

**KOCAELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**2018**

**YAZILIM LABORATUVARI I (BLM307)**

**PROJE 2 RAPORU**

**KARE OLMAYAN MATRİSİN SÖZDE TERSİNİ BULMA**

MEHMET OĞUZ AKTAŞ

150201170

oguzaktass@gmail.com

EBUBAKİR ŞİT

150201245

[sitebubekir@gmail.com](mailto:sitebubekir@gmail.com)

**İçindekiler**

Problem Analizi ve Çözüm Algoritmasının Oluşturulması 2

Yazılım Mimarisi 5

Akış Şeması 5

Yazılımın Çalıştırılması ve Özellikler 6

Kaynakça 7

1 - Problem Analizi ve Çözüm Algoritmasının Oluşturulması

Bu projede, boyutları [1-5] aralığında ve elemanları [1-9] aralığında olacak şekilde kullanıcıdan alınarak veya rastgele oluşturularak kare olmayan bir matris oluşturulması, bu matrisin Moore-Penrose Inverse (pseudoinverse - sözde ters) matrisinin Least Squares Method (en küçük kareler yöntemi) kullanılarak bulunması, yapılan tüm ara işlemler ile birlikte bulunan pseudoinverse matrisin form ekranında gösterilmesi ve tüm işlemler sırasında kullanılan toplama ile çarpma sayısının (computational complexity) form ekranında gösterilmesi istenmektedir. Problemin çözümünü daha kolay yapabilmek için problemi şu şekilde alt parçalara ayırdık;

1 - Matris, determinant ve diğer gerekli lineer cebir konularının tekrar edilmesi

2 - Ana ekranın tasarlanması ve JFrame’de GridLayout classı kullanılarak istenen boyutta nasıl matris gösterileceğinin araştırılması

3 - Rastgele matris oluşturulması ve ana ekrandan matrisin boyutları alınarak matris elemanlarının tek tek belirlenerek oluşturulması modüllerinin yazılması

4 - Pseudoinverse (Moore-Penrose Inverse) matrisi bulmak için Least Squares Method, Singular Value Decomposition ve QR Decomposition yöntemlerinin araştırılması

5 - Moore-Penrose Inverse en küçük kareler yönteminin tüm adımların implementasyonu için gerekli metotların yazılması ve bulunan ara değerler ile pseudoinverse matrisin form ekranında gösterilmesi

6 - Least Squares yöntemine göre yapılan tüm ara işlemlerin (matrisin transpozunun alınması, transpozu ile matrisin çarpılması, çarpım matrisi ile ilk matrisin transpozunun çarpılması) form ekranında gösterilmesi

7 - Toplama ve çarpma sayısı ile birlikte Big-O notasyonu kullanılarak algoritma karmaşıklığının form ekranında gösterilmesi

1 - Matris, determinant ve diğer gerekli lineer cebir konularının tekrar edilmesi

Projede kullanacağımız matris işlemlerinin (determinant, matrislerin çarpımı, matrisin transpozunun alınması, minor matris, matrisin tersinin alınması işlemlerinin) ayrıntılarını hatırlamak amacıyla öncelikle bu konulara çalıştık. 1 ve 2 boyutlu matrislerde nasıl işlem yapılacağını hatırladığımız için 3 ve üzeri boyutlu matrislerdeki çarpma ile başlayarak diğer işlemlerin hangi yöntemler kullanılarak yapıldığını araştırdık. Bunlara çalışırken aynı zamanda ileriki aşamalarda kullanacağımızı bilmeden, double[][] tipinde dizi şeklindeki kare matrisi parametre olarak alıp matrisin transpozunu bulan transpose() metodunu, yine aynı şekilde double[][] tipinde kare matrisin determinantını bulan determinant() metodunu, determinant bulurken kullandığımız matrisin minörünü (alt determinantını) bulan double[][] tipindeki minor() metodunu ve 2 matrisin çarpılması için double[][] tipinde 2 tane matrisi parametre olarak alıp 3 tane iç içe döngü kullanarak matrisleri parametre sırasına göre çarparak çarpım matrisini bulan multiply() metodunu yazdık. Bu metotları yazarken sürekli denemeler yaptık.

2 - Ana ekranın tasarlanması ve JFrame’de GridLayout classı kullanılarak istenen boyutta nasıl matris gösterileceğinin araştırılması

Program ilk açıldığında çalışacak olan ana ekranda (MainWindow classı) Rastgele Matris Oluştur butonunu, boyutları kullanıcıdan almak için 2 tane (sırasıyla satır ve sütun şeklinde) boş JTextField ile birlikte kullanıcının oluşturacağı matris için Matris Elemanlarını Belirle butonunu tasarladık. Rastgele Matris Oluştur butonuna basıldığında RandomMatrixWindow formunun açılmasını sağladık. Matris Elemanlarını Belirle butonuna basıldığında öncelike satır ve sütun boyutlarının alınacağı 2 adet JTextField'in boş olup olmadığını, girilen değerlerin [1-5] aralığında olup olmadığını ve birbirine eşit olup olmadığı kontrollerinin yapıldığı boolean tipindeki checkInputs() metodunu çalıştırarak true dönüyorsa CreateMatrixWindow formunun JTextField'dan alınan satır ve sütun sayısı değerlerinin parametre olarak girilerek açılmasını, eğer false dönüyorsa hangi koşul sağlanmıyorsa o koşul için yazdığımız JOptionPane mesajının görüntülenmesini sağladık. Daha sonra iki seçenekten (boyutlar ile matris elemanlarının rastgele oluşturulması, boyutlar ile matris elemanlarının kullanıcıdan alınması seçeneklerinden) biri kullanılarak oluşturulacak matrisi JFrame formunda göstermek için önce GridLayout classını kullanmayı düşündük. GridLayout’un nasıl kullanıldığını <https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html> adresindeki örneğe bakıp denemeler yaparak öğrendik fakat daha sonra en fazla 5x4 boyutlu matris oluşturulacağı için GridLayout’u gereksiz bulup JLabel’ler ile bu işlemi yapmaya karar verdik.

3 - Rastgele matris oluşturulması ve ana ekrandan matrisin boyutları alınarak matris elemanlarının tek tek belirlenerek oluşturulması modüllerinin yazılması

Öncelikle oluşturulacak rastgele matrisi göstermek için RandomMatrixWindow JFrame classını oluşturduk. Bu formda matrisin tüm elemanlarını ayrı JLabel’lerde göstermek için (satır sayısı \* sütun sayısı) kadar JLabel gerekiyordu. Biz en baştan 25 tane JLabel oluştururak labelleri setVisible(false) metodu ile default olarak görünmez yaptık. Daha sonra java.util kütüphanesindeki Random hazır classını kullanarak önce matrisin satır ve sütun boyutlarını [1-5] aralığında ve birbirinden farklı olacak şekilde rastgele oluşturduk. Satır ve sütun sayısına göre if - else-if blokları içinde gereken miktarda JLabel’i aktif ederek yine Random classını kullanarak matris elemanlarını [1-9] aralığında olacak şekilde belirleyip jLabel’larda matris elemanlarını gösterdik. Tüm bunları showMatrix() metodu içinde yaptık ve bu metodu RandomMatrixWindow formunun default constructorında çalıştırdık, böylece ana ekranda her Rastgele Matris Oluştur butonuna basıldığında rastgele satır ve sütun sayısı ile rastgele matris elemanları oluşturulacak. Matris boyutunun ana ekrandan alınıp matris elemanlarının kullanıcı tarafından belirlenmesi seçeneği için de CreateMatrixWindow ismindeki, default constructorında Integer tipindeki satır ve sütun sayısını parametre olarak alan JFrame classını oluşturduk. Bu formda yine benzer şekilde 25 tane JTextField oluşturarak Matrisi Onayla butonu ile kullanıcıdan alınan input değerlerinin matrisin elemanları olarak atanmasını sağladık. Matrisi Onayla butonuna basıldığında önce girilen inputların [1-9] aralığında olup olmadığı ve boş JTextField olup olmadığı checkInputs() metodu ile kontrol ediliyor, eğer true dönüyorsa matris oluşturularak elemanları atanıyor. Son olarak RandomMatrixWindow ve CreateMatrixWindow formlarında Matrisin Tersini Al butonunu oluşturarak diğer aşamaya geçtik.

4 - Pseudoinverse (Moore-Penrose Inverse) matrisi bulmak için Least Squares Method, Singular Value Decomposition ve QR Decomposition yöntemlerinin araştırılması

Matrisin sözde tersini bulmak için bize önerilen metotlar (Least Squares Method ve Singular Value Decomposition) ile birlikte QR Decomposition yöntemlerini araştırdık. Singular Value Decomposition yönteminin determinantı 0 olan matrislerde de işe yaradığını ve Octave gibi sayısal hesaplama platformlarında default pinv fonksiyonu olarak kullanıldığını öğrendik, ancak bu yöntem matrisin özdeğer ve özvektörlerini bulmayı gerektirdiği için ve bunları bulmak da projenin süresini uzatacağı için Least Squares Method (en küçük kareler yöntemimi) daha kullanışlı bulduk. Bunun dışında bazı Java kütüphanelerini (JAMA, Efficient Java Matrix Library) inceledik. Bu kütüphanelerde genellikle QR Decomposition yönteminin kullanıldığını gördük, nasıl kullanıldığını kodları inceleyerek anlamaya çalıştık. Bu kütüphaneleri kod kısmında kesinlikle kullanmadık, sadece hangi yöntem kullanıldığını inceledik. Bu 3 yöntemden en başta düşündüğümüz gibi Moore-Penrose Inverse en küçük kareler yönteminin en kullanışlı yöntem olduğuna karar verdik.

5 - Moore-Penrose Inverse en küçük kareler yönteminin tüm adımların implementasyonu için gerekli metotların yazılması ve bulunan pseudoinverse matrisin form ekranında gösterilmesi

Moore-Penrose least squares method için oluşturduğumuz PseudoinverseWindow ismindeki JFrame’den default constructor ile RandomMatrixWindow (rastgele matris için) ve CreateMatrixWindow (kullanıcıdan alınan matris için) formlarından double[][] tipinde matris ile elemanlarını, satır ve sütun sayısını aldık. Yöntemdeki formüle göre sözde ters matrisi bulmak için satır > sütun olduğu durumlarda inverse(transpose(matrix) \* matrix) \* transpose(matrix) sonucunu, satır < sütun olduğu durumlarda da transpose(matrix) \* inverse(matrix \* transpose(matrix)) sonucunu hesaplayarak pseudoinverse matrisi bulmamız gerekiyordu. İlk adımda yazdığımız transpose(), determinant() ve minor() metotları ile matris çarpımı için yazdığımız multiply() metodunu üzerinde birkaç değişiklik yaparak bu classa aktardık. Matrisin tersini bulmak için double[][] tipindeki inverse() metodunu yazdık. Tüm yazdığımız metotları yeniden gözden geçirerek hata aldığımız bazı alt sınırlar için bazı değişiklikler yaptık. Örneğin 1x2, 1x3, 1x4 ve 1x5 matrislerde transpozuyla çarpımının determinantını bulurken (1x1 boyutlu matrisin determinantı) genel formülde döngü içindeki “*det += Math.pow(-1, (double) i) \* matrix[0][i] \* determinant(minor(matrix, 0, i));*” kodu bu durum için doğru çalışmadığından dolayı “*if(matrix.length == 1)*” ile koşul ekleyip “*return matrix[0][0]*” ile 1x1 boyutlu matrisin determinantını döndürdük. Metotları doğru şekilde yazdıktan sonra “*pinv = multiply(inverse(multiply(transpose(matrix), matrix)), transpose(matrix));*” ile hesapladığımız pseudoinverse matrisi Octave’deki (MATLAB’ın Linux’taki alternatifi) pinv() fonksiyonu ile test ettik. Farklı boyuttaki matrisler için birçok denemeden sonra doğru çalıştığını teyit ettik. Bulduğumuz pseudoinverse matrisi formda gösterebilmek için yine 25 tane JLabel oluşturduk. Daha sonra satır ve sütun sayısı durumları için teker teker labelleri bulunan pseudoinverse matrisin elemanları ile (virgülden sonra 4 basamak olacak şekilde) doldurarak tasarımı tamamladık.

6 - Least Squares yöntemine göre yapılan tüm ara işlemlerin (matrisin transpozunun alınması, transpozu ile matrisin çarpılması, çarpım matrisi ile ilk matrisin transpozunun çarpılması) form ekranında gösterilmesi

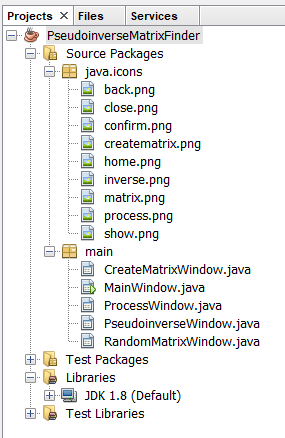
Ara değerleri göstermek için ProcessWindow formunu oluşturduk. PseudoinverseWindow formunda Ara Değerleri Göster butonuna basılınca bu form ekranının açılmasını sağladık. PseudoinverseWindow formunda sözde ters matrisi bulurken hesapladığımız 3 ara değer için double[][] tipinde transpose, multiplywithtranspose ve inverseofmultiplication değişkenlerini tanımlayarak, en küçük kareler yöntemini uygularken bulduğumuz ara matrisleri bu değişkenlere atadık. Daha sonra bu matrisleri parametre olarak ProcessWindow formuna gönderdik. Ara değerleri göstermek yine 3 farklı matris için 25’er tane JLabel oluşturduk, bir tanesi ekrana sığmadığı için onu aynı class içinde jFrame2 isminde başka bir form içinde gösterip ana JFrame’de Matrisi Göster butonu koyduk.

7 - Toplama ve çarpma sayısı ile birlikte Big-O notasyonu kullanılarak algoritma karmaşıklığının form ekranında gösterilmesi

Pseudoinverse matrisi hesaplarken kullanılan toplama ve çarpma sayısını bulmak için önce classın global scope’unda (yani her metodun erişebileceği şekilde) başlangıç değeri 0 olan int tipindeki toplamasayisi ve carpmasayisi değişkenlerini tanımladık. Yazdığımız multiply() ve determinant() metotlarında her toplama kullanıldığında “*toplamasayisi++*” ve her çarpma işlemi kullanıldığında “*carpmasayisi++*” (veya birden fazla işlem yapıldıysa “*carpmasayisi += [yapılan işlem sayısı]*” kodu ile) toplama ve çarpma sayılarını hesapladık. Bu sayıları butonların üzerinde oluşturduğumuz labellerde gösterdik. Son olarak Moore-Penrose Inverse Least Squares Method’un Big-O notasyonu cinsinden algoritma karmaşıklığının ne olduğunu araştırdık. Inverse() ve multiply() metotlarının karmaşıklığının O(n^3) olduğunu bulduk, bu metotları iç içe çalıştırmamıza rağmen döngü içinde sürekli birbirlerini tekrar ederek çalışmadıkları için yöntemin “*multiply(inverse(multiply(transpose(matrix), matrix)), transpose(matrix))*” formülüne göre genel Big-O gösterimini yine O(n^3) olarak hesaplayıp, oluşturduğumuz labelde bunu da form ekranında gösterdik.

2 - Yazılım Mimarisi

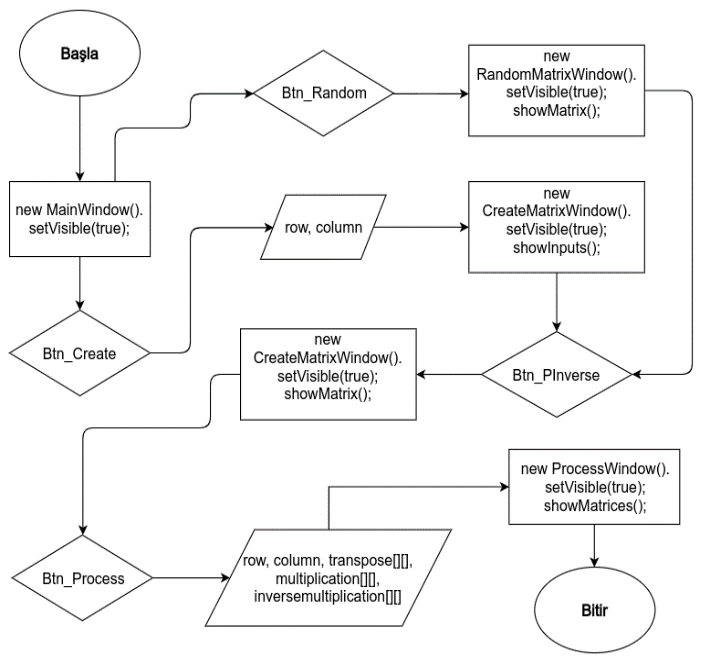
Programı yazmak için JDK 8u191 sürümüyle NetBeans 8.2 IDE’yi ve JDK 11.0.1 sürümüyle JetBrains IntelliJ IDEA’yı, GUI için de Swing ve AWT API’lerini kullandık.



*Şekil 1: Programda bulunan paketler ve sınıflar*

Uygulamada MainWindow, RandomMatrixWindow, CreateMatrixWindow, PseudoinverseWindow ve ProcessWindow isimlerinde 5 tane JFrame formu şeklinde GUI classı kullandık. Tüm classlar main kütüphanesinde bulunuyor. java.icons kütüphanesinde, uygulamada kullandığımız butonlar için gerekli .png biçiminde ikonlar bulunuyor.

3 - Akış Şeması



*Şekil 2: Akış şeması*

4 - Yazılımın Çalıştırılması ve Özellikleri

Programı çalıştırınca ilk olarak ana menü açılıyor.



*Şekil 3: Programın ilk çalıştırılması, ana menü*

Buradan rastgele [1-5] boyutlarında satır ve sütun sayısı eşit (kare matris) olmayacak şekilde rastgele matris oluştur butonu ve matrisin boyutları input olarak girilerek matris elemanlarını belirle butonu kullanılarak istenilen ekrana gidilebilir. Her GUI classında bulunan Ana Ekrana Git butonu ile ana ekrana tekrar tekrar dönülerek aynı işlemler yapılabilir.



*Şekil 4: Rastgele matris oluşturulması*

RandomMatrixWindow classında [1-5] aralığındaki boyutlarda (satır ve sütun sayısı eşit olmayacak şekilde) rastgele matris oluşturularak formda gösterildi.



*Şekil 5: Matrisin sözde tersinin bulunması*

PseudoinverseWindow classı içinde matrisin sözde tersi bulunarak form üzerinde gösterildi. Ayrıca bu formda algoritma karmaşıklığı da hesaplanarak gösterildi.

5 - Kaynakça

[1] <https://stackoverflow.com/>

[2] <https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html> (Erişim tarihi: 09.11.2018)

[3] <http://www.matematikvegeometri.com/matematik-2-konu-anlatimlari-oku/matris-ve-determinant.html> (Erişim tarihi: 09.11.2018)

[4] <http://bilgisayarkavramlari.sadievrenseker.com/2008/11/19/matrisin-tersinin-alinmasi-mantrix-inverse/> (Erişim tarihi: 09.11.2018)

[5] <https://www.codeproject.com/Tips/680040/Matrix-in-Java-GUI> (Erişim tarihi: 09.11.2018)

[6] <https://icons8.com/> (Erişim tarihi: 09.11.2018)

[7] <http://www.math.ucla.edu/~laub/33a.2.12s/mppseudoinverse.pdf> (Erişim tarihi: 10.11.2018)

[8] <https://math.stackexchange.com/questions/458404/how-can-we-compute-pseudoinverse-for-any-matrix> (Erişim tarihi: 10.11.2018)

[9] <https://math.stackexchange.com/questions/2624440/how-to-find-moore-penrose-inverse> (Erişim tarihi: 10.11.2018)

[10] <http://www.cameronmusco.com/personal_site/pdfs/linear_least_squares.pdf> (Erişim tarihi: 11.11.2018)

[11] <https://www.youtube.com/watch?v=PW960_3SZmo> (Erişim tarihi: 24.11.2018)

[12] <http://www.math.ucla.edu/~laub/33a.2.12s/mppseudoinverse.pdf> (Erişim tarihi: 24.11.2018)

[13] <http://www.bluebit.gr/NET/Library/topic181.html> (Erişim tarihi: 24.11.2018)

[14] <http://help.matheass.eu/en/Pseudoinverse.html> (Erişim tarihi: 25.11.2018)

[15] https://cs.stackexchange.com/questions/24060/complexity-of-finding-the-pseudoinverse-matrix (Erişim tarihi: 26.11.2018)