\mathbf{A}}{\mathrm{Re}} + \mathbf{D}\right)\mathbf{u}_\alpha\right]^{n+1/2} \Delta t \tag{2.4}$$

$$\mathbf{A}p^{n+1} = 2\mathbf{E}_\alpha\mathbf{u}_\alpha^* / \Delta t - \mathbf{A}p^{n+1} \tag{2.5}$$

$$\mathbf{M}\mathbf{u}_\alpha^{n+1} = \mathbf{M}\mathbf{u}_\alpha^* - \frac{1}{2}\mathbf{E}_\alpha(p^{n+1} - p^n)\Delta t \tag{2.6}$$

denklemlerdir. Bu notasyonda $e$ elemanı, $\alpha$ Kartezyen koordinatları olan x,y ve z den birini, **M** toplanmış kütle matrisini, **D** taşınım matrisini, **A** katılık matrisini, **C** basınç katsayı matrisini, **B** sınır şartlarından gelen katkı matrisini ve **E** de sıkıştırılamazlığın göstergesi olan matrisi ifade etmektedir.

Elde edilen bu ayrıklaştırılmış denklemlerle çözümü bir adım ilerletmek için gereken işlemler:

i) (2.3) nolu denklem n+1/2 hız alanını bilinenler cinsinden bulmada kullanılır,

ii) Ara hız değerleri olan $\mathbf{u}^*$, (2.4) denkleminden elde edilir,

iii) Ara hız değerlerini bulduktan sonra basınç (2.5) denkleminden bölgelere ayırma yoluyla hesaplanır,

iv) (2.6)'dan yeni hız alanı hesaplanır,

Bütün bu hesaplarda kütle matrisinin toplanmış hali kullanılmıştır.

### 2.3 Bölgelere Ayırma

Bölgelere ayırmada kullanılan metot sırasıyla başlangıç aşaması, birim problem aşaması ve sonuç aşamasından oluşmaktadır.

Başlangıç aşaması : (2.5) denklemi, $\partial\Omega_i$ sınırı ve $S_j$ arakesiti ile sınırlandırılmış her bir $\Omega_i$ bölgesi için bölgeler arası arakesitte sıfır akı şartı olacak şekilde çözülür.

$$
\begin{aligned}
-\Delta y_i &= f_i \quad in \quad \Omega_i \\
y_i &= g_i \quad on \quad \partial\Omega_i \\
\frac{\partial y_i}{\partial n_i} &= 0 \quad on \quad S_j
\end{aligned}
\qquad
\begin{aligned}
\mu^0 &: \ keyfi \\
g^0 &= \mu^0 - (y_2 - y_1)_{S_j} \\
w^0 &= g^0
\end{aligned}
$$

burada $y_i$ p yerine geçmektedir.

Birim problem: Asıl problemden kaynaklanan arakesit akı şartını çıkarmak için bir birim problem aşağıdaki şekilde yaratılır.

$$
\begin{aligned}
-\Delta x_i^n &= 0 \quad in \quad \Omega_i \\
x_i^n &= 0 \quad on \quad \partial\Omega_i \\
\frac{\partial x_i^n}{\partial n_i} &= (-1)^{i-1} w^n \quad on \quad S_j
\end{aligned}
$$

En hızlı iniş (Steepest Descent): Ara yüzeydeki akı değerini hesaplamak üzere iteratif şema kullanılmıştır.

$$aw^n = \left(x_1^n - x_2^n\right)_{S_j}$$

$$g^0 = \left(y_1^0 - y_2^0\right)_{S_j}$$

$$\mu^{n+1} = \mu^n - \beta^n w^n$$

$$\beta^n = \frac{\sum_j \int_{S_j} |g^n|^2 \, ds}{\sum_j \int_{S_j} (aw^n) w^n \, ds}$$

$$g^{n+1} = g^n - \beta^n a w^n$$

$$s^n = \frac{\sum_j \left\| g^{n+1} \right\|^2}{\sum_j \left\| g^n \right\|^2}$$

$$w^{n+1} = g^{n+1} + s^n w^n$$

Yakınsma Kriteri $\left| \frac{\mu^{n+1} - \mu^n}{\mu^1} \right| \le \varepsilon$

Sonuçlandırma: Yakınsamış ara kesit akı şartlarını bulduktan sonra çözüm son olarak aşağıdaki şekilde yapılır.

$$\begin{aligned} -\Delta y_i &= f_i \quad in \quad \Omega_i \\ y_i &= g_i \quad on \quad \partial\Omega_i \\ \frac{\partial y_i}{\partial n_i} &= (-1)^{i-1}\mu \quad on \quad S_j \end{aligned}$$

Bu bölümde $i$ ve $j$ alt indisleri, sırasıyla, bölge ve arakesitleri $n$ üst indisi de iterasyon seviyesini göstermektedir.

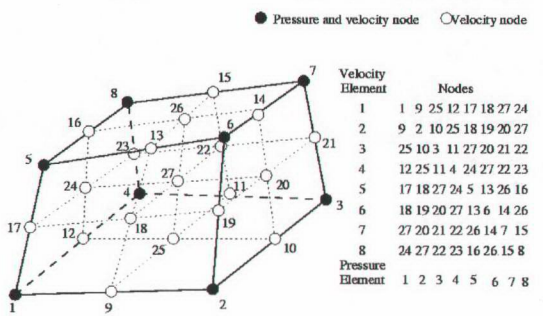
**2.4 Paralel Uygulama**

Paralel uygulama kısmında, usta işlemci tarafından alt bölge çözücüleri olan çırak işlemcilerde uygulama başlatılır. Her çırak kendi bölgesine ait ağ, sınır şartlarını ve başlangıç şartlarını ilgili dosyalardan okur. Alt bölge çözücüleri ara adım hızlarını hesaplayabilmek için (2.3) ve (2.4) denklemlerini arka arkaya iki kez açık olarak çözer. Bölge ara yüzlerindeki düğümler için eleman katkıları bölgeler arasında değiş tokuş edilir. Her alt bölge için Poisson denkleminden ortaya çıkan katılık matrisleri oluşturulur. Basınç değerlerinin bulunması için Poisson denklemini çözmek gerekmektedir. (2.4) nolu denklemin çözümüyle elde edilen ara hız vektöründen o zaman adımındaki basınç değerlerini hesaplamak için (2.5) denkleminde bilinen olarak Poisson denkleminin sağ tarafı oluşturulur. Alt bölgelerde Bölge Ayrıklaştırması çözümü için başlatma aşaması hesaplanır. Bu çözüm paralel ortamda Bölgelere Ayırma yöntemi ile arakesitte iterasyonla yapılmaktadır. Ara yüzde elde edilen düğüm basınçları değiş tokuş edilir. $\beta$'ya (Bakınız Bölüm 2.3) alt bölge katkıları usta işlemeciye gönderilir. Usta işlemci kendisine gelen $\beta$ katkılarının toplamlarını çırak işlemcilere yayınlar. Çıraklar $s$ ve $\mu$ katkılarını hesaplayarak ustaya gönderir. Usta s katkılarını toplayıp sonucu çıraklara bildirir. Usta $\mu$ katkılarına göre yakınsamayı kontrol eder ve çıraklara işaret gönderir. Yakınsama halinde çıraklar sonuçlandırma aşamasına geçerler. Sonuçlandırma aşamasında düğümlerdeki basınç gradyenleri vektörü hesaplanır ve bölge katkıları değiş tokuş edilir. Alt bölge çözücüleri olan çıraklar tam adım hızlarını hesaplar ve bir sonraki zaman adımına geçerler.

## 2.5 Eleman Tipleri

Sayısal çözümleme aracı olarak İTÜ Uçak ve Uzay Bilimleri Fakültesi Uzay Mühendisliği bölümünce geliştirilmiş bulunan bir bilgisayar programı kullanılmıştır[1-8]. Bu program gerek tek bir makine ile analiz ve büyük problemler için de paralel çözümleme olanağı sunmaktadır. Gereksinime göre yine zamanda birinci mertebe ya da ikinci mertebe hassas şema seçilebilmektedir[1-2]. Ayrıca, geometrinin karmaşıklığı ve sayısal ağın oluşturulabilirliğine göre farklı sonlu eleman tipleri de kullanılabilmektedir. Bunlar Q1Q1 ve pQ2Q1 adı verilen elemanlardır. Özellikle pQ2Q1 elemanları aynı akım alanında basınç alan tasvirini üç boyutta 1/8 oranında eleman kullanarak gerçekleştirebilmesi nedeni ile çok faydalı olmaktadır.

Akım alanının, zamanda bir adım, çözülmesi sırasında harcanan bilgisayar zamanının büyük bir yüzdesi basınç çözümünde kullanılmaktadır. Basınç alanı çözümünde çözümün hassasiyetini bozmadan zamandan tasarruf edebilmek için basınç alanını hız alanına oranla daha seyrek bir hesap ağıyla hesaplamak mümkündür[6-8]. Bu yaklaşım sanki ikinci derece hız birinci derece basınç elemanı (pseudo quadratic velocity linear pressure, pQ2Q1 element) olarak bilinmektedir. Seyrek basınç alanlı çözümlerle yapılan uygulamalar paralel hesaplama için bölgelere ayırma yönteminin, özellikle bölgeler arası arakesitlerde, özel uygulamasını gerektirmektedir. Şekil 2.1' de pQ2Q1 elemanı hız ve basınç düğüm noktalarını belirtecek şekilde verilmiştir. Bu şekilde kalın çizgiler bölgeleri, kalın çizgilerin sınırlamış olduğu daha ince çizgiler basınç alanı için ayrıklaştırılmış elemanları en ince çizgiler de hız alanını ayrıklaştırmak için çizilmiştir.




Şekil 2.1. pQ2Q1 elemanında hız ve basınç düğüm noktaları [7].

pQ2Q1 elemanı için ;

Şekil fonksiyonları: $N_p(\xi,\eta,\zeta)=\frac{1}{8}(1-\xi_p\xi)(1-\eta_p\eta)(1-\zeta_p\zeta)$ (2.7)

Hız elemanı: $\xi,\eta,\zeta \in \Omega_e^v$ (2.8)

Basınç elemanı: $\xi,\eta,\zeta \in \Omega_e^p$ (2.9)

olarak tanımlanmıştır.

## 2.6 Neuman Sınır Şartı ile Bölge Ayrıklaştırması (Domain Decomposition) Çözümü İçin Sabitlerin Belirlenmesi

Poisson denkleminin alt bölgelere ayırma ile paralel çözümünde bazı alt bölgelerin tamamen Neuman sınır şartı ile çözümü gerekmektedir. Ancak, bu durumda çözümün tek ve belirli olabilmesi için akım alanında bir noktada çözümün keyfi bir sabit (genellikle sıfır) ile verilmesi gerekmektedir. Benzer durumdaki diğer bir alt bölgede verilecek sabitin diğer sabitlerle uyum içinde olması gerekmektedir. Bu amaçla alt bölgelerde bu sabitlerin tanımlanması ve hesaplanması aşağıdaki şekilde yapılmıştır.


Şekil 2.2. 4 Bölgeli Problemde Neuman Sınır Şartı İle Poisson Denkleminin Çözümü İçin Sabitlerin Belirlenmesi

Bölgeler arasında sabitler aşağıdaki şekilde ilişkilendirilir.

$$\int_{A_{12}}[(\phi_1-C_1)-(\phi_2-C_2)]dA=0 \quad (2.10)$$

$$\int_{A_{23}}[(\phi_2-C_2)-(\phi_3-C_3)]dA=0 \quad (2.11)$$

$$\int_{A_{34}}[(\phi_3-C_3)-(\phi_4-C_4)]dA=0 \quad (2.12)$$

$C_4$=0 olarak alınır ve sabitleri içeren terimler integre edilir ve düzenlenirse:

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} C_1 \\ C_2 \\ C_3 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{A_{12}} \int_{A_{12}} (\phi_1 - \phi_2) dA \\ \frac{1}{A_{23}} \int_{A_{23}} (\phi_2 - \phi_3) dA \\ \frac{1}{A_{34}} \int_{A_{31}} (\phi_3 - \phi_4) dA \end{Bmatrix}$$

elde edilir bu sistem Çözülürse;

$$C_3 = \frac{1}{A_{34}} \int_{A_{34}} (\phi_3 - \phi_4) dA \qquad C_2 = C_3 + \frac{1}{A_{23}} \int_{A_{23}} (\phi_2 - \phi_3) dA$$

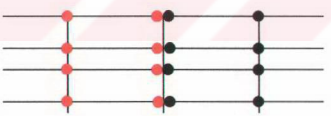
$C_1 = C_2 + \frac{1}{A_{12}} \int_{A_{12}} (\phi_1 - \phi_2) dA$ , olarak bulunur.

Bu sabitlerin belirlenmesi 'Domain Decomposition' yöntemi ile birlikte tekrarlı olarak yapılır. Başlangıçta tüm sabitlere '0' değeri atanır.

# 3. PARALEL HESAPLAMA

## 3.1 Bölge Ayrıklaştırması

Kullandığımız yazılım Q1Q1 ve pQ2Q1 eleman tanımlamaları ile sonlu elemanlar yöntemine dayalı bir parçalı adım şeması ile sıkıştırılamaz akışı yöneten denklemleri uzay ve zamanda ikinci mertebe hassasiyet ile çözmektedir. Kompleks yapıya sahip cisimler üzerinde, Navier-Stokes denklemlerini tek bir makinada ve makul sürede çözmemiz mümkün olamamaktadır. Karmaşık yapıya sahip cisimler etrafındaki akım alanlarını bölgelere ayırarak paralel hesaplama ile çözmek gerekmektedir. Bu denklemlerin paralel çözümünde hesaplama bölgesi seçilmiş bir doğrultuda istenen sayıda (kullanılan sistemde en fazla 6) alt bölgeye ayrılmaktadır. Çözüm her bir bölgede bütünün parçası olacak şekilde ayrı ayrı yapıldığından, çözümün bütünlüğünü sağlamak üzere bölgeler arası haberleşme ile eksik bilgilerin diğer bölgelere aktarılması ve diğer bölgelerden alınması gerekmektedir. Bölgeler birbirinin içine girmeyecek ve düğümleri uyuşacak (matching-nonoverlapping) şekilde seçilmektedir (Şekil 3.1).



Şekil 3.1. Düğümleri Uyuşan Alt Bölgeler

Bölümümüzde bugüne kadar yapılan uygulamalarda, kübik kavite problemi için sıralı bölge ayrıklaştırması yapılmıştır. Neuman sınır şartı ile çözüm yapabilmek için kübik kavite probleminde, kapak olarak adlandırılan üst yüzeyin tümünde basıncın belirli olduğu varsayılarak çözüm yapılmıştır.

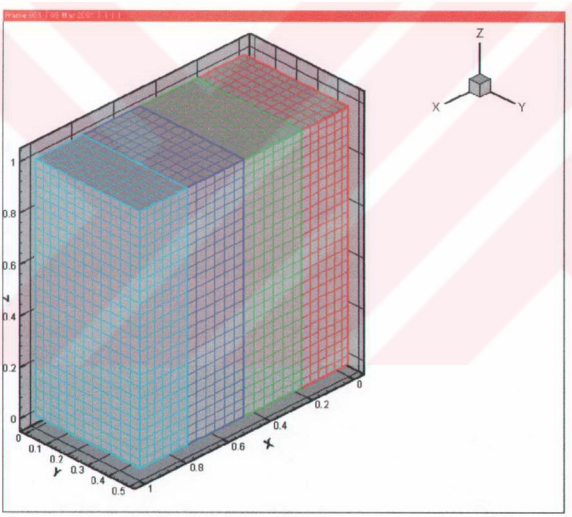
Bu çalışmada yukarıda belirtilen kısıtlamaları kaldırarak daha genel şekilde paralel işlem yapılmasını sağlayacak düzeltmeler ile bahsedilen program genelleştirilmiştir. Bunlar:

1) İstenildiği kadar işlemci tanımlayabilmek üzere her bir CPU' da birden fazla işlemin tanımlanabilmesi sağlanmıştır.

2) Alt bölgelerin keyfi doğrultularda ve keyfi şekillerde yapılabilmesi için bilgi alış veriş sistemi genişletilmiş ve genelleştirilmiştir.

3) 2 nolu maddenin uygulanması için özellikle Poisson denkleminin Neuman sınır şartları ile çözümünde atanan sabitin bölgeler arası uyumunu sağlayacak düzenleme programa eklenmiştir.

Uygulama olarak, daha önce tek ve paralel olarak çözümü bulunan kübik kavite problemi seçilmiştir. Akış Reynolds sayısı diğer uygulamalarda olduğu gibi 1000 olarak seçilmiştir.



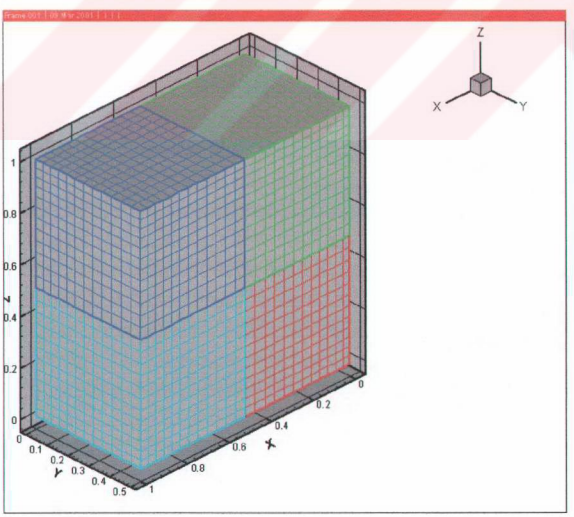

Şekil 3.2. Tek Yönde Bölge Ayrıklaştırması

Bölgelere ayırmada, "Tek yönde ayırma" olarak adlandırabileceğimiz çözüm, alanın bir ekseni boyunca yapılan bölmedir. Bölge ayrıklaştırmasında bölgeleri, mümkün olan en basit şekilde ayırmak ve arayüzeylerde en az nokta olmasına dikkat etmek çözümü kolaylaştırır. Ara yüzeylerin minimum nokta ile temsil edilmesini sağlarsak, bölgelerin birbirlerine gönderdikleri veri sayısı ve haberleşme zamanı da azalır.

Keyfi bölme ile geometriye uygun olarak bölme işlemini gerçekleştirebiliriz. Load Balancing olarak bilinen yük dengesi problemini de iyileştirmiş oluruz. Keyfi bölme ile çözmemiz gereken akım alanını en uygun şartlarla alt bölgelere ayırmak, hafıza ve zaman problemini azaltmakta hem de karmaşık yapıya sahip problemlerin çözümünü basitleştirmektedir.

Bu çalışmada daha önce çözümü, sıralı bölge ayrıklaştırması ile yapılan Kübik kavite problemini (Şekil 3.2) keyfi bölgelere ayırarak çözüm yapılmıştır. İlk iş olarak çözüm bölgemiz '+' olarak adlandırabileceğimiz alt bölgelere bölünmüştür. Alt bölge sayısının artması ile birbirine komşu olan bölge sayıları da artmaktadır. Tek yönde ve '+' şeklindeki bölmede komşu sayısı iki adet iken (Şekil 3.3), 16'lı bölge ayrıklaştırmasında komşu bölge sayıları daha fazla olmaktadır (Şekil 3.5-3.6).

Komşu bölgelerin sayısının artması haberleşmede bazı sorunlarla karşılaşmamıza neden olmaktadır. Arayüzeylerde her alt bölgeye ortak kenarlar ve noktalar yer almaktadır. Bir bölgedeki ortak kenarlar ve noktalar üzerindeki veriler, diğer komşu bölgelere katkı olarak gönderilmiş, gönderilen bu veriler de komşu bölgeler tarafından düzeltildikten sonra tekrar aynı bölgeye geri gönderilerek doğru sonuca ulaşılmıştır.




Şekil 3.3. '+' Şeklinde Bölge Ayrıklaştırması

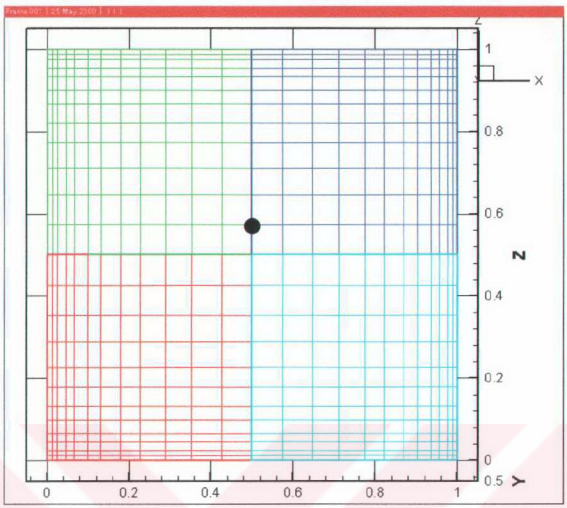
### 3.2 Bir İşlemcide Birden Fazla İşlem

Bölge ayrıklaştırması uygulamalarında her bir işlemciye bir bölgeyi çözdürerek sonuca ulaşmaktaydık. Alt bölgelerimizin, işlemci sayımızla sınırlı olması, her işlemcide bir bölgenin çözümünün yapılmasındandır. Bunu genelleştirmek ve alt bölgelerimizin sayısını arttırarak daha verimli çözümler elde etmek üzere bir işlemciye birden fazla bölgeyi çözdürebiliriz. Bölge sayısını arttırarak, daha basit geometriler elde edebilir ve ara yüzeyleri daha az nokta ile temsil edebiliriz. Ara yüzeylerin daha az nokta ile temsil edilmesi ile işlemciler daha az veri alıp göndereceği için haberleşmeye ayrılan süreleri bu kapsamda kısalmaktadır.

Aynı işlemcide yapılan işler, programın farklı klasörler altından tek bir işlemci üzerinden çalıştırılmasıdır. Bir bölgeden başlayan çözüm, sırasıyla diğer bölgelerle devam eder. Bu sırada diğer bölge ise komşu bölgeden gelecek olan verileri beklemektedir. Bu veriler sırasıyla bölgeler arasında aktarılarak çözüm elde edilir.

### 3.3 Sınır Şartı Uygulamaları

Poisson denkleminin Neuman sınır şartı ile çözümünde, çözümün yapılabilmesi için keyfi bir noktada bir sabitin verilmesi gerekmektedir. Her ne kadar tek bir problemi çözüyor olsak da bölge ayrıklaştırması yaptığımız zaman her bölgeyi birbirinden bağımsız olarak çözmekteyiz. Verilen sabitin de, bölgeler arasında uyumlu olması gerekmektedir. Daha önce uygulanan kübik kavite probleminde ise, tüm kapakta sınır şartı verilmiştir. Aslında tek bir bölgede verilen sınır şartının, çözüm ile birlikte diğer bölgelere taşınması gerekmektedir. Bu yönde yapılan çalışmalar doğrultusunda, '+' olarak bölge ayırmasında bütün bölgelere ortak olan orta noktada sınır şartının verilmesi problemin çözümü için yeterli olmaktadır. Sınır şartı, daha karmaşık problemlerde ise diğer bölgelere, çözüm ile birlikte taşınmalıdır.

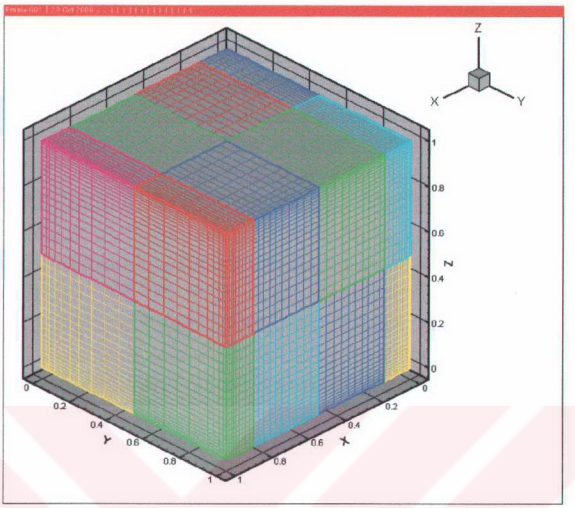



Şekil 3.4. Ortak Noktalı Dört Bölge İçin Sınır Şartı Verilmesi

Bu uygulamada sınır şartı keyfi olarak herhangi bir noktada verilebilirdi. Ancak her bölgede mevcut olan bir noktanın var olması işimizi kolaylaştırmış olup sınır şartı bu noktada verilmiştir (Şekil 3.4).

### 3.4 Keyfi (N) Adet Bölge Ayrıklaştırması

Keyfi bölge ayrıklaştırması ile bir işlemcide birden fazla bölge çözümünü kullanarak alt bölge sayılarını istediğimiz kadar çoğaltırız. Daha önceki uygulamalarda alt bölge sayımız işlemci sayımızla sınırlı iken programda yapılan değişiklikler ile n adet alt bölge oluşturulabilir. Uygulama olarak daha önce keyfi ve sıralı bölme işlemleri ile alt bölgelere ayrılmış ve çözümü yapılmış olan kübik kavite problemi ele alınmıştır. Bu uygulamada kübik kavite 16 alt bölgeye bölünmüştür.

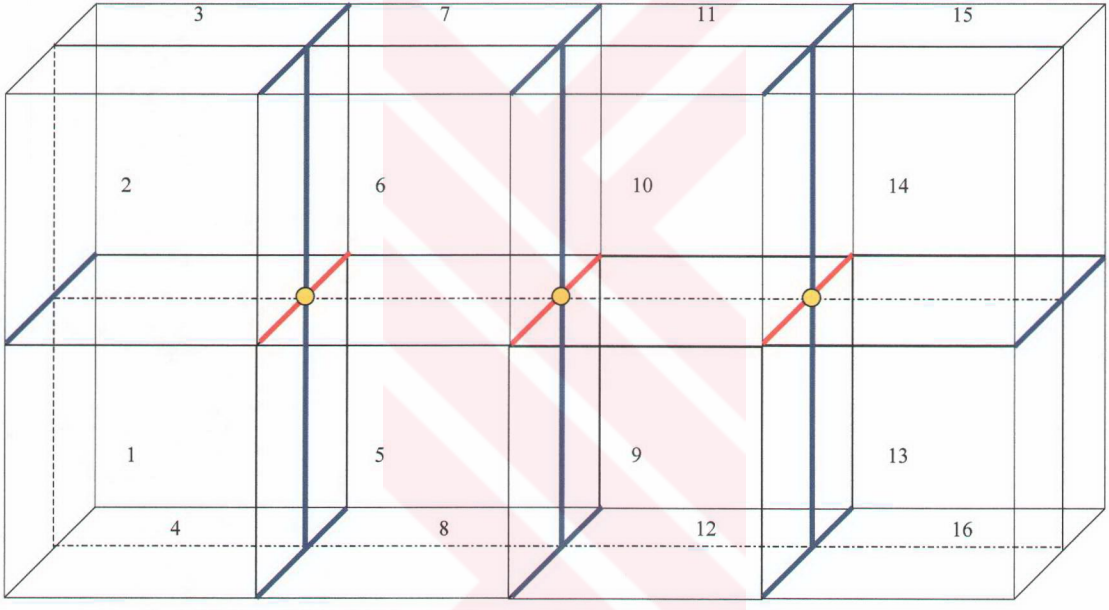



Şekil 3.5. 16'lı Bölge Ayrıklaştırması ve Sayısal Ağ

Şekil 3.6' de görüldüğü üzere her bölgede ortak kenarların yanı sıra ortak noktalar da bulunmaktadır. '+' şeklindeki keyfi bölge ayrıklaştırmasında 1 adet ortak nokta mevcut iken 16'lı keyfi bölmede 14'ü 4 bölgeli ve 3'de 8 bölgeli olmak üzere 17 ortak nokta bulunmaktadır. '+' şeklindeki bölge ayrıklaştırmasında her alt bölgede 2 arayüzey var iken bu uygulamada bazı bölgelerde 3 bazılarında ise 4 arayüzey vardır. Arayüzey sayısının artmasından dolayı, programa hangi bölgenin hangi bölgeye temas ettiğini bildiren kısım ilave edilmiştir.

Sadece 4 bölgeye ortak olan noktalarda çözüm, bir önceki uygulama olan keyfi bölge ayrıklaştırmasında yapıldığı gibi yapılmıştır. Ortak olan noktada ki veriler diğer üç bölgeye gönderilmiş, düzeltilen değerler tekrar aynı bölgelere yollanmıştır. Bu işlem bütün 4'lü ortak noktaların hepsi için yapılmıştır. 8 bölgeye de temas eden noktalar da ise veriler, diğer 7 ortak bölgeye gönderilip geri alınmıştır. Bu işlem 8 bölgeye de ortak olan üç noktada yapılmıştır.

16'lı keyfi bölge ayrıklaştırmasının çözümü için 5 adet iş istasyonu kullanılmış olup bunlardan bir tanesi master makina olup geriye kalan diğer 4 slave makinada 4'er bölge çözdürülmüştür. Aynı makinadaki bölgeler için o makinada farklı klasörler açılıp gerekli olan başlangıç verileri girilmiş ve program çalıştırılmıştır.




Şekil 3.5. 16'lı Keyfi Bölge Ayrıklaştırması, Ortak Kenarlar ve Noktalar

8 Bölgeye Ortak Olan Kenarlar

4 Bölgeye Ortak Olan Kenarlar

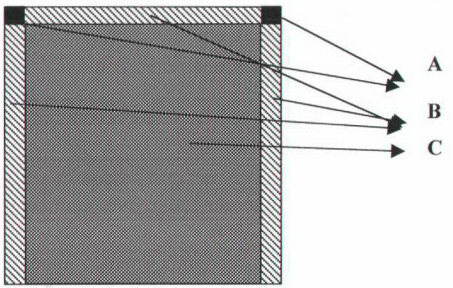
8 Bölgeye Ortak Olan Noktalar

Tablo 3.1. İletişim Tablosu

| CPU | BÖLGE | VERİ GÖNDERİLEN BÖLGE | VERİ ALINAN BÖLGE | VERİ GÖNDERİLEN BÖLGE SAYISI | VERİ ALINAN BÖLGE SAYISI |
|---|---|---|---|---|---|
| 1. | 1 | 2-4 | 5 | 2 | 1 |
| 1. | 2 | 6 | 3-1 | 1 | 2 |
| 1. | 3 | 7-2 | 4 | 2 | 1 |
| 1. | 4 | 3 | 1-8 | 1 | 2 |
| 2. | 5 | 1-6-8 | 9 | 3 | 1 |
| 2. | 6 | 7-10 | 5-2 | 2 | 2 |
| 2. | 7 | 11 | 3-8-6 | 1 | 3 |
| 2. | 8 | 4-7 | 5-12 | 2 | 2 |
| 3. | 9 | 5-10-12 | 13 | 3 | 1 |
| 3. | 10 | 11-14 | 6-9 | 2 | 2 |
| 3. | 11 | 15 | 7-10-12 | 1 | 3 |
| 3. | 12 | 8-11 | 9-16 | 2 | 2 |
| 4. | 13 | 9-16 | 14 | 2 | 1 |
| 4. | 14 | 13 | 10-15 | 1 | 2 |
| 4. | 15 | 14-16 | 11 | 2 | 1 |
| 4. | 16 | 12 | 13-15 | 1 | 2 |
| | | | Toplam : | **28** | **28** |

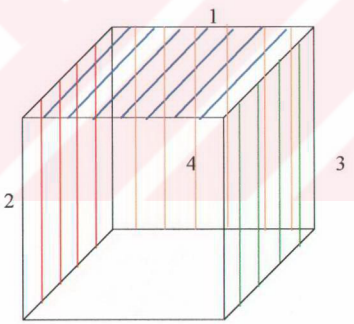
Tablo 3.1'de her bir işlemcinin çözüm yaptığı bölgeler ve bilgi alışverişinin yapıldığı komşu bölgeler yer almaktadır. 16 bölge 4 işlemci ile çözülmektedir. Her bölgede en az bir en fazla da üç veri gönderme ve ya alma işlemi vardır. Dış bölgeler olan 1-2-3-4-12-13-14-15-16 nolu bölgelerde veri alışverişleri üç adet olurken iç bölgelerde bu sayı dört olmaktadır. Bölge ayrıklaştırması ile 16 bölge elde ederken, arayüzey sayımız 28 adet olmuştur. Her bir işlemci bir bölgenin yüzeyinin, hangi bölgenin hangi yüzeyine temas ettiğini bilmektedir. Ara yüzeydeki veriler belirli bir kodla master makinaya gönderilir. Master makina, slave makinalardan gelen verileri düzenli bir şekilde sıralayarak bir dizin haline getirir. Bu sıralama ile farkları alınması gereken doğru değerler birbirinden çıkarılır. Elde edilen yeni değerler master makina tarafından ilgili slave makinalara gönderilir. Bu işlemler yakınsama kriteri sağlanıncaya kadar yapılır.

Sıralı bölge ayrıklaştırmasında arayüzeylere sadece iki bölge, '+' şeklindeki bölge ayrıklaştırmasın da ise iki bölge ve bir kenara temas eden dört bölge bulunmaktadır. Alt bölge sayısının artması ile arayüzeylerde birbirine temas eden bölge sayıları artmıştır.




Şekil 3.7. Arayüzey Temas Bölgeleri

Şekil 3.7'de **A** bölgesi olarak gösterilen yerler arayüzeyde 7 bölgeye de ortak olan noktaları temsil etmektedir. Buradaki iki nokta, farklı 7 bölgeyi göstermektedir. **B** bölgesi ise üç adet olup 3 bölgeye de ortak olan kenarlardır. Geriye kalan **C** bölgesi de komşu 1 bölgeyi göstermektedir. Arayüzeydeki haberleşmeler için mevcut olan programa, ortak kenarlar ve noktalar için iki adet alt program eklenmiştir. Alt programlar ekler kısmında bulunmaktadır.




Şekil 3.8. Bir Bölgenin Temas Eden Yüzeyleri

Şekil 3.8'de bir bölgenin dört farklı bölgeyle teması vardır. En öndeki ve alttaki yüzeyde, yüzey şartları diğer yüzeyler de ise gerekli olan düğüm numaraları verilmektedir. Birbirine temas eden tüm bu yüzeylerde düğüm numaraları sırasıyla ve kaçar adet olduğunu bildirecek şekilde hazırlanmış ve ilgili slave makinalara bildirilmiştir. Şekil 3.8'de iç tarafta yer alan bölgenin komşuları gösterilmiştir. Dış taraftaki bölgede ise üç adet komşu bölge bulunmaktadır.

Tablo 3.2 bir kenara dört bölgenin de ortak olduğu durumlarda, veri alışverişini gösteren tablodur. Dört bölgeden biri alan bölge geriye kalan üçü ise gönderen bölgelerdir. Gönderen bölgeler, temas ettikleri kenardaki verileri alan bölgeye gönderirler. Düzeltilen veriler aynı bölgelere alan bölge tarafından geri gönderilir.

Tablo 3.2. Dört Bölgeye Ortak Kenar Alışverişleri

| Adet | Gönderen Bölge | **Alan Bölge** | Gönderen Bölge | Gönderen Bölge |
|---|---|---|---|---|
| 1 | 1 | **2** | 5 | 6 |
| 2 | 5 | **6** | 9 | 10 |
| 3 | 9 | **10** | 13 | 14 |
| 4 | 13 | **14** | 15 | 16 |
| 5 | 1 | **2** | 3 | 4 |
| 6 | 2 | **3** | 6 | 7 |
| 7 | 6 | **7** | 10 | 11 |
| 8 | 10 | **11** | 14 | 15 |
| 9 | 1 | **4** | 5 | 8 |
| 10 | 5 | **8** | 9 | 12 |
| 11 | 9 | **12** | 13 | 16 |
| 12 | 3 | **4** | 7 | 8 |
| 13 | 7 | **8** | 11 | 12 |
| 14 | 11 | **12** | 15 | 16 |

Tablo 3.3'de de sekiz bölgeye ortak olan tek bir nokta için yapılan veri alışveriş durumu bulunmaktadır. Ortak kenarlar için yapılan işlemler tek bir nokta içinde yapılmaktadır.

Tablo 3.3. Sekiz Bölgeye Ortak Nokta Alışverişleri

| Adet | Gön. Bölge | **Alan Bölge** | Gön. Bölge | Gön. Bölge | Gön. Bölge | Gön. Bölge | Gön. Bölge | Gön. Bölge |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | **2** | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 5 | **6** | 7 | 8 | 9 | 10 | 11 | 12 |
| 3 | 9 | **10** | 11 | 12 | 13 | 14 | 15 | 16 |

Tablo 3.4'de gösterilen değerler, master makinaya gönderilen verilerin görev numaralarına göre düzenlenmiş halidir. Master, karışık halde slave makinalardan aldığı verileri yine slave makinalardan gönderilen kod aracılığıyla oluşturması gereken dizinde nereye koyacağına karar verir ve bunları sıralayarak bir dizin haline getirir. Master makina kendisine gelen verileri yine aynı kod ile slave makinalara gönderir. Tablo 3.4'de ki son iki sütunda yer alan sayılar da arayüzeylerde bulunan nokta sayılarıdır. Sayıların farklı oluşu arayüzeylerin farklı eksenler boyunca birbirlerine temas etmeleridir.

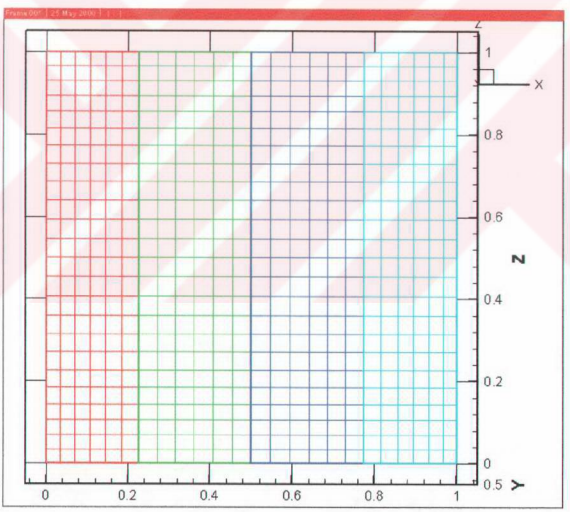
Tablo 3.4. Görev Numaraları

| GÖNDERENİN GÖREV NUMARASI | ALANIN GÖREV NUMARASI | GÖNDERENİN GÖREV NUMARASI | ALANIN GÖREV NUMARASI | NOKTA SAYISI | NOKTA SAYISI |
|---|---|---|---|---|---|
| 1 | 2 | 2 | 1 | 25 | 25 |
| 1 | 4 | 4 | 1 | 50 | 50 |
| 2 | 6 | 6 | 2 | 50 | 50 |
| 3 | 2 | 2 | 3 | 50 | 50 |
| 3 | 7 | 7 | 3 | 50 | 50 |
| 4 | 3 | 3 | 4 | 25 | 25 |
| 5 | 1 | 1 | 5 | 50 | 50 |
| 5 | 6 | 6 | 5 | 20 | 20 |
| 5 | 8 | 8 | 5 | 40 | 40 |
| 6 | 7 | 7 | 6 | 40 | 40 |
| 6 | 10 | 10 | 6 | 50 | 50 |
| 7 | 11 | 11 | 7 | 50 | 50 |
| 8 | 4 | 4 | 8 | 50 | 50 |
| 8 | 7 | 7 | 8 | 20 | 20 |
| 9 | 5 | 5 | 9 | 50 | 50 |
| 9 | 10 | 10 | 9 | 20 | 20 |
| 9 | 12 | 12 | 9 | 40 | 40 |
| 10 | 11 | 11 | 10 | 40 | 40 |
| 10 | 14 | 14 | 10 | 50 | 50 |
| 11 | 15 | 15 | 11 | 50 | 50 |
| 12 | 8 | 8 | 12 | 50 | 50 |
| 12 | 11 | 11 | 12 | 20 | 20 |
| 13 | 9 | 9 | 13 | 50 | 50 |
| 13 | 16 | 16 | 13 | 50 | 50 |
| 14 | 13 | 13 | 14 | 25 | 25 |
| 15 | 14 | 14 | 15 | 50 | 50 |
| 15 | 16 | 16 | 15 | 25 | 25 |
| 16 | 12 | 12 | 16 | 50 | 50 |
| | | | Toplam : | **1140** | **1140** |

# 4. SONUÇLAR

## 4.1 Tez Çalışmasına Başlandığı Andaki Durum

Navier-Stokes denklemlerinin sonlu elemanlarla çözümü için bölgelere ayırma yöntemi kullanılarak kübik kavite problemi çözülmüştür. Bu problem bir test uygulaması olup 25x13x25 lik (simetrik) bir hesap ağıyla ayrıklaştırılan küp kapağının çekilmesi problemidir. Akış Reynolds sayısı kapak boyu ve kapak hızına bağlı olarak 1000 alınmıştır. Çözüm, akış durağan hale gelene kadar 600 zaman adımı atılarak yapılmış olup bu sürede 30 boyutsuz zaman geçmiştir.




Şekil 4.1. Bir Yönde Bölge Ayrıklaştırması (Simetri Düzlemi)

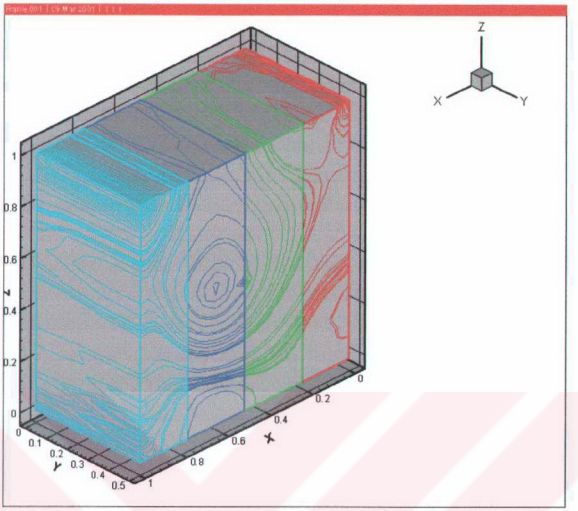
Şekil 4.1'de görüldüğü üzeri 4 eşit bölgeye ayrılan geometrimizde üç adet ara yüzey bulunmaktadır. Her ara yüzey 325 (25 x 13) adet nokta ile temsil edilmektedir. Ara yüzeylerde bulunan nokta sayısının fazla oluşu haberleşme süresini uzatacağından bölge ayrıklaştırmasının, ara yüzeylerde en az nokta olacak şekilde yapılması gerekmektedir.

Bu uygulamada Poisson denkleminin Neuman sınır şartı ile çözümünde verilmesi gereken sabit değer bir bölgede verilmiş ve diğer bölgelere uygun olan sınır şartı değerleri hesaplanarak problem çözülmüştür. Tablo 4.1'de 600 adım dt= 0.05 ve basınç için çözüm toleransı $10^{-5}$, paralel çözüm toleransı ise $10^{-3}$ olarak elde edilen çözümün zamanları, şekil 4.2' de eş basınç yüzey eğrileri ve şekil 4.3' de simetri düzlemindeki eş basınç eğrileri yer almaktadır.

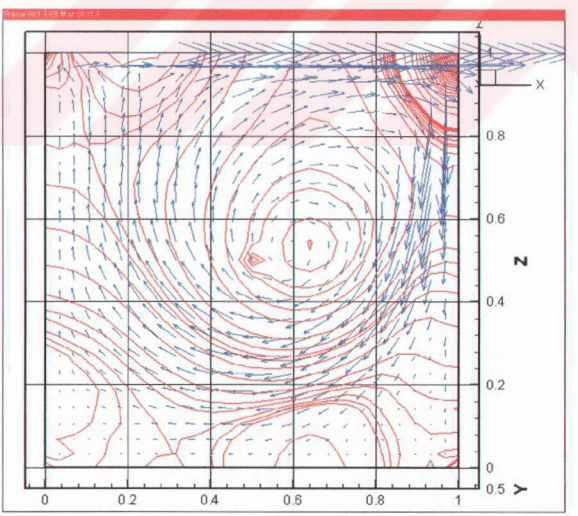
Tablo 4.1. Bir Yönde Bölmeli Çözüm Zamanları , Sn

| **Bölge Sayısı** | **4 x (25x13x7)** |
|---|---|
| Geçen Zaman, Sn | 2843 |
| CPU Zamanı (Usta) , Sn | 1562 |
| CPU Zamanı (Çırak1) , Sn | 1338 |
| CPU Zamanı (Çırak2) , Sn | 1342 |
| CPU Zamanı (Çırak3) , Sn | 1219 |
| CPU Zamanı (Çırak4) , Sn | 1103 |

Yukarıda tabloda CPU zamanı, bilgisayarın işlem yaptığı zamanı gösterir. Geçen zaman ile CPU zamanı arasındaki fark ise bilgisayarların haberleşmeye ayırdıkları ve boşta bekledikleri zamanı göstermektedir.




Şekil 4.2. Bir Yönde Bölmeli, Eş Basınç Eğrileri




Şekil 4.3. Bir Yönde Bölmeli, Simetri Düzleminde Eş Basınç Eğrileri

### 4.2 Kübik kavite'nin '+' Şeklinde Bölünmesi

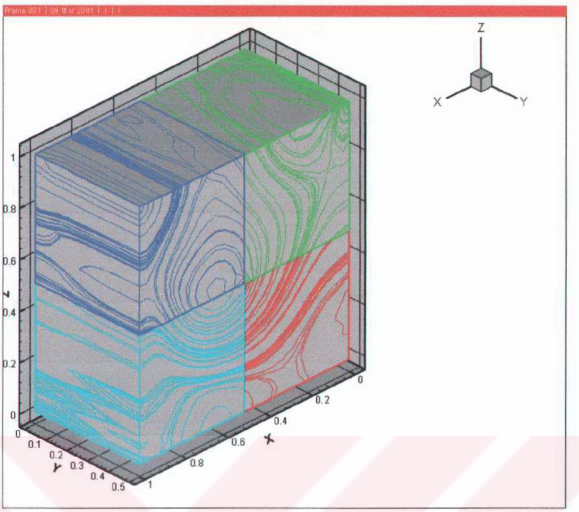
Belirli bir yöne bağlı olarak bölme işlemine karşı daha genel bir bölme ile paralel çözümleme yapılmasına örnek oluşturmak üzere kübik kavite bu defa '+' şeklinde 4 bölgeye ayrılmıştır.

Simetrik olarak yapılan çözümlerde, '+' bölme ile oluşturulan her bölge (13x13x13)'lük bir hesap ağından oluşmuş iken bir yönde sıralı bölmede ise her bölge (25x13x7)'lik bir ağdan oluşmuştur. Bölge ağların birbirinden farklı olması haberleşme için geçen sürenin değişmesine sebep olmuştur. '+' şeklinde keyfi bölmede ara yüzeyde bulunan nokta sayımız 169 iken sıralı bölmede 325 nokta bulunmaktadır.

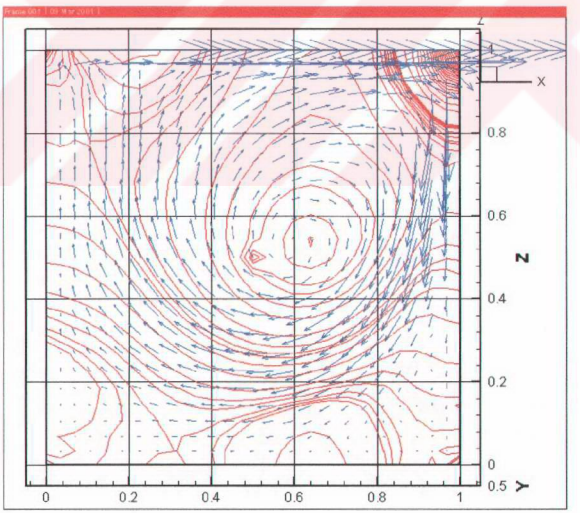
Tablo 4.2. '+' Bölmeli Çözüm Zamanları, Sn

| **Bölge Sayısı** | **4 x (13x13x13)** |
|---|---|
| Geçen Zaman, Sn | 2133 |
| CPU Zamanı (Usta) , Sn | 1211 |
| CPU Zamanı (Çırak1) , Sn | 1957 |
| CPU Zamanı (Çırak2) , Sn | 1972 |
| CPU Zamanı (Çırak3) , Sn | 1756 |
| CPU Zamanı (Çırak4) , Sn | 1967 |

Çözümler zamanda 600 adım giderek yapılmıştır. İterasyon toleransları, bölgelere ayırmada $1x10^{-3}$ ve basınç için yakınsamalarda ise $1x10^{-5}$ olarak alınmıştır.




Şekil 4.4. '+' Bölmeli, Eş Basınç Eğrileri




Şekil 4.5. '+' Bölmeli, Simetri Düzleminde Eş Basınç Eğrileri

Şekil 4.4 ve 4.5'de '+' bölmeli pQ2Q1 elemanları ile elde edilen basınç eğrileri verilmiş olup, eş basınç eğrilerinin tek yönde bölme ile elde edilen basınç eğrileri ile aynı olduğu gözlenmektedir.

### 4.3 Keyfi (N) Adet Bölge Ayrıklaştırması Sonuçları

Keyfi olarak ve işlemci sayımıza bağlı kalmadan yaptığımız bölge ayrıklaştırmasında ki farklardan birisi yüzey sayımızın bölge sayımızdan fazla oluşudur. Diğer uygulamalarda, sıralı bölme için 4 bölge mevcut iken 3 adet yüzeyimiz, '+' şeklindeki bölmede ise 4 adet yüzeyimiz vardı. Bu uygulamada ise 16 bölge var iken yüzey sayısı 28 adete çıkmıştır. Her bir bölgede Şekil 3.6'da görüldüğü üzere ikiden fazla birbiriyle temas eden yüzey bulunmaktadır. Bu da mevcut programda işlemci sayısı kadar yapılan işlemlerin temas eden yüzey sayısı kadar yapılması gerektiği ortaya çıkmış ve gerekli düzeltmeler yapılmıştır.

16'lı bölmeli sonuçlar hesaplama sisteminde oluşan bir arıza nedeni ile gerçekleştirilememiştir.

# 5. DEĞERLENDİRME VE ÖNERİLER

Üç boyutlu sıkıştırılamaz akışların paralel hesaplamasına yönelik olarak daha önce geliştirilen bir yazılım karmaşık ve büyük ölçekli problemleri çözümleyebilmek üzere genelleştirilmiştir. Bu yazılım parçalı adımlar zaman ayrıklaştırması ve galerkin sonlu elemanlar uzaysal ayrıklaştırması ile Navier-Stokes denklemlerini çözmektedir. Hız düzeltici potansiyele ilişkin Poisson denkleminin etkin paralel hesabı için bölge ayrıklaştırması (domain decomposition) yöntemi kullanılmıştır. Momentum denklemleri tüm bir bölgenin parçaları olacak şekilde açık bir şema ile paralel olarak çözülmektedir. PVM kütüphanesi yardımı ile farklı işlemciler arası haberleşme sağlanmaktadır. Çalışma başlangıcında ele alınan programda paralel hesaplama amacı ile çözümleme bölgesinin parçalara ayrılması yalnızca bir yönde sıralı olarak yapılmakta ve işlemci sayısı kadar alt bölge oluşturulabilmekteyken, programın geliştirilen yeni hali ile herhangi bir şekilde alt bölgeler oluşturulabilmekte ve bir işlemcide birden fazla işlem çalıştırılabilmektedir. Dolayısı ile daha büyük boyutlu ve karmaşık geometrili problemlerin çözülebilmesi gündeme gelmiştir. Uygulama olarak kübik kavite problemi seçilmiş ve Re=1000 için 4 bölgeye ilişkin sonuçlar elde edilmiş, 16 bölgeye ilişkin gerekli veriler tamamı ile hazırlanmıştır.

Bundan sonraki uygulamalarda, keyfi bölme ve birden fazla işlemi bir işlemcide yaptırarak, iş istasyonu grubunda istenilen sayıda alt bölgeye bölerek yapılacak uygulamalar için gerekli olan verilerin oluşturulmasını sağlayan otomatik bir programın hazırlanması ve/veya MeTis gibi hazır programlardan yararlanılması verilerin oluşturulma sürecini kısaltacağından çok yararlı olacaktır. Ayrıca, programda halihazırda kullanılan Master/Slave uygulamasında iş yükü dağılımını optimize etmek ve işlemciler arası haberleşmeye ve çalışmadan beklemeye giden zamanı kısaltmak açısından işlerin olabildiğince slave'ler tarafından yapılması, Master tarafından olabildiğince işlem yapılmaması yönünde bir düzenlemenin programa eklenmesi de faydalı olacaktır.

## KAYNAKLAR


**[1] Gülçat, Ü.**, and **Aslan, A.R.**, 1997. Aerodynamic Flow Calculations with an Accelerated FEM Solver, Chapter 8 in Flows at Large Reynolds Numbers, edited by H. Schmitt, *Computational Mechanics publications, UK.*

**[2] Gülçat, Ü.** and **Aslan, A.R.**, 1997. Accurate 3-D Viscous Incompressible Flow Calculations with the FEM. *International Journal for Numerical Methods in Fluids*, vol.**25**, pp.985-1001.

**[3] Aslan, A.R.**, **Edis F.O.**, **Gülçat, Ü.**, and **Mısırlıoğlu, A.**,1998. Accurate Incompressible N-S Solution on Cluster of Work Stations, *Parallel CFD , Taiwan.*

**[4] Aslan, A.R.**, **Gülçat, Ü.**, and **Edis, F.O.**,1998. Accurate Solution of Navier-Stokes Equations with Parallel Computations, *Fourth European Fluid Dynamics Conference (ECCOMAS 98),* Athens, Greece, 7-11 September, pp.484-489.

**[5] Aslan, A.R.**, **Gülçat, Ü.**, **Mısırlıoğlu, A.** and **Edis, F.O.**, 1999. Domain Decomposition Implementations for Parallel Solution of 3D Navier-Stokes Equations, *Parallel CFD'99,* Williamsburg, Virginia, USA, May 23-26.

**[6] Aslan, A.R.**, **Edis, F.O.**, **Gülçat, Ü.** and **Mısırlıoğlu, A.**, 1999. Fast and Accurate Parallel Navier-Stokes Solutions with PQ2Q1 Finite Elements, *Parallel CFD WORKSHOP*, İstanbul, Turkey, June 16-18.

**[7] Edis, F.O.**,and **Aslan, A.R.**, 1998.Efficient Incompressible Flow Calculations Using PQ2Q1 Element, *Com. in Numerical Methods in Engineering*, Vol.14, pp.161-178.

**[8] Edis, F.O.**, and **Aslan, A.R.**, 1997. Efficient Incompressible Flow Calculations Using Pseudo-Second Order Finite Elements, In Numerical Methods in Laminar and *Turbulent Flow,* vol.**10**, Editors C. Taylor and J.T. Cross, pp 61-72, 21-25 July, Swansea, UK.

**[9]** PVM Handbook, ftp: netlib2.cs.utk.edu.

[10] **Usta, E.**, 1997. Paralel Sanal Makina Ortamında Maxwell Denklemlerinin Paralel Hesabı'', *Yüksek Lisans Tezi,* İ.T.Ü Fen Bilimleri Enstitüsü, İstanbul.

## EK: BİLGİSAYAR PROGRAMI

Bölgeler arasında ortak olan kenarların ve noktaların katkılarını birbirine gönderen bilgisayar programı aşağıda yer almaktadır

```
Subroutine exchscal4(noar,loar,ar11,
ar22,ar33,nsn4sndb4,nrn4,rcvb4,tsnd4,
trcv4,iencd,
temp11,temp12,trcvv,temp21,temp22,
temp31,temp32,isend,ircv)
implicit double precision(a-h,o-z)
include 'fpvm3.h'
integer sndb4(nsn4), rcvb4(nrn4),
tsnd4, trcv4, trcvv(3)
dimension
ar11(loar),ar22(loar),ar33(loar)
dimension temp11(nsn4)
dimension temp12(nsn4)
dimension temp21(nsn4)
dimension temp22(nsn4)
dimension temp31(nsn4)
dimension temp32(nsn4)
print*,(trcvv(k),k=1,3)
c
c --- send SNDB to TSND
if(isend.eq.1)then !alan uc bolgeden
birisi ise, I, III ve IV

do i=1, nsn4
temp11(i) = ar11(sndb4(i))
enddo

if(noar.gt.1) then
do i=1,nsn4
temp21(i) = ar22(sndb4(i))
end do
end if

if(noar.gt.2) then
do i=1,nsn4
temp31(i) = ar33(sndb4(i))
end do
end if

call pvmfinitsend(iencd,ierr)
call pvmfpack(real8, temp11, nsn4, 1,
ierr)
if(noar.gt.1)
call pvmfpack(real8, temp21, nsn4, 1,
ierr)
if(noar.gt.2)
call pvmfpack(real8, temp31, nsn4, 1,
ierr)
call pvmfsend(tsnd4, 3, ierr)
endif
c --- receive RCVB from TRCV

if(ircv.eq.1)then
do i=1,nrn4
temp11(i) = ar11(rcvb4(i))
end do

if(noar.gt.1) then
do i=1,nrn4
temp21(i) = ar22(rcvb4(i))
end do
end if

if(noar.gt.2) then
do i=1,nrn4
temp31(i) = ar33(rcvb4(i))
end do
end if

do k=1,3
call pvmfrecv(trcvv(k), 3, ierr)
call pvmfunpack(real8, temp12, nrn4,
1, ierr)
if(noar.gt.1)call pvmfunpack(real8,
temp22, nrn4, 1, ierr)
if(noar.gt.2)call pvmfunpack(real8,
temp32, nrn4, 1, ierr)
c --- add contribs to local array
```

```
do i=1, nrn4
temp11(i) = temp11(i) + temp12(i)
enddo

if(noar.gt.1) then
do i=1,nrn4
temp21(i) = temp21(i) + temp22(i)
enddo
endif

if(noar.gt.2) then
do i=1,nrn4
temp31(i) = temp31(i) + temp32(i)
end do
endif
enddo

do i=1, nrn4
ii = rcvb4(i)
ar11(ii)  = temp11(i)
enddo

if(noar.gt.1) then
do i=1,nrn4
ii = rcvb4(i)
ar22(ii) = temp21(i)
end do
endif

if(noar.gt.2) then
do i=1,nrn4
ii = rcvb4(i)
ar33(ii) = temp31(i)
end do
endif

c
c --- send results to RCVB at TRCV
c
call pvmfinitsend(iencd,ierr)
do k=1,3
call pvmfpack(real8, temp11, nrn4, 1,
ierr)
if(noar.gt.1)
call pvmfpack(real8, temp21, nrn4, 1,
ierr)
if(noar.gt.2)
call pvmfpack(real8, temp31, nrn4, 1,
ierr)
call pvmfsend(trcvv(k), 3, ierr)
enddo
endif

c --- receive contributed arrays from
SNDB at TSND and override local
array
c
if(isend.eq.1)then
call pvmfrecv(tsnd4, 3, ierr)
call pvmfunpack(real8, temp11, nsn4,
1, ierr)
if(noar.gt.1)call pvmfunpack(real8,
temp21, nsn4, 1, ierr)
if(noar.gt.2)call pvmfunpack(real8,
temp31, nsn4, 1, ierr)
do i=1, nsn4
ii = sndb4(i)
ar11(ii)  = temp11(i)
enddo

if(noar.gt.1) then
do i=1,nsn4
ii = sndb4(i)
ar22(ii) = temp21(i)
end do
endif

if(noar.gt.2) then
do i=1,nsn4
ii = sndb4(i)
ar33(ii) = temp31(i)
end do
endif
endif
c
return
end

subroutine
exchvect4(noar,loar,ar11,ar22,ar33,ns
n4,sndb4,
*
nrn4,rcvb4,tsnd4,trcv4,iencd,temp11v,
temp12v,trcvv,
*                temp21v,
temp22v,temp31v,temp32v,isend,ircv)
implicit double precision(a-h,o-z)
include 'fpvm3.h'
integer sndb4(nsn4), rcvb4(nrn4),
tsnd4,trcv4, trcvv(3)
```

```
dimension
ar11(3,loar),ar22(3,loar),ar33(3,loar)
dimension temp11v(3,nsn4)
dimension temp12v(3,nsn4)
dimension temp21v(3,nsn4)
dimension temp22v(3,nsn4)
dimension temp31v(3,nsn4)
dimension temp32v(3,nsn4)

c --- send SNDB to TSND

if(isend.eq.1)then !alan uc bolgeden
birisi ise, I, III ve IV
do i=1, nsn4
temp11v(1,i) = ar11(1,sndb4(i))  !!!
temp11v(2,i) = ar11(2,sndb4(i))  !!!
temp11v(3,i) = ar11(3,sndb4(i))  !!!
enddo

if(noar.gt.1) then
do i=1,nsn4
temp21v(1,i) = ar22(1,sndb4(i))
temp21v(2,i) = ar22(2,sndb4(i))
temp21v(3,i) = ar22(3,sndb4(i))
end do
end if

if(noar.gt.2) then
do i=1,nsn4
temp31v(1,i) = ar33(1,sndb4(i))
temp31v(2,i) = ar33(2,sndb4(i))
temp31v(3,i) = ar33(3,sndb4(i))
end do
end if

call pvmfinitsend(iencd,ierr)
call pvmfpack(real8, temp11v,3*nsn4,
1, ierr)
if(noar.gt.1)
call pvmfpack(real8, temp21v,3*nsn4,
1, ierr)
if(noar.gt.2)
call pvmfpack(real8, temp31v,3*nsn4,
1, ierr)
call pvmfsend(tsnd4, 3, ierr)
endif

c --- receive RCVB from TRCV

if(ircv.eq.1)then
do i=1,nrn4
temp11v(1,i) = ar11(1,sndb4(i))
temp11v(2,i) = ar11(2,sndb4(i))
temp11v(3,i) = ar11(3,sndb4(i))
end do

if(noar.gt.1) then
do i=1,nrn4
temp21v(1,i) = ar22(1,sndb4(i))
temp21v(2,i) = ar22(2,sndb4(i))
temp21v(3,i) = ar22(3,sndb4(i))
end do
end if

if(noar.gt.2) then
do i=1,nrn4
temp31v(1,i) = ar33(1,sndb4(i))
temp31v(2,i) = ar33(2,sndb4(i))
temp31v(3,i) = ar33(3,sndb4(i))
end do
end if

do k=1,3
call pvmfrecv(trcvv(k), 3, ierr)
call pvmfunpack(real8, temp12v,
3*nrn4, 1, ierr)
if(noar.gt.1)
call pvmfunpack(real8, temp22v,
3*nrn4, 1, ierr)
if(noar.gt.2)
call pvmfunpack(real8, temp32v,
3*nrn4, 1, ierr)
c --- add contribs to local array
do i=1, nrn4
temp11v(1,i) = temp11v(1,i) +
temp12v(1,i)
temp11v(2,i) = temp11v(2,i) +
temp12v(2,i)
temp11v(3,i) = temp11v(3,i) +
temp12v(3,i)
enddo

if(noar.gt.1) then
do i=1,nrn4
temp21v(1,i) = temp21v(1,i) +
temp22v(1,i)
temp21v(2,i) = temp21v(2,i) +
temp22v(2,i)
temp21v(3,i) = temp21v(3,i) +
temp22v(3,i)
```

```
enddo
endif

if(noar.gt.2) then
do i=1,nrn4
temp31v(1,i) = temp31v(1,i) +
temp32v(1,i)
temp31v(2,i) = temp31v(2,i) +
temp32v(2,i)
temp31v(3,i) = temp31v(3,i) +
temp32v(3,i)
end do
endif
enddo

do i=1, nrn4
ii = rcvb4(i)
ar11(1,ii) = temp11v(1,i)
ar11(2,ii) = temp11v(2,i)
ar11(3,ii) = temp11v(3,i)
enddo

if(noar.gt.1) then
do i=1,nrn4
ii = rcvb4(i)
ar22(1,ii) = temp21v(1,i)
ar22(2,ii) = temp21v(2,i)
ar22(3,ii) = temp21v(3,i)
end do
endif

if(noar.gt.2) then
do i=1,nrn4
ii = rcvb4(i)
ar33(1,ii) = temp31v(1,i)
ar33(2,ii) = temp31v(2,i)
ar33(3,ii) = temp31v(3,i)
end do
endif

c --- send results to RCVB at TRCV
call pvmfinitsend(iencd,ierr)
do k=1,3
call pvmfpack(real8, temp11v, 3*nrn4,
1, ierr)
if(noar.gt.1)call pvmfpack(real8,
temp21v, 3*nrn4, 1, ierr)
if(noar.gt.2)call pvmfpack(real8,
temp31v, 3*nrn4, 1, ierr)
call pvmfsend(trcvv(k), 3, ierr)
enddo
endif
c
c --- receive contributed arrays from
SNDB at TSND and override local
array
c
if(isend.eq.1)then
call pvmfrecv(tsnd4, 3, ierr)
call pvmfunpack(real8, temp11v,
3*nsn4, 1, ierr)
if(noar.gt.1)call pvmfunpack(real8,
temp21v, 3*nsn4, 1, ierr)
if(noar.gt.2)call pvmfunpack(real8,
temp31v, 3*nsn4, 1, ierr)
do i=1, nsn4
ii = sndb4(i)
ar11(1,ii)  = temp11v(1,i)
ar11(2,ii)  = temp11v(2,i)
ar11(3,ii)  = temp11v(3,i)
enddo

if(noar.gt.1) then
do i=1,nsn4
ii = sndb4(i)
ar22(1,ii) = temp21v(1,i)
ar22(2,ii) = temp21v(2,i)
ar22(3,ii) = temp21v(3,i)
end do
endif

if(noar.gt.2) then
do i=1,nsn4
ii = sndb4(i)
ar33(1,ii) = temp31v(1,i)
ar33(2,ii) = temp31v(2,i)
ar33(3,ii) = temp31v(3,i)
end do
endif
endif
c
return
end
c
subroutine
exchscal16(noar,loar,ar111,ar222,ar33
3,nsn8,sndb8,nrn8,rcvb8,tsnd8,trcv8,ie
ncd,temp111,temp112,trcvv, temp221,
temp222,temp331,temp332,isend8,
ircv8)
```

```
implicit double precision(a-h,o-z)
include 'fpvm3.h'
integer sndb8(nsn8), rcvb8(nrn8),
tsnd8, trcv8, trcvv(7)
dimension
ar111(loar),ar222(loar),ar333(loar)
dimension temp111(nsn8)
dimension temp112(nsn8)
dimension temp221(nsn8)
dimension temp222(nsn8)
dimension temp331(nsn8)
dimension temp332(nsn8)
c
c --- send SNDB to TSND
if(isend8.eq.1)then !alan yedi
bolgeden birisi ise,
c       print*,' isend ilk gir',nsn4
do i=1, nsn8
temp111(i) = ar111(sndb8(i))  !!!
enddo

if(noar.gt.1) then
do i=1,nsn8
temp221(i) = ar222(sndb8(i))
end do
end if

if(noar.gt.2) then
do i=1,nsn8
temp331(i) = ar333(sndb8(i))
end do
end if

call pvmfinitsend(iencd,ierr)
call pvmfpack(real8, temp111, nsn8,
1, ierr)
if(noar.gt.1)call pvmfpack(real8,
temp221, nsn8, 1, ierr)
if(noar.gt.2)call pvmfpack(real8,
temp331, nsn8, 1, ierr)
call pvmfsend(tsnd8, 7, ierr)
!trcv8=yedisinden de alan, ornegin II
endif
c --- receive RCVB from TRCV
if(ircv8.eq.1)then
do i=1,nrn8
temp111(i) = ar111(rcvb8(i))
end do

if(noar.gt.1) then
do i=1,nrn8
temp221(i) = ar222(rcvb8(i))
end do
end if

if(noar.gt.2) then
do i=1,nrn8
temp331(i) = ar333(rcvb8(i))
end do
end if

do k=1,7
call pvmfrecv(trcvv(k), 7, ierr)
call pvmfunpack(real8, temp112, nrn8,
1, ierr)
if(noar.gt.1)call pvmfunpack(real8,
temp222, nrn8, 1, ierr)
if(noar.gt.2)call pvmfunpack(real8,
temp332, nrn8, 1, ierr)
c --- add contribs to local array
do i=1, nrn8
temp111(i) = temp111(i) + temp112(i)
enddo

if(noar.gt.1) then
do i=1,nrn8
temp221(i) = temp221(i) + temp222(i)
enddo
endif

if(noar.gt.2) then
do i=1,nrn8
temp331(i) = temp331(i) + temp332(i)
end do
endif
enddo
end
```

# ÖZGEÇMİŞ

05.04.1975 yılında Uşak'ta doğdu. İlk, orta ve lise öğrenimini İzmir'de bitirdi. 1993 yılında İ.T.Ü Uçak ve Uzay Bilimleri Fakültesi Uzay Mühendisliği bölümünde üniversite öğrenimine başladı. 1997 yılında Uzay Mühendisi olarak mezun oldu. Aynı yıl İ.T.Ü Fen Bilimleri Enstitüsü Uzay Mühendisliği anabilim dalında yüksek lisans öğrenimine başladı. Halen Açıkhava Reklamcılık firmasında çalışmaktadır.