KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

# KESİK ULV AYRIŞIMININ BLOK GÜNCELLEMESİ

Ebru AYDOĞAN

HAZİRAN 2013

Bilgisayar Mühendisliği Anabilim Dalında Ebru AYDOĞAN tarafından hazırlanan KESİK ULV AYRIŞIMININ BLOK GÜNCELLEMESİ adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof. Dr. Hasan ERBAY
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Prof. Dr. Hasan ERBAY
Danışman

Jüri Üyeleri

Başkan(Danışman) : Prof. Dr. Hasan ERBAY _____

Üye : Prof. Dr. Ediz POLAT _____

Üye : Yrd. Doç. Dr. Gökçe NUR _____

…/…/2013

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Doç. Dr. Erdem Kamil YILDIRIM
Fen Bilimleri Enstitüsü
Müdürü

# ÖZET

KESİK ULV AYRIŞIMININ BLOK GÜNCELLEMESİ

AYDOĞAN, Ebru
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi
Danışman: Prof. Dr. Hasan ERBAY
Haziran 2013, 40 sayfa

Sayısal rankın hesaplanması ve temel uzayların yaklaşık olarak bulunması problemi pek çok alanda uygulama alanı bulmaktadır. Sinyal işleme, görüntü işleme, kontrol ve istatistik bu alanlar arasındadır. Sayısal rankın konu olduğu matris hesaplamalarında ortogonal dönüşümler önemli bir rol oynar. Çünkü ortogonal dönüşümler sayısal olarak kararlıdır, *2-normu* korur ve bir matrisin sayısal rankı hakkında bilgi verir. Ayrıca, ortogonal dönüşümlerde mevcut veri matrisine satır eklenmesi veya çıkarılması işlemleri güvenilir bir biçimde gerçekleştirilebilir.

Bir matrisin ortogonal dönüşümünün bulunması için en sık kullanılan yöntem Tekil Değer Ayrışımı (SVD) algoritmasıdır. Alternatif diğer ortogonal dönüşümler ise QR, URV ve ULV ayrışımlarıdır.

Bu tez, ULV temel matris ayrışımı, kesik ULV ayrışımı ve kesik ULV ayrışımının blok güncelleme algoritması hakkında bilgi vermektedir. Önerilen algoritma, diğer algoritmalardan farklı olarak yeni eklenen verileri bloklar halinde güncellemektedir. Teorik ve sayısal sonuçlar, ULV ayrışımının SVD'ye iyi bir alternatif olduğunu göstermektedir.

**Anahtar Kelimeler:** ULV Ayrışımı, Kesik ULV Ayrışımı, Blok Güncelleme

## ABSTRACT

BLOCK UPDATE ON TRUNCATED ULV DECOMPOSITION

AYDOĞAN, Ebru
Kırıkkale University
Graduate School of Natural and Applied Sciences
Depertment of Computer Engineering, M.Sc. Thesis
Supervisor: Prof. Dr. Hasan ERBAY
June 2013, 40 pages

The problem of determining the numerical rank and approximating the subspaces has applications in many fields. Signal processing, image processing, control and statistics are among those fields. Orthogonal transformations play an important role in matrix calculations when the numerical rank is an issue. Because orthogonal transformations are numerically stable, preserve the *2-norm* and give information about the numerical rank of a matrix. Besides, data update and downdate operations of existing matrix are performed reliably in orthogonal transformations.

The most common method for finding the orthogonal transformation of a matrix is Singular Value Decomposition (SVD) algorithm. Other alternative orthogonal transformations are QR, URV and ULV decompositions.

This thesis, presents ULV-based matrix decomposition, the truncated ULV decomposition and block updating algorithm for it. Unlike the other algorithms, proposed algorithm updates the new added data as blocks. The theoretical and numerical results show that the truncated ULV decomposition can be a good alternative to the SVD.



**Keywords:** ULV Decomposition, Truncated ULV Decomposition, Block Update

# TEŞEKKÜR

Tezimin hazırlanmasının her aşamasında bilgi ve önerileri ile beni yönlendiren, yardımlarını esirgemeyen danışman hocam Sayın Prof. Dr. Hasan ERBAY'a, tezimi okuyup değerlendiren ve önerilerini sunan Sayın jüri üyeleri hocalarıma, tezin yazımı sırasında bana yardımcı olan ve yol gösteren bölüm arkadaşlarım Volkan ATEŞ, Yunus KÖKVER ve Almıla ÖZCAN'a teşekkürlerimi sunarım.

Varlıkları ve manevi destekleriyle her zaman yanımda olan aileme ayrıca teşekkürü bir borç bilirim. İyi ki varsınız.

# İÇİNDEKİLER DİZİNİ

Sayfa

# ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

# ÇİZELGELER DİZİNİ

ÇİZELGE Sayfa

# 1. GİRİŞ

Karmaşık bir matrisin, daha basit matrislerin çarpımına dönüştürülmesine matris ayrışımı denir ve matris ayrışımı 1960'lı yıllardan bu yana sayısal analizde çok sık kullanılan yöntemlerden biridir. Önceleri sadece lineer sistem analizine uygulanmış olan matris ayrışımı, son yıllarda yazılım, elektronik sinyal filtrelemesi, görüntü işleme gibi pek çok alanda kullanılmaktadır [1].

En çok kullanılan matris ayrışımlarından biri olan Tekil Değer ayrışımı (SVD), bir matrisi ortogonal bir matris, köşegen bir matris ve ortogonal bir matris olmak üzere üçlü bir çarpıma ayırır. SVD örneğinde olduğu gibi, matris ayrışımı sonucunda ortogonal matrislerin oluştuğu dönüşümlere ortogonal dönüşümler adı verilir [1].

Matris hesaplamalarında, ortogonal dönüşümler önemli bir rol oynar. Bunun sebeplerinden biri, ortogonal dönüşümlerin sayısal olarak kararlı olmasıdır ki bu durum sayısal rank konu olduğunda önemlidir. İkincisi, ortogonal dönüşümler *2-normu* korur ve bu durum da problemlerin basitleştirilmesinde kullanılabilir. Üçüncüsü, ortogonal dönüşümlerde mevcut veri matrislerine satır eklenmesi veya çıkarılması işlemleri güvenilir bir şekilde gerçekleştirilebilir. Son olarak, ortogonal dönüşümler, bir matrisin sayısal rankı ve o matris üzerinde tanımlanmış bazı alt uzaylar hakkında bilgi verebilir [2].

Bir matrisin ortogonal dönüşümünün bulunması için en sık kullanılan yöntem SVD algoritmasıdır [3]. SVD, güvenli bir biçimde sayısal rankı ortaya çıkarır ve gerekli alt uzay bilgilerini içerir. Dönüşüm için kullanılan SDV algoritması sayısal olarak kararlı bir algoritmadır [4].

SVD, çok kuvvetli bir ayrışım aracı olmasına rağmen, bazı dezavantajları da vardır. Örneğin, mevcut veri matrisine yeni veri eklendiğinde veya çıkarıldığında, ayrışımı yinelemenin maliyeti yüksektir. Yani SVD, veri akışının sürekli olduğu problemlerde uygun bir dönüşüm hesaplama yöntemi değildir. Bu yüzden, sayısal

rankı hesaplamada ve değer uzayını bulmada SVD gibi etkili fakat daha hızlı ayrışım algoritmalarına ihtiyaç duyulmaktadır.

SVD'ye alternatif olabilecek ayrışımlardan biri QR ayrışımıdır [5]. QR ayrışımı bir matrisi, ortogonal bir matris ve üst üçgensel bir matrisin çarpımına dönüştürür. Veri akışının devamlı olduğu durumlarda ortogonal dönüşümü yeniden hesaplamanın maliyeti daha düşüktür.

Sayısal sıralamalar ve istenen alt uzayları elde etmek için güvenilir hesaplamalar sağlayan ve rankı ortaya çıkaran diğer ortogonal dönüşümler ise URV ve ULV ayrışımlarıdır [2]. Bu ayrışımlar, SVD gibi rank bilgisini ve alt uzay bilgisini içerir. Veri akışının devamlı olduğu durumlarda ise hesaplama karmaşası SVD'ye göre düşüktür. Bu tezde, ULV tabanlı matris ayrışımı üzerinde durulmaktadır.

Tez boyunca veri matrislerinin sayısal rankını hesaplamada ULV ayrışımını kullanma sebepleri ortaya konmaya çalışılmaktadır. Bundan yola çıkarak ULV tabanlı matris ayrışımı tanımlanmakta ve ayrışıma ait teorik sonuçlar verilmektedir.

Bir veri matrisine yeni veri bloğu eklendiğinde, mevcut bilgiyi kullanarak kesik ULV ayrışımını güncelleyen bir algoritma oluşturmak tezin temel amacıdır. Önerilen algoritma, diğer algoritmalardan farklı olarak yeni gelen verileri birden fazla satırdan oluşan bloklar halinde güncellemektedir. Uygulanan bu yöntemin, veri akışının sürekli olduğu blok güncelleme işlemlerinde, rankı hesaplamada diğer yöntemlere iyi bir alternatif olacağını ortaya koymak amaçlanmaktadır.

Elde edilen kesik ULV ayrışımını blok güncelleme algoritmasının sinyal işleme, görüntü işleme, video işleme gibi alanlarda uygulanabileceği düşünülmektedir.

Günümüzde, çeşitli matris ayrışım yöntemleri ve uygulamaları alanında yapılan çalışmalar aşağıda kısaca özetlenmektedir.

ULV ve URV ayrışımları ilk olarak Faddeev, Kublanowskaya ve Faddeeva [6] ve Hanson ve Lawson [7] tarafından tartışılmıştır. Bir dönem unutulan bu ayrışımlar daha sonra yeniden gündeme gelmiştir.

G. W. Stewart [8], ULV ve URV ayrışımı algoritmalarını ortaya çıkararak araştırmacıların dikkatine sunmuştur. Yaptığı çalışmada URV ayrışımını kullanan Stewart, bir matrise yeni verilerin eklendiği güncelleme işlemini SVD'ye göre daha az maliyetle gerçekleştiren bir algoritma önermiştir. SVD ile $O(n^3)$ maliyetle gerçekleştirilen güncelleme işlemi, URV kullanılarak $O(n^2)$ maliyetle gerçekleştirilmektedir. Önerilen bu algoritma, yüksek ranklı matrisler için uygun bir algoritmadır.

R. D. Fierro ve P. C. Hansen [9], Stewart'ın algoritmasından farklı olarak, düşük ranklı matrisler için uygun bir ULV ayrışımı algoritması önermiştir. Yaptıkları çalışma ile önerdikleri algoritmanın düşük ranklı matrislerin güncellenmesi işleminde etkili bir yöntem olduğunu ortaya koymuşlardır.

H. Erbay [10], kesik ULV ayrışımını kullanarak bir matrisin temel uzaylarını ve rankını bulan bir algoritma önermiştir. Önerilen algoritma, düşük ranklı matrislerin güncellenmesini etkili bir biçimde gerçekleştirmektedir. Algoritma, güncellemedeki maliyeti bakımından SVD'ye alternatif bir öneridir.

J. L. Barlow ve H. Erbay [11], düşük ranklı matrisler için uygun bir başka kesik ULV ayrışımı güncelleme algoritması önermişlerdir. Önerilen bu algoritma, bir matrise satır eklendiğindeki güncelleme (update) ve bir matristen satır çıkarıldığındaki güncelleme (downdate) işlemlerini etkili bir biçimde gerçekleştirmektedir.

H. L. Andrews ve C. L. Patterson [12], SDV'nin görüntü işlemede kullanılması konusunu önermişlerdir. SVD algoritmalarının, görüntülerin temsilinde verimli olarak kullanılabileceğini savunmuşlardır. Yaptıkları çalışmada, SVD algoritmalarının potansiyel uygulamalarını görüntü işlemenin çeşitli alanlarında yorumlamışlardır.

W. Yodchanan [13] ise, UTV ayrışımlarının resim yapısı üzerindeki etkisini araştıran bir çalışma yapmıştır. SVD ve UTV ayrışımlarını görüntü düzeltmede karşılaştırmalı olarak kullanmıştır. Yaptığı çalışmanın sonucunda UTV ayrışımlarının görüntü işleme alanında SVD'ye alternatif olarak kullanılabileceğini göstermiştir.

Tezin ikinci bölümünde, ilk olarak çalışmanın temeli olan lineer cebir hakkında bilgi verilmiştir. Gerekli matris ve matris işlemleri tanımlanarak, bu matrisler üzerinde gerçekleştirilen ortogonal dönüşümlere değinilmiştir. Devamında, çalışmanın uygulanmasında kullanılan Matlab programı ve Matlab'de matris işlemleri anlatılmıştır. Son olarak ise, matris ayrışım çeşitlerinden bahsedilmiştir.

Tezin üçüncü bölümünde, kullanılan matris ayrışımı anlatılmıştır. Tezin temelini oluşturan algoritma ve elde edilen sayısal sonuçlar verilmiştir.

Tezin son bölümünde ise, tez ile ilgili elde edilen sonuçlara ve değerlendirmelere değinilmiştir.

# 2. MATERYAL VE METOD

## 2.1. Lineer Cebir

### 2.1.1. Vektörler ve Matrisler

**Tanım 1:** Bir reel $n$ – *vektör* $\mathbf{x}$, $i = 1, 2, \ldots, n$ için $x_i \in \mathbb{R}$ olmak üzere, $n$ tane reel sayının

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.1}$$

şeklinde sıralanmasıdır.

Skalar $x_i$ 'ler $\mathbf{x}$'in elemanları olarak adlandırılırlar. Tüm reel $n$ – *vektör*'lerin kümesi $\mathbb{R}^n$ olarak gösterilir. Tez boyunca vektörler kalın ve küçük harfle ifade edilecektir.

**Tanım 2:** Bir matris, sayıların $m$ satır ve $n$ sütundan oluşan dikdörtgensel gösterimidir. Yani

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{2.2}$$

$i = 1, 2, \ldots, m$ ve $j = 1, 2, \ldots..., n$ için $a_{ij} \in \mathbb{R}$ şeklindedir. Skalar $a_{ij}$ , $\mathbf{A}$'nın bir elemanıdır. $m \times n$ reel matrisinin tüm kümeleri $\mathbb{R}^{mxn}$ şeklinde gösterilir. Çalışma boyunca aksi söylenmedikçe matrislerin tamamının reel sayılardan oluştuğu varsayılacaktır.

Verilen bir $m \times n$ boyutlu $\mathbf{A}$ matrisinin $i$ – *inci* satırı $i = 1, 2, \ldots, m$ için

$$\begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix} \tag{2.3}$$

ve $\mathbf{A}$'nın $j$ – *inci* sütunu $j = 1, 2, \ldots, n$ için

$$\begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix} \tag{2.4}$$

dir.

**Tanım 3:** $m \times n$ boyutlu bir **A** matrisi için $m = n$ ise kare matristir. Bu durumda **A** matrisi *n – inci* derecedendir.

**Tanım 4:** Tüm elemanları sıfır olan vektöre sıfır vektörü ve tüm elemanları sıfır olan matrise sıfır matrisi denir. Bu matris veya vektörler boyutundan bağımsız olarak **0** ile gösterilir.

**Tanım 5:** *n – inci* dereceden kare bir **I** matrisi

$$\mathbf{I}_{ij} = \begin{cases} 1, \ i = j \\ 0, \ i \neq j \end{cases}$$

olarak tanımlanıyorsa birim matris adını alır.

**Tanım 6:** $\mathbf{A} = \left[ a_{ij} \right]$**,** $m \times n$ boyutlu bir matris ise **A** matrisinin transpozu, $a_{ij}^T = a_{ji}$ olmak üzere $\mathbf{A^T} = \left[ a_{ij}^T \right]$ şeklinde ifade edilen $n \times m$ boyutlu bir matristir. **A**'nın transpozu, satır ve sütunları yer değiştirilerek elde edilmiştir.

**Tanım 7:** *n – inci* dereceden bir $\mathbf{A} = \left[ a_{ij} \right]$ matrisi $i > j$ için $a_{ij} = 0$ ise üst üçgensel matristir. Yani **A** matrisi,

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{nn} \end{bmatrix}$$

şeklindedir.

**Tanım 8:** $n$ – *inci* dereceden bir $\mathbf{A}=\left[a_{ij}\right]$ matrisi $i<j$ için $a_{ij}=0$ ise alt üçgensel matristir. Yani **A** matrisi,

$$\begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

şeklindedir.

**Tanım 9:** $n$ – *inci* dereceden bir $\mathbf{A}=\left[a_{ij}\right]$ matrisi $i\neq j$ için $a_{ij}=0$ ise köşegen matristir. Yani **A** matrisi,

$$\begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

şeklindedir.

### 2.1.2. Matrisin Alt uzayları

**Tanım 10:** **V** vektör uzayı, vektör toplaması ve skalar vektör çarpımı gibi işlemlerin tanımlandığı bir vektörler kümesi olsun. **V**'nin alt kümesi olan **W**, **V**'nin yapabildiği vektör işlemlerini kullanabilen bir vektör uzayı ise **W**'ye **V**'nin alt uzayı denir.

Bir $\mathbf{A}\in\mathbb{R}^{m\times n}$ matrisinin en önemli iki alt uzayı, değer ve çekirdek alt uzaylarıdır.

**Tanım 11:** Bir **A** matrisinin değer alt uzayı

$$range(\mathbf{A})=\left\{\mathbf{y}=\mathbf{Ax}\mid\mathbf{x}\in\mathbb{R}^{n}\right\} \tag{2.5}$$

şeklinde ifade edilir.

**Tanım 12:** Bir **A** matrisi için $\mathbf{Ax} = 0$ eşitliğinin çözüm kümesine **A**'nın çekirdek alt uzayı denir ve

$$ker(\mathbf{A}) = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{Ax} = 0\} \tag{2.6}$$

şeklinde gösterilir.

**Tanım 13:** $a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \cdots + a_n\mathbf{x}_n = 0$ eşitliği sadece $a_1 = a_2 = \cdots = a_n = 0$ koşulunda sağlanıyorsa $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ vektör kümesi lineer bağımsızdır denir. **V** vektör uzayının, maksimum sayıdaki lineer vektörlerinin kümesine **V**'nin bir bazı denir. **V**'nin bazındaki vektör sayısına **V**'nin boyu denir. **V**'nin boyu dim(**V**) ile gösterilir.

**Tanım 14:** Bir $\mathbf{A} \in \mathbb{R}^{m \times n}$ matrisinin rankı

$$rank(\mathbf{A}) = \dim(range(\mathbf{A})) \tag{2.7}$$

şeklinde tanımlanır. Eğer $rank(\mathbf{A}) = \min(m, n)$ ise **A** matrisi tam ranklıdır, aksi halde **A** eksik ranklıdır.

**Tanım 15:** **A** matrisi kare ve tam ranklı bir matris ise tekil olmayan matristir.

**Teorem 2.1:** **A** matrisi kare ve tekil olmayan bir matris olmak üzere,

$$\mathbf{A^{-1}A = AA^{-1} = I} \tag{2.8}$$

şeklinde benzersiz bir $\mathbf{A^{-1}}$ matrisi vardır. Bu matris **A**'nın tersi olarak adlandırılır.

**Tanım 16:** $\mathbf{A} \in \mathbb{R}^{m \times n}$ matrisinin sözde tersi

$$\mathbf{A^{\dagger} = (A^T A)^{-1} A^T} \tag{2.9}$$

şeklinde tanımlanır.

### 2.1.3. Matris Normları

Normlar sayısal problemlerin çözümlerindeki hataları veya hassaslıkları ölçmek için kullanılan skalar büyüklüklerdir. Matris normları $\|.\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ şeklinde,

herhangi bir $\alpha \in \mathbb{R}$ ve $\mathbf{A,B} \in \mathbb{R}^{m\times n}$ için aşağıdaki koşulları sağlayan fonksiyonlardır.

1. $\|\mathbf{A}\| \geq 0 ve \|\mathbf{A}\| = 0 \Leftrightarrow \mathbf{A} = 0$ ,
2. $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|,$
3. $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|.$

Önemli bazı matris normları şu şekilde ifade edilir:

- 1 – norm: Diğer adıyla sütun normu.

$$\|\mathbf{A}\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$$

- ∞ – norm: Diğer adıyla satır normu.

$$\|\mathbf{A}\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

- 2 – norm: Diğer adıyla Euclid normu.

$$\|\mathbf{A}\|_2 = \left( \sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2}$$

- Frobenius norm:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

Frobenius norm ve 2 – norm, alt çarpım özelliği olarak bilinen

$$\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\| \tag{2.10}$$

eşitsizliğini sağlar.

**Tanım 17:** Herhangi bir **x** vektörü için

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\| \tag{2.11}$$

eşitsizliği sağlanıyorsa, $\|\mathbf{A}\|$ matris normuna $\|\mathbf{x}\|$ vektör normu ile uygundur denir.

### 2.1.4. Ortogonal Dönüşümler

**Tanım 18: Q** matrisi *n* boyutlu kare matris olmak üzere,

$$\mathbf{Q^T Q = Q Q^T = I_n} \tag{2.12}$$

şartını sağlıyorsa ortogonal matristir.

**Teorem 2.2: Q** matrisi *n* boyutlu ve ortogonal ise

$$\mathbf{Q^T = Q^{-1}} \tag{2.13}$$

eşitliğini sağlar.

Bir vektör ile ortogonal bir matrisin çarpılmasına ortogonal dönüşüm denir.

**Teorem 2.3:** Ortogonal dönüşümler 2 – normu korur.

**İspat: Q** ortogonal matris, **x** vektör ve **A** bir matris olmak üzere

$$\|\mathbf{Qx}\|_2^2 = \langle \mathbf{Qx}, \mathbf{Qx} \rangle = \mathbf{x}^T \mathbf{Q}^T \mathbf{Qx} = \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|_2^2 \tag{2.14}$$

eşitliğinden vektörün ortogonal dönüşümde 2 – normu koruduğu görülür.
A matrisinin 2 – normu

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2 = 1} \|\mathbf{Ax}\|$$

şeklindedir. Buradan $\|\mathbf{QA}\|_2$ normunu bulursak,

$$\|\mathbf{QA}\|_2 = \max_{\|\mathbf{x}\|_2 = 1} \|\mathbf{QAx}\|$$

olur. $\mathbf{y} = \mathbf{Ax}$ eşitliğini yerine yazarsak

$$\|\mathbf{QA}\|_2 = \max_{\mathbf{y} \in \mathbb{R}^n} \|\mathbf{Qy}\|_2$$

sonucunu elde ederiz. Eşitlik (2.14)'den $\|\mathbf{Qy}\|_2 = \|y\|_2$ olduğundan

$$\|\mathbf{QA}\|_2 = \max_{\|\mathbf{x}\|_2 = 1} \|\mathbf{Ax}\|$$

$$= \|\mathbf{A}\|_2$$

elde edilir.

**Teorem 2.4:** [0, 2π] aralığındaki benzersiz bir θ açısı için, $2\times 2$ boyutlu **Q** matrisi ortogonal ise

$$\mathbf{Q}=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix} \tag{2.15}$$

şeklindedir.

**İspat:** **Q** matrisi ortogonal olduğundan $\mathbf{QQ}^{T}=\mathbf{I}_2$ ve $\Delta=det\mathbf{Q}=1$ eşitliklerini sağlar. $\mathbf{Q}=\begin{bmatrix}a & b\\ c & d\end{bmatrix}$ alınırsa $\mathbf{Q}^{T}=\mathbf{Q}^{-1}=\frac{1}{\Delta}adj\mathbf{Q}$ eşitliğinden

$$\mathbf{Q}=\begin{bmatrix}a & b\\ c & d\end{bmatrix}=\begin{bmatrix}d & -b\\ -c & a\end{bmatrix} \tag{2.16}$$

elde edilir. Matris eşitliğinden yola çıkarak $a=d$, $c=-b$, $\mathbf{Q}=\begin{bmatrix}a & b\\ -b & a\end{bmatrix}$ ve $a^2+b^2=1$ olarak bulunur. Bu durumda $(a,b)$ noktası birim çember üzerindedir. Buradan [0, 2π] aralığında benzersiz bir θ açısı için $a=\cos\theta$ ve $b=\sin\theta$ yazılabilir. Bulunan bu değerler yerine yazıldığında $\mathbf{Q}=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}$ elde edilir.

### 2.1.4.1. Givens Dönüşümü

Lineer cebirde, Givens dönüşümü ikili koordinat düzleminde verilen bir vektörün döndürülmesidir. Givens dönüşümünün kullanılmasının temel amacı vektör veya matrislerdeki sıfırları ortaya koymaktır [14].

Bazı $\theta$ açıları için $c=\cos\theta$ ve $s=\sin\theta$ olduğunda $\mathbf{I}_n$ birim matrisinin rankı 2 olan ve (2.15)'de verilen bir modifikasyonudur [2]. Givens dönüşümünün matris olarak gösterimi

$$\mathbf{G}(i,j,\theta)=\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{2.17}$$

şeklindedir.

$\mathbf{x}\in\mathbb{R}^n$ bir vektör iken $\mathbf{G}(i,j,\theta)$ ile çarpılmadan önce $\mathbf{G}(i,j,\theta)\mathbf{x}=\mathbf{y}$ olacak şekilde $\mathbf{y}\in\mathbb{R}^n$, $\mathbf{y}=\mathbf{x}$ $i,j$ bileşenleri hariç ve

$$\begin{aligned} \mathbf{y}_i &= c\mathbf{x}_i + s\mathbf{x}_j \\ \mathbf{y}_j &= -s\mathbf{x}_i + c\mathbf{x}_j \end{aligned} \tag{2.18}$$

elde edilir.

Eğer

$$c = x_i/\gamma,\ s = x_j/\gamma,\ \gamma = \sqrt{x_i^2 + x_j^2} \tag{2.19}$$

şeklinde düzenlenirse,

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} x_i \\ x_j \end{bmatrix}=\begin{bmatrix} \gamma \\ 0 \end{bmatrix} \tag{2.20}$$

elde edilir. Givens algoritması $\alpha,\beta\in\mathbb{R}$ için $c,s$ ve $\gamma$'yı $-s\alpha+c\beta=0$ olacak şekilde kurar. Bu şekilde tanımlanan Givens dönüşümü, çarpıldığı **x** vektörünü saat yönünde $\theta$ açısı kadar döndürür. Bu durumu örnek üzerinde inceleyelim.

**Örnek 2.1:** $\mathbf{x}=\begin{bmatrix} 1 \\ \sqrt{3} \end{bmatrix}$ ve $\theta=\dfrac{\pi}{3}$ radyan olmak üzere ortogonal Givens matrisi

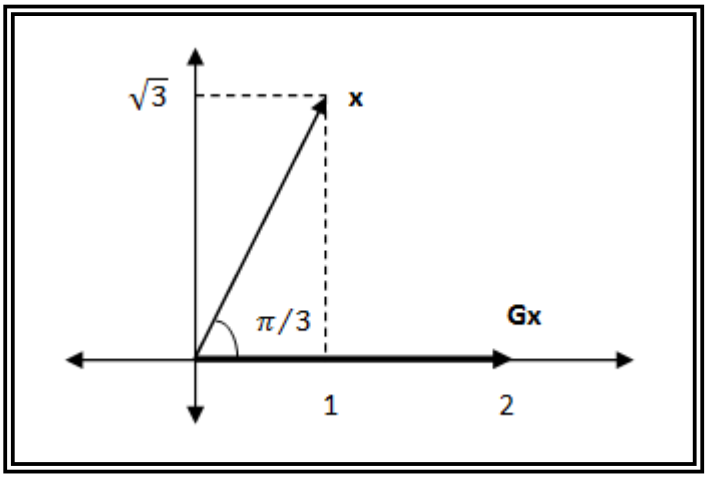
$$\mathbf{G}=\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}=\frac{1}{2}\begin{bmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & 1 \end{bmatrix}$$

şeklindedir. Buradan

$$\mathbf{G}\mathbf{x}=\frac{1}{2}\begin{bmatrix} 1 & \sqrt{3} \\ -\sqrt{3} & 1 \end{bmatrix}\begin{bmatrix} 1 \\ \sqrt{3} \end{bmatrix}=\begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

olarak bulunur.

Şekil 2.1‘de **x** vektörünün saat yönünde döndürülmesi gösterilmiştir.




**Şekil 2.1**: Vektörün Düzlemde Döndürülmesi

Çizelge 2.1’de Givens algoritmasının nasıl çalıştığı verilmiştir.

**Çizelge 2.1:** Givens Algoritması

**Girdi:** α ve β skalarları

**Çıktı:** $\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \gamma \\ 0 \end{bmatrix}$ şeklindeki $c,\ s,\ \gamma$ skalarları

$[c, \quad s, \quad \gamma] = GIVENS(\alpha, \beta)$

(1) if($\beta = 0$)

(2) $c = 1; \quad s = 0; \quad \gamma = \alpha;$

(3) else if ($|\beta| > |\alpha|$)

(4) $t = \alpha/\beta; \quad tt = \sqrt{1 + t^2};$

(5) $s = 1/tt\,; \quad c = t\,s; \quad \gamma = tt\,\beta;$

(6) else

(7) $t = \beta/\alpha\,; \quad tt = \sqrt{1 + t^2};$

(8) $c = 1/tt\,; \quad s = t\,c; \quad \gamma = tt\,\alpha;$

### 2.1.4.2. Householder Dönüşümü

**v**, sıfır olmayan bir *n* – vektör olmak üzere, **H** ile gösterilen Householder matrisi

$$\mathbf{H} = I_n - \frac{1}{\mu}\mathbf{v}\,\mathbf{v}^T, \qquad \mu = \frac{1}{2}\mathbf{v}^T\mathbf{v} \tag{2.21}$$

şeklindeki matristir. Herhangi bir vektör veya matrisin **H** matrisi ile çarpılmasına Householder dönüşümü denir.

**Teorem 2.5:** Householder matrisi hem ortogonal hem de simetriktir. Yani **H** bir Householder matrisi olmak üzere

$$\mathbf{H} = \mathbf{H}^T = \mathbf{H}^{-1} \tag{2.22}$$

şartını sağlar.

**İspat:** $\mathbf{H} = \boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T$ matrisindeki $\mathbf{v}^T\mathbf{v}$ ifadesinin reel sayı olduğu aşikârdır. Öncelikle $\mathbf{H}^T = \mathbf{H}$ olduğunu gösterelim.

$$\mathbf{H}^T = \left(\boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T\right)^T = \boldsymbol{I}_n{}^T - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right)^T$$

$\left(\mathbf{v}\mathbf{v}^T\right)^T = \left(\mathbf{v}^T\right)^T\mathbf{v}^T = \mathbf{v}\mathbf{v}^T$ ve $\boldsymbol{I}_n{}^T = \boldsymbol{I}_n$ eşitliklerini yerlerine yazarsak;

$$\mathbf{H}^T = \boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\mathbf{v}\mathbf{v}^T = \mathbf{H}$$

elde edilir.

Sonra $\mathbf{H}^T = \mathbf{H}^{-1}$ olduğunu gösterelim. Bunun için $\mathbf{H}\mathbf{H}^T = \boldsymbol{I}_n$ olduğunu göstermemiz yeterlidir.

$$\mathbf{H}\mathbf{H}^T = \left(\boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right)\right)\left(\boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right)\right)^T$$

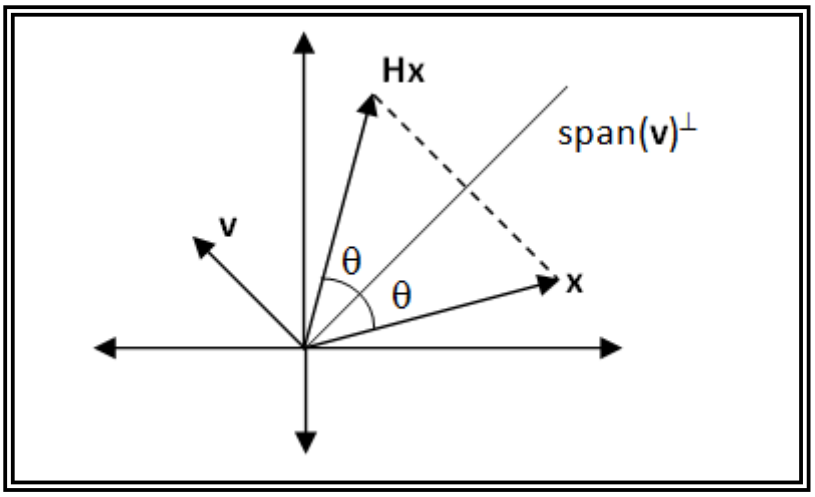
$$= \boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right) - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right) + \frac{4}{\left(\mathbf{v}^T\mathbf{v}\right)^2}\left(\mathbf{v}\mathbf{v}^T\ \mathbf{v}\mathbf{v}^T\right)$$

$\mathbf{v}\mathbf{v}^T\,\mathbf{v}\mathbf{v}^T = \mathbf{v}\left(\mathbf{v}^T\,\mathbf{v}\right)\mathbf{v}^T = (\mathbf{v}^T\mathbf{v})\mathbf{v}\mathbf{v}^T$ değişimini yapar ve yerine koyarsak

$$\mathbf{H}\mathbf{H}^T = \boldsymbol{I}_n - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right) - \frac{2}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right) + \frac{4}{\mathbf{v}^T\mathbf{v}}\left(\mathbf{v}\mathbf{v}^T\right) = \boldsymbol{I}_n$$

olarak bulunur.

Şekil 2.2'de Householder yansıması gösterilmiştir.




**Şekil 2.2:** Householder Yansıması

Givens dönüşümü vektörü $\mathbf{G} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ şeklinde bir ortogonal matrisle çarparak Şekil 2.1'de görüldüğü gibi negatif yönde $\theta$ açısı kadar döndürür. Householder yansımasında $v_1 = cos\theta$ olarak düşünüldüğünde vektör,

$$\mathbf{H} = \begin{bmatrix} 1-2v_1^2 & -2v_1v_2 \\ -2v_1v_2 & 1-2v_2^2 \end{bmatrix} = \begin{bmatrix} \cos 2\theta & -\sin 2\theta \\ -\sin 2\theta & -\cos 2\theta \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ -\sin\alpha & -\cos\alpha \end{bmatrix}$$

matrisiyle çarpılacaktır. Bu durumda Householder dönüşümü Şekil 2.2'de görüldüğü gibi vektörü pozitif yönde $2\theta$ kadar döndürür. [2]

Çizelge 2.2'de Householder algoritması verilmiştir.

**Çizelge 2.2:** Householder Algoritması

**Girdi:** n – vektör **x**

**Çıktı:** n – vektör **v**

$[\boldsymbol{v}, \quad \mu, \quad \sigma_{yeni}] = HOUSE(\boldsymbol{x})$

(1) $\sigma = (\boldsymbol{x^T x})^{1/2}$;

(2) $\alpha_1 = \boldsymbol{x^T e_1}$;

(3) $\sigma_{yeni} = -sgn(\alpha_1)\sigma$;

(4) $\boldsymbol{v} = \boldsymbol{x} - \sigma_{yeni}\boldsymbol{e_1}$;

(5) $\mu = \sigma(\sigma + |\alpha_1|)$;

## 2.2. Matlab

### 2.2.1. Matlab nedir?

Matlab, Matworks firmasının oluşturduğu, teknik hesaplamalar, matematiksel problemlerin çözümü ve analizi için geliştirilmiş bir programlama dilidir. Programın ismi "MATrix LABoratory" kelimesinin kısaltmasından gelmektedir. Özellikle mühendislik alanındaki sistemlerin analizinde kullanılan Matlab, kontrol, görüntü işleme, istatistik, optimizasyon, finans, bulanık mantık, sinir ağları, grafik, veri tabanı gibi başlı başına her bir konu için farklı bir araç kutusuna sahiptir [15][16].

C, C++, Fortran, Delphi, Pascal ve benzeri geleneksel programlama dillerinin aksine Matlab, daha özel amaçları gerçekleştirebilmek için tasarlanmıştır. Matlab, Diferansiyel denklemlerin çözümü, simülasyon geliştirme, grafikler çizdirme ve bunlar üzerinde değişiklikler yapma, finansal veri analizi yapma gibi birçok özel işlemlerin yapılmasını inanılmaz derecede kolaylaştırır. Matlab'in tipik kullanım alanlarını şu şekilde sıralayabiliriz [16]:

- Matematik ve hesaplama,

- Algoritma geliştirme,
- Veri toplama ve gerçek zamanlı kontrol,
- Modelleme, simülasyon ve prototip geliştirme,
- Veri analizi ve grafiksel görüntüleme,
- Grafiksel arabirimi ile uygulama geliştirme [16].

Matlab'in çeşitli işletim sistemlerinde çalışan sürümleri mevcuttur. Bu tezin gerçekleştirilmesinde MATLAB R2009b sürümü kullanılmıştır. Çalışmanın yapıldığı bilgisayarın özellikleri, işlemci: Intel Core2Duo, 3.06GHz, ram: 3Gb, işletim sistemi: Windows 7 şeklindedir.

### 2.2.2. Matlab'de Matris İşlemleri

Matlab, adından da anlaşılacağı gibi matrisler diğer bir deyişle diziler ile çalışır. Yani Matlab, kullanacağı görüntü vb. verileri matrisler aracılığı ile hafızada saklar ve işlem yapar.

Bir dizi tek bir değerden, vektörden, matristen veya çok boyutlu diziden oluşabilir. Bir dizi tek bir değer veya vektör ve matristen de oluşmuş olsa aslında iki boyutludur.

Matlab'de matrisler köşeli parantezler içine bileşenler arasına bir boşluk ve satır sonlarına noktalı virgül konularak ifade edilir. Sadece boyutun belirtildiği özel matrisler ise parantez içine boyut bilgisinin girilmesi ve aralarına virgül konulması ile tanımlanır.

Matlab'de tanımlanan dizi veya matrisler tam sayılardan, ondalıklı sayılardan veya karakterlerden oluşabilirler. Tez boyunca kullanılacak olan dizi ve matrisler reel sayılarda tanımlı olacaklardır.

Matlab'de tanımlanan matrisin istenilen kısımlarını görmek için aşağıdaki komutlar kullanılır. Örnek matrisimiz **A** olmak üzere;

- **A**'nın *i-inci* satırındaki *j-inci* sütun elemanına erişmek için,

$$\mathbf{A}(i, j)$$

- **A**'nın *i-inci* satırının tüm elemanlarına erişmek için,

$$\mathbf{A}(i,:)$$

- **A**'nın *j-inci* sütunun tüm elemanlarına erişmek için,

$$\mathbf{A}(:, j)$$

- **A**'nın *k-ıncı* ve *l-inci* satırları arasındaki elemanlarını göstermek için,

$$\mathbf{A}(k:l,:)$$

- Benzer şekilde **A**'nın *k-ıncı* ve *l-inci* sütunları arasındaki elemanlarını göstermek için,

$$\mathbf{A}(:,k:l)$$

komutları kullanılır.

Matlab'de matrislerin toplanması, çıkarılması veya çarpılması işlemleri tek satırlık komutlarla gerçekleştirilebilir. Dikkat edilmesi gereken işlem yapılacak olan matrislerin boyutlarının birbirine uygun olmasıdır. Matlab'de uygulanan matris işlemlerinin bazıları aşağıdaki gibidir.

- Matrisin boyutu yani satır ve sütun sayısı

$$size(\mathbf{A})$$

- Matrislerin toplanması

$$\mathbf{A} + \mathbf{B}$$

- Matrislerin farkı

$$\mathbf{A} - \mathbf{B}$$

- Matrislerin çarpımı

$$\mathbf{A}*\mathbf{B}$$

- Matrisin skalar ile çarpımında her eleman o skalar ile çarpılır

$$\alpha*\mathbf{A}$$

- Matrisin tersi

$$inv(\mathbf{A})$$

- Matrisin devriği(transpozu)

$$\mathbf{A}'$$

Matlab'de bazı özel matrislerin tanımlamaları için özel komutlar vardır. Bunlardan bazılarını m ve n satır ve sütun boyutları olmak üzere aşağıdaki gibi ifade edebiliriz.

- Sıfır matrisi

$$zeros(m,n)$$

- Birim matris

$$eye(m,n)$$

- Rastgele sayı üreten matris, bu matris 0 ve 1 arasında değerler üretir

$$rand(m,n)$$

## 2.3. Matris Ayrışımları

Daha önce de belirtildiği gibi karmaşık bir matrisin daha basit matrislerin çarpımı biçiminde ifade edilmesine matris ayrışımı denir. Elde edilen matrislerin, temel matrisin rank bilgisini ve öz vektörlerini koruması sebebiyle matris ayrışımı pek çok alanda kullanışlı bir yöntemdir.

### 2.3.1. SVD Ayrışımı

Herhangi bir matrisin tekil değer ayrışımı (Singular Value Decomposition) SVD ayrışımı olarak adlandırılır. SVD ayrışımı, bir matrisin çarpanlarına ayrılma türlerinden biridir. İnsan yüzlerini modellemeden gen analizine, bilgi çıkarımından veri sıkıştırmaya kadar uzanan geniş bir çalışma alanının temel adımıdır [17]. Bu yüzden, istatistik, görüntü işleme ve sinyal işleme gibi pek çok alanda kullanılmaktadır. Aynı zamanda SVD ayrışımı, bir matrisin rankını ortaya çıkarmak için en çok kullanılan tekniktir.

Matematiksel olarak, $m \times n$ boyutlu ve rankı $r$ olan bir $\mathbf{A}$ matrisinin SVD ayrışımı, $m \geq n$ olmak üzere,

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} \tag{2.23}$$

şeklinde ifade edilir. Burada $\mathbf{U} \in \mathbb{R}^{m \times m}$ ve $\mathbf{V} \in \mathbb{R}^{n \times n}$ olmak üzere ikisi de ortogonaldir. $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ şeklinde köşegen matristir. $\mathbf{A}$'nın tekil değerleri olarak adlandırılan $\sigma_i$ ler $\boldsymbol{\Sigma}$'nın köşegen elemanlarıdır. $\mathbf{A}$'nın rankına göre aşağıdaki durumlardan bahsedilebilir:

1. $rank(\mathbf{A}) = n$ ise

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & \sigma_n \end{bmatrix} \tag{2.24}$$

şeklindedir. $\boldsymbol{\Sigma}$'nın tekil değerleri $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$ biçiminde sıralanmış olabilir.

2. $rank(\mathbf{A}) = r < \min(m,n)$ ise $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ sıralaması geçerlidir.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.25}$$

şeklinde yazılabilir ve $\mathbf{\Sigma}_1 = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ 'dir.

Bir matrisin SVD ayrışımını bulmak için kullanılan Matlab kodu,

**[U,S,V] = svd(A)**

şeklindedir. Bu komut satırı ile **A** matrisinin ortogonal çarpanları olan **U** ve **V** matrisleri ile tekil değerlerini içeren **S** matrisi bulunur.

Örnek bir **A** matrisi

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

şeklinde belirlensin. Bu matrisin SVD ayrışımı **[U,S,V] = svd(A)** ile bulunursa;

$$\mathbf{U} = \begin{bmatrix} -0.2663 & -0.5774 & 0.4072 & 0.6557 & 0 \\ -0.2240 & -0.5774 & -0.6619 & -0.1883 & -0.3780 \\ -0.4903 & 0.5774 & -0.2548 & 0.4674 & -0.3780 \\ -0.5122 & -0.0000 & 0.5446 & -0.5461 & -0.3780 \\ -0.6133 & 0.0000 & -0.1861 & -0.1335 & 0.7559 \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} 3.6246 & 0 & 0 & 0 & 0 \\ 0 & 1.4142 & 0 & 0 & 0 \\ 0 & 0 & 1.2286 & 0 & 0 \\ 0 & 0 & 0 & 0.5941 & 0 \\ 0 & 0 & 0 & 0 & 0.0000 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.4458 & -0.8165 & 0.0844 & -0.3570 & 0 \\ -0.4458 & 0.4082 & 0.0844 & -0.3570 & 0.7071 \\ -0.3663 & 0.0000 & -0.8976 & 0.2452 & 0.0000 \\ -0.4458 & 0.4082 & 0.0844 & -0.3570 & -0.7071 \\ -0.5193 & -0.0000 & 0.4158 & 0.7466 & -0.0000 \end{bmatrix}$$

matrisleri elde edilir. Elde edilen **S** matrisinden yola çıkarak **A** matrisinin rankının dört olduğu bilgisine ulaşılır ve **A**'nın tekil değerleri sıralı bir şekilde ortaya çıkar. Ayrıca **U** ve **V** matrisleri de ortogonalliğin tanımı olan $\mathbf{U}\mathbf{U}^T = \mathbf{V}\mathbf{V}^T = \mathbf{I}$ koşulunu sağlarlar.

#### 2.3.1.1. Sayısal Rank

Veri matrisinde belirsizlikler ve yuvarlanmış hataların olmadığı durumlarda SVD, matrisin rankını bulmada kullanışlı bir yöntemdir. Fakat hataların varlığı ve belirsizlik gibi durumlar rank sorusunu anlamsız kılar. Tam ranklı olmayan matrislerdeki ufak bir değişim rankı etkileyecektir. Bu durumda rank tanımını yeniden uyarlamak gerekmektedir [18].

**Tanım 18:** Bir $\mathbf{A} \in \mathbb{R}^{m \times n}$ matrisi için

$$k = \min\{rank(\mathbf{B}) : \|\mathbf{A} - \mathbf{B}\|_2 \leq \varepsilon\} \tag{2.26}$$

olacak şekilde bir **B** matrisi varsa **A**, *k sayısal* $\varepsilon - rank$ 'a sahiptir denir.

$\sigma_i, i = 1,2,\ldots,\min(m,n)$ **A**'nın tekil değerleri olmak üzere **A** matrisinin *k sayısal* $\varepsilon - rankı$ için

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > \varepsilon \geq \sigma_{k+1} \geq \cdots \geq \sigma_n \tag{2.27}$$

sıralaması geçerlidir.

**A** matrisinin sayısal rankının *k* olduğunu varsayalım. $\mathbf{\Sigma}_1 = diag(\sigma_1, \sigma_2, \ldots, \sigma_k)$ ve $\mathbf{\Sigma}_2 = diag(\sigma_{k+1}, \sigma_{k+2}, \ldots, \sigma_n)$ köşegen matrisleri olmak üzere A'nın tekil değer ayrışımı

$$\mathbf{A} = (\mathbf{U}_1\ \mathbf{U}_2)\begin{pmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & \mathbf{\Sigma}_2 \\ 0 & 0 \end{pmatrix}(\mathbf{V}_1\ \mathbf{V}_2)^T \tag{2.28}$$

şeklindedir.

$\mathbf{A}_1 = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$ olmak üzere, Tanım 17'dan yola çıkarsak,

$$\|\mathbf{A} - \mathbf{A}_1\|_2 = \|\mathbf{A}\mathbf{V}_2\|_2 \leq \varepsilon, \quad \mathbf{V}_2 = (v_{k+1}, v_{k+2}, \ldots, v_n) \tag{2.29}$$

eşitlikleri sağlanır. Ayrıca,

$$range(\mathbf{V}_2) = span(v_{k+1}, v_{k+2}, ..., v_n) \tag{2.30}$$

ifadesine **A** matrisinin sayısal çekirdek uzayı veya kirlilik uzayı denir.

SVD, **A** matrisinin alt uzay, rank ve matris iyileştirme gibi bilgilerini barındırır. Düşük rank varsayımının hesabında kullanışlı bir yöntemdir [19]. SVD'nin bu ve benzeri yararlarının aksine, veri matrisine yeni veri eklendiğinde veya çıkarıldığında gerekli güncellemenin matematiksel hesapla karmaşasının $O(n^3)$ olması gibi dezavantajları da bulunmaktadır. Bu sebeple, alternatif olarak güncelleme işlemlerini daha hızlı gerçekleştiren ULV ve URV ayrışımları kullanılmaktadır [20]. Devam eden kesimde bu ayrışım yöntemlerine değinilecektir.

### 2.3.2. QR Ayrışımı

Sütun rankı tam olan bir matrisin, ortogonal bir matris ve satır rankı tam olan üst üçgensel matrisin çarpımı şeklinde ifade edilmesi QR ayrışımıdır. Tam ranklı matrislerin tekil değerlerini bulmada kullanışlı bir yöntemdir.

$m \times n$ boyutlu bir **A** matrisinin QR ayrışımı, $m \geq n$ olmak üzere,

$$\mathbf{A} = \mathbf{QR} \tag{2.31}$$

şeklinde tanımlanır. Burada $\mathbf{Q} \in \mathbb{R}^{m \times n}$ şeklinde ortogonal matris ve $\mathbf{R} \in \mathbb{R}^{n \times n}$ şeklinde üst üçgensel matristir.

**A** matrisinin QR ayrışımının hesaplanmasında kullanılabilen ideal algoritmalardan biri Gram-Schmidt algoritmasıdır. Bunu uygulamada Householder, Givens, Fast Givens algoritmalarından biri kullanılabilir.

Bir matrisin QR ayrışımını bulmak için kullanılan Matlab kodu

**[Q,R]=qr(A)**

şeklindedir.

Örnek bir **A** matrisi

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

şeklinde oluşturulsun. Bu matrisin QR ayrışımı **[Q,R]=qr(A)** Matlab kodu ile hesaplanırsa;

$$\mathbf{Q} = \begin{bmatrix} -0.5000 & -0.3536 & 0.2041 & 0.6660 & -0.3739 \\ -0.5000 & -0.3536 & -0.6124 & -0.4705 & -0.1693 \\ 0 & 0.7071 & -0.4082 & 0.1955 & -0.5432 \\ -0.5000 & 0.3536 & 0.6124 & -0.4705 & -0.1693 \\ -0.5000 & 0.3536 & -0.2041 & 0.2749 & 0.7126 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} -2.0000 & -1.0000 & -1.0000 & -1.0000 & -1.5000 \\ 0 & 1.4142 & 0.7071 & 1.4142 & 1.0607 \\ 0 & 0 & -1.2247 & -0.0000 & 0.2041 \\ 0 & 0 & 0 & -0.0000 & 0.6660 \\ 0 & 0 & 0 & 0 & -0.3739 \end{bmatrix}$$

matrisleri oluşur. **R** matrisi ayrışımın koşulunu sağlayan üst üçgensel bir matristir. **Q** matrisi de ortogonalliğin tanımı olan $\mathbf{QQ}^T = \mathbf{I}$ koşulunu sağlar.

### 2.3.3. ULV ve URV Ayrışımı

ULV ve URV ayrışımları, tam ranklı olmayan sistemlerin çözümünde kullanışlı ve SVD'ye alternatif olarak görülen matris ayrışımlarıdır.

Matematiksel olarak, $m \times n$ boyutlu bir **A** matrisinin $\mathbf{U} \in \mathbb{R}^{m \times m}$ ve $\mathbf{V} \in \mathbb{R}^{n \times n}$ ortogonal matrisler ve $\mathbf{L} \in \mathbb{R}^{m \times n}$ alt üçgensel matris olmak üzere,

$$\mathbf{A} = \mathbf{ULV}^{\mathbf{T}} \tag{2.32}$$

şeklinde tanımlanan ayrışımına ULV ayrışımı denir.

Benzer şekilde, $m \times n$ boyutlu bir **A** matrisinin $\mathbf{U} \in \mathbb{R}^{m \times m}$ ve $\mathbf{V} \in \mathbb{R}^{n \times n}$ ortogonal matrisler ve $\mathbf{R} \in \mathbb{R}^{m \times n}$ üst üçgensel matris olmak üzere,

$$\mathbf{A} = \mathbf{URV}^{\mathbf{T}} \tag{2.33}$$

şeklinde belirtilen ayrışımına da URV ayrışımı denir.

Bu ayrışımların SVD'ye göre avantajı, **A** matrisine satır eklendiğinde veya çıkarıldığında güncellenmesinin hesaplama karmaşasının düşük olmasıdır.

Pek çok uygulamada **A** matrisi, eski veri setinin etkisini azaltmak için 1'den küçük herhangi bir sabitle çarpılır. Yapılan bu işlem üstten pencereleme yöntemi olarak bilinir [21]. Gerçekleştirilen bu tezde de üstten pencereleme yöntemi uygulanmıştır.

# 3. ARAŞTIRMA BULGULARI

## 3.1. Kesik(Truncated) ULV Ayrışımı

Kesik ULV ayrışımı, temeli ULV ayrışımına dayanan ve uygulama alanında esas alınan matris ayrışımıdır.

$m \geq n$ olmak üzere, $\mathbf{A} \in \mathbb{R}^{m \times n}$ matrisinin kesik ULV ayrışımı(TULV)

$$\mathbf{A} = \mathbf{U}_1 \mathbf{L} \mathbf{V}_1^T + \mathbf{E} \tag{3.1}$$

şeklinde ifade edilir. Burada $k \leq n$ olmak üzere $\mathbf{U}_1 \in \mathbb{R}^{m \times k}$ ve $\mathbf{V}_1 \in \mathbb{R}^{n \times k}$ matrisleri soldan ortogonal, $\mathbf{L} \in \mathbb{R}^{k \times k}$ matrisi tersi alınabilen alt üçgensel matris ve $\mathbf{E} \in \mathbb{R}^{m \times n}$ hata matrisidir. **L** ve **E** matrisleri

$$\left\| \mathbf{L}^{-1} \right\|_2 \leq \varepsilon^{-1}, \quad \left\| \mathbf{E} \right\|_2 < \varepsilon, \quad \mathbf{U}_1^T \mathbf{E} = \mathbf{0} \tag{3.2}$$

şartlarını sağlar. Buradaki $\varepsilon$ değeri belirtilen toleranstır [11].

## 3.2. Kesik ULV Ayrışımının Blok Güncellemesi

**A** veri matrisine yeni bir veri satırı eklendiğinde, oluşan yeni matrisin kesik ULV ayrışımının elde edilmesi sürecine güncelleme denir. **a**, yeni veri sütunu olmak üzere oluşan yeni veri matrisi;

$$\overline{\mathbf{X}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{a}^T \end{pmatrix} \tag{3.3}$$

şeklindedir.

Oluşan yeni matrisin tamamının ULV ayrışımını yeniden hesaplamak yerine, önceki matrisin ULV ayrışımını ve yeni eklenen verinin bilgilerini kullanarak, yeni matrisin ULV ayrışımını bulmak daha az maliyetli bir işlemdir. Özellikle büyük boyutlu matrisler söz konusu olduğunda, bu yöntemi kullanmak avantajlıdır.

Bu çalışmada, $\mathbf{X}$ matrisin kesik ULV ayrışımı kullanılarak, yeni veri matrisi olan $\bar{\mathbf{X}}$ matrisinin kesik ULV ayrışımını hesaplayan bir blok güncelleme algoritması geliştirilmiştir. Çalışmanın temel amacı, $\mathbf{A}^T$ yeni eklenen matris bloğu olmak üzere

$$\bar{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \mathbf{A}^T \end{pmatrix} \tag{3.4}$$

matrisinin kesik ULV ayrışımını hesaplayan bir algoritmanın oluşturulmasıdır. Geliştirilen algoritma benzerlerinden farklı olarak bloklar halinde eklenen verilerin kesik ULV ayrışımının güncelleme işlemini yapmaktadır.

Önceki bölümde de bahsedildiği gibi, bir matrisin QR ayrışımını bulmada kullanılabilecek en temel yöntemlerden biri Classical Gram-Schmidth algoritmasıdır. Bu çalışmada geliştirilen algoritma da Block Classical Gram-Schmidth (BCGS2) [22] algoritmasını kullanmaktadır. BCGS2 algoritmasının temeli, bir matrisin ortogonal çarpanlarına ayırma işlemi olan **local_qr** fonksiyonudur. Bu fonksiyon bir matrisin QR ayrışımını hesaplamayı sağlar. Fonksiyon, $p \le n \le m$ olmak üzere, $\bar{\mathbf{X}} \in \mathbb{R}^{m \times p}$ matrisi için

$$[\bar{\mathbf{Q}}, \bar{\mathbf{R}}] = \mathbf{local\_qr}(\bar{\mathbf{X}}) \tag{3.5}$$

matrislerini üretir. Burada $\bar{\mathbf{R}} \in \mathbb{R}^{p \times p}$ üst üçgensel matris ve $\bar{\mathbf{Q}} \in \mathbb{R}^{m \times p}$ soldan ortogonal matristir. $\bar{\mathbf{Q}}$ ve $\bar{\mathbf{R}}$, [22]'de verilen

$$\left\| \mathbf{I}_p - \bar{\mathbf{Q}}^T \bar{\mathbf{Q}} \right\| \le \varepsilon_M L_1(m, p) < 1 \tag{3.6}$$

$$\bar{\mathbf{X}} + \Delta\bar{\mathbf{X}} = \bar{\mathbf{Q}}\bar{\mathbf{R}}, \qquad \left\| \Delta\bar{\mathbf{X}} \right\| \le \varepsilon_M L_1(m, p) \left\| \bar{\mathbf{X}} \right\| \tag{3.7}$$

denklemlerini sağlar. Burada $L_1(m, p)$ küçük değerli bir fonksiyon ve $\varepsilon_M$ makine sayısıdır. **local_qr** fonksiyonu Householder veya Givens QR ayrışımı algoritmalarından biri kullanılarak kodlanabilir. Householder QR ayrışımının hata

analizi [23]'de verilmiştir, bunun sonucunda üretilen $L_1(m, p) = d_1 mp^{3/2}$ ifadesinde $d_1$ bir sabittir.

Çalışmada kullanılan BCGS2 algoritması, aşağıda Çizelge 3.1'de verilen One Block CGS Step algoritmasını kullanmaktadır. Bu algoritma da kendi içinde **local_qr** fonksiyonunu kullanmaktadır.

**Çizelge 3.1:** Algoritma 1 - One Block CGS Step

**function** $\left[\bar{Q}, \bar{R}, \bar{S}\right]$ **= block_CGS_step(**$U, A$**)**
$\bar{S} = U^T A;$
$\bar{X} = A - U\bar{S};$
$\left[\bar{Q}, \bar{R}\right] =$ **local_qr**$(\bar{X});$
**end block_CGS_step**

Kullanılan bu algoritmanın giriş değerleri, **X** matrisinin kesik ULV ayrışımı sonucu elde edilen **U** matrisi ve yeni eklenen veri bloğu olan **A** matrisidir. Bu giriş matrisleri için çalıştırılan algoritma, BCGS2 algoritması için gerekli olan çıkış değerlerini yani $\bar{\mathbf{Q}}$, $\bar{\mathbf{R}}$ ve $\bar{\mathbf{S}}$ matrislerini oluşturur. Algoritmanın çıktısı olan $\bar{\mathbf{R}}$ üst üçgensel matristir, $\bar{\mathbf{Q}}$ matrisi yaklaşık sol ortogonaldir ve (3.6) ile verilen eşitsizliği sağlar. Bunun yanında, $\bar{\mathbf{Q}}$ ve $\bar{\mathbf{X}}$ matrisleri (3.7)'yi sağlarlar [22].

Çalışmada kullanılan BCGS2 algoritmasının adımları aşağıda Çizelge 3.2'de verilmiştir. Bu algoritma One Block CGS Step algoritmasının iki kere uygulanmasına dayanmaktadır. Benzer şekilde **X** matrisinin kesik ULV ayrışımı ile elde edilen **U** matrisini ve yeni eklenen veri bloğu olan **A** matrisini giriş olarak almaktadır.

**Çizelge 3.2:** Algoritma 2 - Two Steps of Block CGS

**function** $\left[Q_A, R_A, S_A\right]$ **= block_CGS2_step(**$U, A$**)**
$\left[\bar{Q}_1, \bar{R}_1, \bar{S}_1\right] =$ **block_CGS_step**$(U, A)$;
$\left[Q_A, \bar{R}_2, \bar{S}_2\right] =$ **block_CGS_step**$(U, \bar{Q}_1)$;
$S_A = \bar{S}_1 + \bar{S}_2 \bar{R}_1$;
$R_A = \bar{R}_2 \bar{R}_1$;
**end block_CGS2_step**

### 3.3. Blok Güncelleme Algoritması

Günümüzde görüntü ve sinyal işleme gibi pek çok alanda üzerinde çalışılan veriler sürekli güncellenmektedir. Bu veriler işlenirken matrislerden faydalanılmakta ve veriler matrisler üzerinde işlenmektedir. Mevcut verilere yenileri eklenmekte veya tam tersi mevcut verilerden veri eksilmektedir. Yani matrisler üzerindeki veriler artmakta veya azalmaktadır.

Verilerin devamlılığının olduğu bu gibi durumlarda matris ayrışımlarının daha etkin, hızlı ve az maliyetli olarak kullanılabilmesi için güncelleme algoritmalarına ihtiyaç duyulmaktadır. Güncelleme işlemini tek bir satırdan daha fazlası olarak yani bloklar halinde yapabilen algoritmalar çok daha avantajlıdır. Geliştirilen algoritma ile bu işlemin gerçekleştirilmesi sağlanmıştır.

Mevcut matrisin kesik ULV ayrışımı bilgilerinden yola çıkılarak, yeni oluşan matrisin kesik ULV ayrışımının hesaplanması sağlanmıştır. Şöyle ki, $m \geq n$ olmak üzere $\mathbf{X} \in \mathbb{R}^{m \times n}$ matrisi kesik ULV ayrışımı (3.1) ile verilen matris olsun. (3.4) ile verilen $\bar{\mathbf{X}}$ matrisi,

$$\bar{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \mathbf{A}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U}_1\mathbf{L}\mathbf{V}_1^T + \mathbf{E} \\ \mathbf{A}^T \end{pmatrix} = \begin{pmatrix} \mathbf{U}_1\mathbf{L}\mathbf{V}_1^T \\ \mathbf{A}^T \end{pmatrix} + \begin{pmatrix} \mathbf{E} \\ 0 \end{pmatrix}$$

biçiminde yeniden yazılabilir [24].

Çizelge 3.2 ile verilen **block_CGS2_step** algoritmasının önceden belirlenen **A** blok matrisi ve $\mathbf{V}_1$ matrisi girdileri ile kullanımı sonucunda

$$\left[\mathbf{V}_{new}, \mathbf{L}_{new}^T, \mathbf{S}_{new}\right] = \mathbf{block\_CGS2\_step(V_1, A)};$$

Eşitliği elde edilir ve buradan,

$$\mathbf{A}^T = \mathbf{S}_{new}^T\mathbf{V}_1 + \mathbf{L}_{new}\mathbf{V}_{new}^T$$

sonucuna ulaşılır. Yani $\bar{\mathbf{X}}$ matrisi,

$$\bar{\mathbf{X}} = \begin{pmatrix} \mathbf{U}_1 & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{L} & 0 \\ \mathbf{S}_{new}^T & \mathbf{L}_{new} \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^T \\ \mathbf{V}_{new}^T \end{pmatrix} + \begin{pmatrix} \mathbf{E} \\ 0 \end{pmatrix}$$

şeklinde yeniden yazılabilir. Kesik ULV ayrışımın tam sonucunu elde edebilmek için ortada bulunan $\begin{pmatrix} \mathbf{L} & 0 \\ \mathbf{S}_{new}^T & \mathbf{L}_{new} \end{pmatrix}$ matrisine Barlow, Erbay ve Slapničar [25] tarafından önerilen **refinement** algoritması uygulanmıştır. Bu algoritmanın uygulanması, olası hataları ortadan kaldırmış ve daha kesin sonuçlar elde edilmesini sağlamıştır.

Kesik ULV ayrışımının blok güncelleme algoritmasının Matlab benzeri kodu Çizelge 3.3'de verilmiştir. Oluşturulan bu algoritma **X** matrisinin kesik ULV

ayrışımını ve yeni eklenen veri bloğu olan **A** matrisini giriş olarak almaktadır. Sonuç olarak ise $\overline{\mathbf{X}}$ matrisinin kesik ULV ayrışımını bulmaktadır.

**Çizelge 3.3:** Algoritma 3 - Kesik ULV Ayrışımı Blok Güncelleme Algoritması

**function** $\left[\bar{U}_1, \bar{L}_1, \bar{V}_1\right]$ **= TULV_block_update(**$U_1, L, V_1, A$**)**

$\bar{X} = \begin{pmatrix} X \\ A^T \end{pmatrix}$;

$\left[V_{new}, L_{new}^T, S_{new}\right] =$ **block_CGS2_step(**$V_1, A$**);**

$\tilde{U}_1 = \begin{pmatrix} U_1 & 0 \\ 0 & I \end{pmatrix}$; $\tilde{L} = \begin{pmatrix} L & 0 \\ S_{new}^T & L_{new} \end{pmatrix}$; $\tilde{V}_1 = \begin{pmatrix} V_1 & V_{new} \end{pmatrix}$; $\tilde{E} = \begin{pmatrix} E \\ 0 \end{pmatrix}$;

$\left[\bar{U}_1, \bar{L}_1, \bar{V}_1\right] =$ **TULV_refinement(**$\bar{X}, \tilde{U}_1, \tilde{L}, \tilde{V}$**);**

**end TULV_block_update**

Yapılan çalışmanın sonucunda elde edilen bazı teorik sonuçlar aşağıda teoremler olarak verilmiştir. Elde edilen bu sonuçlar, oluşturulan algoritmanın çalışmasını ve sonuçlarını doğrular niteliktedir.

**Teorem 3.1:** $\overline{\mathbf{X}} = \bar{\mathbf{U}}_1 \bar{\mathbf{L}} \bar{\mathbf{V}}_1^T + \bar{\mathbf{E}}$ eşitliği verilmiş olsun. O halde, $\bar{\mathbf{U}}_1^T \bar{\mathbf{E}} = 0$ 'dır.

**İspat:** $\mathbf{U}_1^T \mathbf{E} = 0$ olduğu biliniyor. O zaman,

$$\begin{aligned}
\bar{\mathbf{U}}_1^T \bar{\mathbf{E}} &= \bar{\mathbf{U}}_1^T \left( \bar{\mathbf{X}} - \bar{\mathbf{U}}_1 \bar{\mathbf{L}} \bar{\mathbf{V}}_1^T \right) \\
&= \bar{\mathbf{U}}_1^T \bar{\mathbf{X}} - \bar{\mathbf{U}}_1^T \bar{\mathbf{U}}_1 \bar{\mathbf{L}} \bar{\mathbf{V}}_1^T \\
&= \bar{\mathbf{U}}_1^T \bar{\mathbf{X}} - \bar{\mathbf{L}} \bar{\mathbf{V}}_1^T \\
&= \begin{pmatrix} \mathbf{U}_1^T & 0 \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{A}^T \end{pmatrix} - \begin{pmatrix} \mathbf{L} & 0 \\ \mathbf{S}_{new}^T & \mathbf{L}_{new} \end{pmatrix} \begin{pmatrix} \mathbf{V}_1^T \\ \mathbf{V}_{new}^T \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{U}_1^T \mathbf{X} \\ \mathbf{A}^T \end{pmatrix} - \begin{pmatrix} \mathbf{L} \mathbf{V}_1^T \\ \mathbf{S}_{new}^T \mathbf{V}_1 + \mathbf{L}_{new} \mathbf{V}_{new}^T \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{U}_1^T \mathbf{X} - \mathbf{L} \mathbf{V}_1^T \\ \mathbf{A}^T - \mathbf{S}_{new}^T \mathbf{V}_1 - \mathbf{L}_{new} \mathbf{V}_{new}^T \end{pmatrix} \\
&= 0
\end{aligned}$$

$\bar{\mathbf{U}}_1^{\dagger}$, $\bar{\mathbf{U}}_1$'in Moore-Pensore sözde tersi olmak üzere bu matris, aşağıda verilen eşitlikleri sağlayan benzersiz bir matristir [3].

$$\bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{U}}_1 = \bar{\mathbf{U}}_1 \tag{3.8}$$

$$\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger} = \bar{\mathbf{U}}_1^{\dagger} \tag{3.9}$$

$$\left(\bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\right)^T = \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger} \tag{3.10}$$

$$\left(\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{U}}_1\right)^T = \bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{U}}_1 \tag{3.11}$$

**Teorem 3.2:** $\bar{\mathbf{X}} = \bar{\mathbf{U}}_1\bar{\mathbf{L}}\bar{\mathbf{V}}_1^T + \bar{\mathbf{E}}$ eşitliği verilmiş olsun. O zaman,

$$\bar{\mathbf{E}} = \bar{\mathbf{P}}\bar{\mathbf{X}}, \quad \bar{\mathbf{P}} = \mathbf{I} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}$$

eşitliği sağlanır.

**İspat:** $\bar{\mathbf{P}} = \mathbf{I} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}$ olmak üzere;

$$\begin{aligned}
\bar{\mathbf{P}}\bar{\mathbf{X}} &= \left(\mathbf{I} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\right)\bar{\mathbf{X}} \\
&= \bar{\mathbf{X}} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{X}} \\
&= \bar{\mathbf{X}} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\left(\bar{\mathbf{U}}_1\bar{\mathbf{L}}\bar{\mathbf{V}}_1^T + \bar{\mathbf{E}}\right) \\
&= \bar{\mathbf{X}} - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{U}}_1\bar{\mathbf{L}}\bar{\mathbf{V}}_1^T - \bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\bar{\mathbf{E}}
\end{aligned}$$

eşitliğine (3.8) ve (3.10) Moore-Pensore durumları uygulanırsa,

$$\begin{aligned}
\bar{\mathbf{P}}\bar{\mathbf{X}} &= \bar{\mathbf{X}} - \bar{\mathbf{U}}_1\bar{\mathbf{L}}\bar{\mathbf{V}}_1^T + \left(\bar{\mathbf{U}}_1\bar{\mathbf{U}}_1^{\dagger}\right)^T\bar{\mathbf{E}} \\
&= \bar{\mathbf{E}} - \left(\bar{\mathbf{U}}_1^{\dagger}\right)^T\left(\bar{\mathbf{U}}_1\right)^T\bar{\mathbf{E}}
\end{aligned}$$

olarak bulunur. Teorem 3.1'in sonucu olarak $\bar{\mathbf{U}}_1^T\bar{\mathbf{E}} = 0$ eşitliği yerine yazıldığında,

$$\begin{aligned}
\bar{\mathbf{P}}\bar{\mathbf{X}} &= \bar{\mathbf{E}} - \left(\bar{\mathbf{U}}_1^{\dagger}\right)^T\left(\bar{\mathbf{U}}_1\right)^T\bar{\mathbf{E}} \\
&= \bar{\mathbf{E}}
\end{aligned}$$

bulunur.

### 3.4. Sayısal Sonuçlar

Daha önce de belirtildiği gibi çalışmada gerçekleştirilen güncelleme algoritması üstten pencereleme yöntemi kullanılarak test edilmiştir. Bu yöntemde eski verilerin etkisini azaltmak için $\mathbf{X} \in \mathbb{R}^{m \times n}$ veri matrisi, $\alpha (0 < \alpha \leq 1)$ unutma faktörü ile çarpılarak, **X** veri matrisine $\mathbf{A}^T$ yeni veri bloğu eklenmiştir. *t* zamanlı **X**(*t*) veri matrisi

$$\mathbf{X}(t) = \begin{pmatrix} \alpha \mathbf{X}(t-1) \\ \mathbf{A}^T \end{pmatrix} \tag{3.12}$$

şeklinde oluşturulmuştur.

Uygulama, Matlab R2009b ile test edilmiştir. **X**(*t*) matrisinin her *t* adımdaki sayısal rankını bulmak için Çizelge 3.3 ile verilen kesik ULV ayrışımının blok güncelleme algoritması kullanılmıştır. Elde edilen sonuçlara referans olması için ise her *t* adımda Matlab'in SVD'si ile sayısal rank hesaplanmıştır. **X**(*t*) veri matrisinin rankı her iki yöntemle de bulunup, karşılaştırılmıştır.

$\mathbf{V}_1(t)$ ve $\mathbf{U}_1(t)$ matrislerinin soldan ortogonal olup olmadığını kontrol etmek için sırayla

$$\left\| \mathbf{I} - \mathbf{V}_1^T(t)\mathbf{V}_1(t) \right\|_2$$

$$\left\| \mathbf{I} - \mathbf{U}_1^T(t)\mathbf{U}_1(t) \right\|_2$$

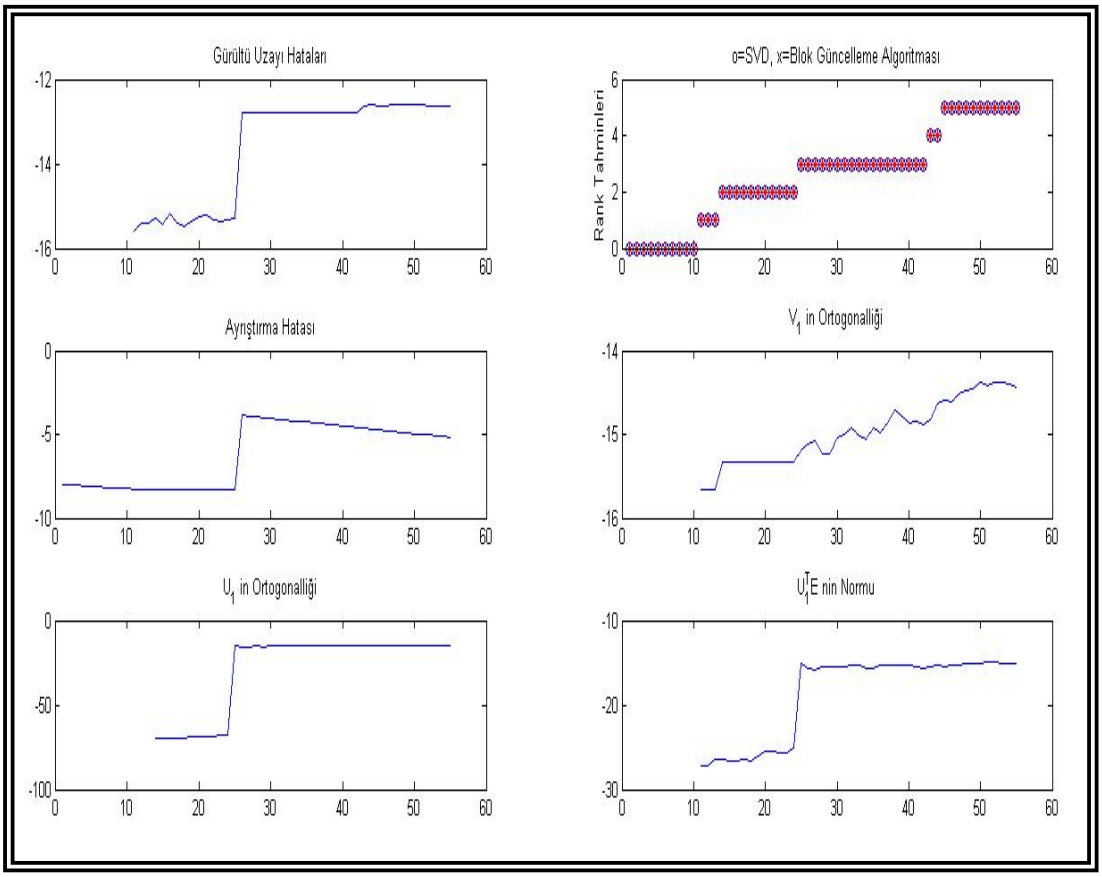
normları hesaplatılmıştır. Ayrıca, ayrışım hataları her *t* adımda

$$\left\| \mathbf{E} \right\| = \left\| \mathbf{X}(t) - \mathbf{U}_1(t)\mathbf{L}(t)\mathbf{V}_1^T(t) \right\|$$

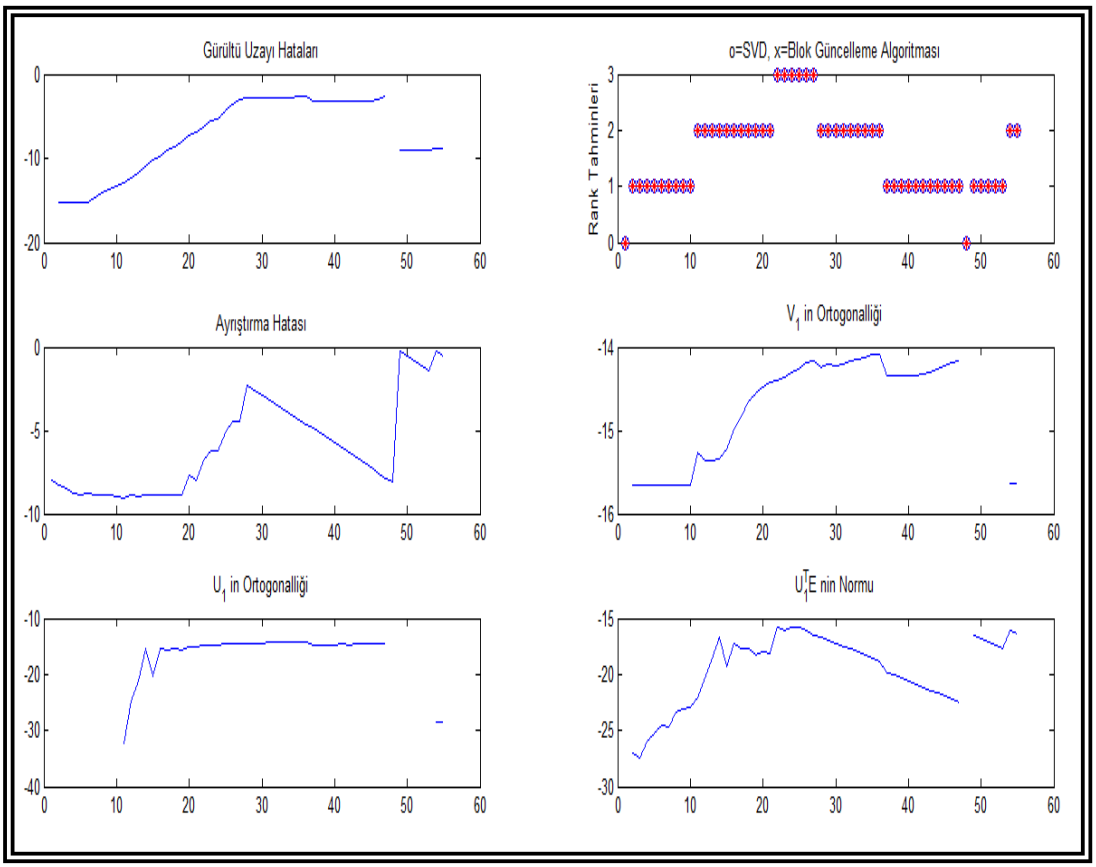
denklemi ile hesaplatılmış ve bir matris içinde saklanmıştır.

Kesik ULV ayrışımının blok güncelleme algoritmasının her adımında, hatalar, daha iyi bir görüntü elde etmek için log10 tabanında çizilmiştir.

**Örnek 3.1:** **X**, 150*x*8 boyutunda, (0,1) aralığında rastgele değerlerden üretilmiş bir matris olsun. **X** matrisinin rankını değiştirmek amacıyla, rastgele seçilen 145 satırı $\eta = 10^{-9}$ ile çarpılsın ve $\varepsilon \approx 10^{-8}$ olsun. Başlangıç matrisi olan **X**(0), $40 \times 8$ boyutundadır. Onu takip eden **X**(1) bloğu $4x8$ boyutundadır. Sonrasında, her bir adımda birbiri ardına $2 \times 8$ boyutunda veri blokları eklenmiştir. Uygulama için seçilen unutma faktörleri $\alpha = 0.9$ ve $\alpha = 0.5$ olarak belirlensin. Elde edilen sonuçlar sırasıyla Şekil 3.1 ve Şekil 3.2'de gösterilmiştir.




**Şekil 3.1:** $\alpha = 0.9$ için elde edilen sonuçlar




**Şekil 3.2:** $\alpha = 0.5$ için elde edilen sonuçlar

Şekil 3.1 ve Şekil 3.2'deki ilk çizimler, blok güncelleme algoritması ile elde edilen alt uzay hatalarını göstermektedir. İkinci çizimlerde ise Matlab'in SVD algoritması referans olarak kullanılarak blok güncelleme algoritmasının hesapladığı rank tahminleri gösterilmektedir. Çizimlerden de görüldüğü üzere, kullanılan algoritma rank değişiminin sıklığına rağmen rankı doğru tahmin etmektedir. Çizimde yatay eksen pencere adımlarını temsil etmektedir. Üçüncü çizimler ayrıştırma hatası olarak belirtilen **E** matrisinin normunu göstermektedir. Teorik olarak sıfır kabul edilen bu matrisin norm değerleri, sayısal olarak da sıfıra çok yakındır. Dördüncü ve beşinci pencerelerde sırası ile $\mathbf{V}_1$ ve $\mathbf{U}_1$ matrislerinin ortogonalliği kontrol edilip çizdirilmiştir. Son çizimde ise, teorik olarak doğru kabul ettiğimiz $\mathbf{U}_1^T\mathbf{E} = \mathbf{0}$ eşitliğinin sayısal olarak da doğru olduğu gösterilmiştir.

# 4. TARTIŞMA VE SONUÇ

Matrisler, pek çok bilimsel konunun uygulanması sırasında kullanılan işlevsel bilgi saklayıcılardır. Günümüzde görüntü işleme, sinyal işleme, video işleme, veri sıkıştırma, istatistik ve benzeri pek çok alanda verilerin işlenmesi sırasında matrislerden faydalanılmaktadır. Her türlü bilgi matrislerde saklanmakta, bu matrisler üzerinde işlemler yapılmakta ve geriye dönüş sağlanmaktadır. Tüm bu işlemler gerçekleştirilirken karmaşık bir matrisin yerine bu matrisin özelliklerini taşıyan daha basit bir matrisin incelenmesi daha kullanışlıdır. Bunu sağlayan en temel yöntem ise matris ayrışımıdır. Ayrışım sonucunda oluşan matrisler, temel matrise ait rank bilgisi, öz değerler, öz vektörler gibi önemli bilgileri muhafaza ederler. Dolayısıyla, oluşan daha küçük boyutlu bu matrislerle çalışmak daha hızlı ve daha az maliyetlidir.

Matris ayrışımı denildiğinde akla gelen en temel yöntem SVD ayrışımıdır. SVD, bir matrisin sayısal rankını güvenli bir biçimde ortaya çıkarır ve gerekli öz değer ve öz vektör bilgilerini barındırır. Fakat SVD algoritması veri akışının devamlı olduğu ve güncelleme gerektiren işlemler için oldukça maliyetli bir algoritmadır. Bu durum, alternatif ayrışım yöntemlerini araştırmacıların gündemine getirmiştir. Son zamanlarda öne çıkan alternatif ayrışımlar ULV ve URV ayrışımlarıdır.

Tezin çalışma konusu olan kesik ULV ayrışımının temeli ULV ayrışımıdır. Bu ayrışım, düşük ve yüksek ranklı matrisler için kullanılabilecek, sayısal rankı güvenli bir şekilde ortaya çıkaran ve gerekli alt uzay bilgilerini içeren bir ayrışım yöntemidir. ULV ayrışımı, SVD'ye göre daha hızlı ve daha az maliyetli olması sebebiyle öne çıkmaktadır. Günümüzde yapılan pek çok araştırma bu durumu doğrulayan sonuçlar ortaya koymaktadır. Yapılan çalışmalarda, mevcut veri matrisine satır eklenmesi veya çıkarılması işlemlerinin ULV ayrışımı ile güvenilir bir biçimde gerçekleştirildiği gösterilmektedir. Yani, özellikle veri akışının devamlı olduğu durumlarda, ULV ayrışımını kullanmak SVD'ye göre daha avantajlıdır.

Bu bilgiler göz önünde bulundurularak yapılan çalışmada, mevcut matris ayrışımı algoritmalarına alternatif olabilecek bir algoritma geliştirilmesi amaçlanmıştır. Tezde bir veri matrisine yeni veri bloğu eklendiğinde, mevcut bilgiyi kullanarak kesik ULV ayrışımını güncelleyen bir algoritma verilmiştir. Mevcut çalışmalarda güncelleme işlemi her satırın tek tek eklenmesi ile yapılmaktadır. Geliştirilen bu algoritma, benzerlerinden farklı olarak birden fazla veri satırının eklendiği problemlerde etkin bir güncelleme işlemi sağlamaktadır.

Çalışmada elde edilen sonuçların daha doğru olmasını sağlamak için algoritmaya bir adım eklenmiştir. Güncelleme süreci sonunda oluşan alt üçgensel matris bozulmuş ranklı olabilir. Bu durumun sonuçları etkilememesi ve olası hataları düzeltmek için **refinement** algoritması kullanılmıştır. Kullanılan algoritma, hata matrisinin etkisini azaltır ve daha doğru sonuçların elde edilmesini sağlar. Böylece daha keskin sonuçlar elde edilmesi sağlanır.

Çalışmanın devamında, örnek bir veri matrisine eklenen bloklarla elde edilen nümerik sonuçlar, SVD ile karşılaştırmalı olarak verilmiştir. Ortaya çıkan teorik sonuçlar algoritmanın doğruluğunu; sayısal sonuçlar ise kesik ULV ayrışımının SVD'ye alternatif olabileceği gösterilmiştir.

Ayrıca yapılan bu çalışmada, görüntü işlemenin temellerine değinilmiştir. Yapılan literatür taraması göstermiştir ki, kesik ULV ayrışımı görüntü işleme, sinyal işleme vb. konularda SVD'ye alternatif olarak kullanılmaktadır. Elde edilen sonuçlar ve benzer çalışmaların ışığında ULV ayrışımının, görüntü işleme ve sinyal işleme gibi pek çok alanda uygulanabilecektir.

İlerleyen çalışmalarda ise bu çalışmada geliştirilen blok update algoritmasına ilaveten blok downdate algoritmasının geliştirilmesi planlanmaktadır. Ayrıca, kesik ULV ayrışımının blok güncellemesi algoritmasının değişik alanlarda özellikle mühendislikteki uygulamaları merak konumuz arasındadır. Bu algoritmanın görüntü düzeltmede uygulanması yakın hedeflerimiz arasındadır.

# KAYNAKÇA


[1] Zhaoyang, L., Matris Ayrışımı, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü Yüksek Lisans Tezi, İstanbul, 2006.

[2] Kocaoğlu, S., ULV Ayrışımı ve Uygulamaları, Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, Kırıkkale, 2006.

[3] Golub, G. H., Van Loan, C. F., Matrix Computations Third Edition, The Johns Hopkins University Press, Baltimore, 1996.

[4] Scharf, L.L., "The SVD and Reduced Rank Signal Processing", Signal Processing, Volume 25, Issue 2, 113-133 (1991)

[5] Chan, T. F., "Rank Revealing QR Factorizations", Linear Algebra and Applications, Volumes 88-89, 67-82 (1987)

[6] Faddeev, D. K., Kublanovskaja, V. N., Faddeeva, V. N., "Solutions of Linear Algebraic Systems with Rectangular Matrices", Proceedings of the Steklov Institute of Mathematics, Volume 96, 93-111 (1968)

[7] Hanson, R. J., Lawson, C. L., "Extensions and Applications of The Householder Algorithm for Solving Linear Least Squares Problems", Mathematics of Computation, Volume 23, 787-812 (1969)

[8] Stewart, G. W., "An Updating Algorithm for Subspace Tracking", Signal Processing, Volume 40, Issue 6, 1535-1541 (1992)

[9] Fierro, R. D., Hansen, P. C., "Low Rank Revealing UTV Decompositions", Numerical Algorithms, Volume 15, 37-55 (1997)

[10] Erbay, H., "An Efficient Algorithm for Rank and Subspace Tracking", Mathematical and Computer Modelling, Volume 44, 742-748 (2006)

[11] Barlow, J. L., Erbay, H., "Modifiable Low-Rank Approximation to a Matrix", Numerical Linear Algebra with Applications, Volume 10, Issue 10, 833-860 (2009)

[12] Andrews , H., Patterson, C., "Singular Value Decompositions and Digital Image Processing", Acoustics, Speech and Signal Processing, Volume 24, 26-53 (1976)

[13] Yodchanan, W., "The Effect of Image Rotation on UTV Decomposition", Audio, Language and Image Processing, 1467-1470 (2008)

[14] Anonim, http://en.wikipedia.org/wiki/Givens_rotation (Erişim Tarihi: 18.12.2012)

[15] Anonim, http://matlab.nedir.com (Erişim Tarihi: 24.12.2012)

[16] Çetin, A. E., Herkes İçin Matlab 7, Alfa Yayınları, İstanbul, 2006.

[17] Arı, İ., http://ismailari.com/blog/tekil-deger-ayrisimi-1/ (Erişim Tarihi: 20.05.2013)

[18] Watkins, D. S., Fundemantels of Matrix Computations, John Wiley and Sons, Inc., Canada, 1991.

[19] Hosur, S., Tewfik, A. H., Boley, D., "Generalized URV Subspace Tracking LMS Algorithm", Acoustics, Speech and Signal Processing, Volume 3, 409-412 (1994)

[20] Aba, K., URV Ayrışımı ve Uygulamaları, Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, Kırıkkale, 2008.

[21] Erbay, H., Modifying Rank-Revealing Decompositions, The Pennsylvania State University, The Graduate School, Doktora Tezi, Pennsylvania, 2000.

[22] Barlow, J. L., Smoktunowicz, A., "Reorthogonalized Block Classical Gram-Schmidt", Numerische Mathematic, Volume 123, Issue 3, 395-423 (2003)

[23] Higham, N. J., Accuracy and Stability of Numerical Analysis, Cambirdge University Press, Cambridge, 2002.

[24] Aydoğan, E., Erbay, H., "Block Updates on Truncated ULV Decompositions", Conferance on Computational Science, Engineering and Information Technology, CCSEIT-2013, kabul edildi.

[25] Barlow, J. L., Erbay, H., Slapnicar, I., "An Alternative Algorithm for the Refinement of ULV Decompositions", SIAM Journal on Matrix Analysis and Applications, Volume 27, 198-211 (2005)

[26] Kolman B., Hill, D. R., Uygulamalı Lineer Cebir, Palme Yayıncılık, Ankara, 2002.