NÜMERİK ANALİZ

Bilimsel Hesaplama Matematiği, Gazi Kitabevi 2012

Nuri ÖZALP

LİNEER SİSTEMLERİN ÇÖZÜMÜ

Gauss elemesi iyi şekilde çalışmaktayken, Doolittle, Crout ve Cholesky ayrıştırmalarına neden gerek duymaktayız?

Gauss elemesi iyi şekilde çalışmaktayken, Doolittle, Crout ve Cholesky ayrıştırmalarına neden gerek duymaktayız?

Hesap makinelerinin kullanıldığı günlerde, bu yordamlardan birinin diğeri üzerinde olası üstünlükleri mevcut idi. Ancak, bilgisayarlardaki ve matematiksel yazılımlardaki gelişmeler sonucunda bu küçük **avantajlar kaybolmuştur**.

Gauss elemesi iyi şekilde çalışmaktayken, Doolittle, Crout ve Cholesky ayrıştırmalarına neden gerek duymaktayız?

Hesap makinelerinin kullanıldığı günlerde, bu yordamlardan birinin diğeri üzerinde olası üstünlükleri mevcut idi. Ancak, bilgisayarlardaki ve matematiksel yazılımlardaki gelişmeler sonucunda bu küçük **avantajlar kaybolmuştur**.

Bundan dolayı, *Doolittle ve Crout* yöntemlerinden bahsedilmesi temelde **tarihsel nedenler** içindir.

Gauss elemesi iyi şekilde çalışmaktayken, Doolittle, Crout ve Cholesky ayrıştırmalarına neden gerek duymaktayız?

Hesap makinelerinin kullanıldığı günlerde, bu yordamlardan birinin diğeri üzerinde olası üstünlükleri mevcut idi. Ancak, bilgisayarlardaki ve matematiksel yazılımlardaki gelişmeler sonucunda bu küçük **avantajlar kaybolmuştur**.

Bundan dolayı, *Doolittle ve Crout* yöntemlerinden bahsedilmesi temelde **tarihsel nedenler** içindir.

Diğer yandan, *Cholesky* yordamı **simetrik, pozitif tanımlı** matrisler için özellikle iyi çalışmaktadır.

Gauss elemesi iyi şekilde çalışmaktayken, Doolittle, Crout ve Cholesky ayrıştırmalarına neden gerek duymaktayız?

Hesap makinelerinin kullanıldığı günlerde, bu yordamlardan birinin diğeri üzerinde olası üstünlükleri mevcut idi. Ancak, bilgisayarlardaki ve matematiksel yazılımlardaki gelişmeler sonucunda bu küçük **avantajlar kaybolmuştur**.

Bundan dolayı, *Doolittle ve Crout* yöntemlerinden bahsedilmesi temelde **tarihsel nedenler** içindir.

Diğer yandan, *Cholesky* yordamı **simetrik, pozitif tanımlı** matrisler için özellikle iyi çalışmaktadır.

Basit Gauss Elemesi

Gauss algoritmasını hatırlayalım:

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 34 \\ 27 \\ -38 \end{bmatrix} \quad \begin{array}{l} \text{operasyonlar} \\ S_2 - 2S_1 \to S_2 \\ S_3 - \frac{1}{2}S_1 \to S_3 \\ S_4 - (-1)S_1 \to S_4 \end{array} \tag{1}$$

$2, \frac{1}{2}$ ve -1 sayıları eleme sürecinin ilk adımı için **çarpanlar** olarak adlandırılır. Bu çarpanların herbirini oluşturmada bölen olarak kullanılan 6 sayısına bu adımın **pivot elemanı** ve değişmeyen 1. satıra **1. pivot satır** denir. İlk adım tamamlandığında sistem şu şekilde olacaktır ve .

İlk adım tamamlandığında sistem şu şekilde olacaktır:

$$\text{pivot} \rightarrow \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 21 \\ -26 \end{bmatrix} \qquad \begin{array}{c} \text{operasyonlar} \\ \\ S_3 - 3S_2 \longrightarrow S_3 \\ S_4 - \left(-\frac{1}{2}\right)S_2 \longrightarrow S_4 \end{array} \tag{2}$$

-4 pivot eleman ve bu adımın çarpanları 3 ve $-\frac{1}{2}$ dir.

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -21 \end{bmatrix} \qquad \begin{matrix} \text{operasyonlar} \\ \\ \\ S_4 - 2S_3 \longrightarrow S_4 \end{matrix} \tag{3}$$

pivot eleman 2 ve çarpan 2 dir.

Sonuç; **İleri Gauss elemesi**:

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -3 \end{bmatrix} \tag{4}$$

Dördüncü satırdan başlayıp, geriye doğru giderek kolayca çözersek; **Geri Gauss yerleştirmesi**:

$$x = \begin{bmatrix} 1 \\ -3 \\ -2 \\ 1 \end{bmatrix}$$

Sistemi dönüştürmede kullanılan çarpanlar bir $L = (\ell_{ij})$ birim alt üçgensel matriste gösterilebilir:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \tag{5}$$

$\longmapsto$Birinci kolonu sıfırlayan çarpanlar $2, 1/2, -1$
$\longmapsto$İkinci kolonu sıfırlayan çarpanlar $3, -1/2$
$\longmapsto$Üçüncü kolonu sıfırlayan çarpanlar 2.

Son sistemin katsayı matrisi bir üst üçgensel $U = (u_{ij})$ matrisi

$$U = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \tag{6}$$

Orjinal sistemin katsayı matrisi A olmak üzere, bu iki matris A nın LU-ayrışımını verir. Böylece,

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \tag{7}$$

Pivotlama

- Gauss algoritması $a_{ii} = 0$ (veya çok küçük iken) çalışmaz:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

olup, algoritmanın basit hali burada işlemez, çünkü ikinci denklemde x_1 in katsayısını 0 yapmak için birinci denklemin bir katını ikinci denkleme eklemenin hiçbir yolu yoktur. (Bkz. Problem **4.2.7**, s. 159.)

Pivotlama

- Gauss algoritması $a_{ii} = 0$ (veya çok küçük iken) çalışmaz:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

  olup, algoritmanın basit hali burada işlemez, çünkü ikinci denklemde x_1 in katsayısını 0 yapmak için birinci denklemin bir katını ikinci denkleme eklemenin hiçbir yolu yoktur. (Bkz. Problem **4.2.7**, s. 159.)

- ε nun 0 dan farklı küçük bir sayı olduğu aşağıdaki sistemde de aynı zorluk devam eder.

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Gauss algoritması uygulandığında

$$\begin{bmatrix} \varepsilon & 1 \\ 0 & 1-\varepsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2-\varepsilon^{-1} \end{bmatrix}$$

üst üçgensel sistemini üretecektir. Bunun (bilgisayar) çözümü ise

$$\begin{cases} x_2 = (2-\varepsilon^{-1})/(1-\varepsilon^{-1}) \approx 1 \\ x_1 = (1-x_2)\varepsilon^{-1} \approx 0 \end{cases}$$

olur. Eğer ε yeterince küçük ise, bilgisayarda $2-\varepsilon^{-1}$ sayısı $-\varepsilon^{-1}$ ile aynı değer olarak hesaplanacaktır. Benzer şekilde $1-\varepsilon^{-1}$ böleni de $-\varepsilon^{-1}$ ile aynı değeri üretir. Böylece, x_2 nin değeri 1 olarak ve x_1 de 0 olarak hesaplanacaktır. Halbuki gerçek çözüm

$$\begin{cases} x_1 = 1/(1-\varepsilon) \approx 1 \\ x_2 = (1-2\varepsilon)/(1-\varepsilon) \approx 1 \end{cases}$$

olduğundan, hesaplanmış çözüm x_2 için duyarlıdır, fakat x_1 için oldukça duyarsızdır!

ε yeterince küçük iken, bilgisayarda $2-\varepsilon^{-1}$ hesabı neden $-\varepsilon^{-1}$ ile aynı olan makine sayısını üretmektedir? Bunun nedeni, bilgisayarda çıkarma işlemi yapılmadan önce 2 nin ve ε^{-1} in, kayan-nokta formundaki *üsler* aynı olacak şekilde taban nokta kaydırması yapıldığındandır. Eğer bu kaydırma yeterince büyük ise, 2 nin mantissası 0 olacaktır. Örneğin Teorik `Marc-32` bilgisayarına benzer bir yedi-nokta desimal makinede, $\varepsilon = 10^{-8}$ için, $\varepsilon^{-1} = 0.1000000 \times 10^{9}$ ve $2 = 0.2000000 \times 10^{1}$ e sahip oluruz. 2 yi 9 üssüne göre tekrar yazarsak $2 = 0.000000002 \times 10^{9}$ ve $2-\varepsilon^{-1} = -0.099999998 \times 10^{9}$, ve böylece makine kaydında $2-\varepsilon^{-1} = -0.1000000 \times 10^{9} = -\varepsilon^{-1}$ elde ederiz.

Problemi yaratan aslında a_{11} katsayısının küçüklüğü değil, aslında aynı satırdaki diğer katsayılara göre a_{11} in *bağıl küçüklüğüdür*. Gerçekten

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2 \end{bmatrix}$$

Basit Gauss algoritması

$$\begin{bmatrix} 1 & \varepsilon^{-1} \\ 0 & 1-\varepsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \varepsilon^{-1} \\ 2-\varepsilon^{-1} \end{bmatrix}$$

sonucunu verir. Bunun çözümü

$$\begin{cases} x_2 = (2-\varepsilon^{-1})/(1-\varepsilon^{-1}) \approx 1 \\ x_1 = \varepsilon^{-1} - \varepsilon^{-1}x_2 \approx 0 \end{cases}$$

dır. Tekrar, küçük ε lar için x_2, 1 olarak ve x_1 de 0 olarak hesaplanır ki, bu önce olduğu gibi hatalıdır!

Bu örneklerdeki zorluklar, denklemlerin sırasını değiştirmekle ortadan kalkar:

$$\begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Gauss elemesi uygulandığında

$$\begin{bmatrix} 1 & 1 \\ 0 & 1-\varepsilon \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-2\varepsilon \end{bmatrix}$$

elde edilir. Böylece, çözüm

$$\begin{cases} x_2 = (1-2\varepsilon)/(1-\varepsilon) \approx 1 \\ x_1 = 2 - x_2 \approx 1 \end{cases}$$

Sonuç; iyi bir algoritmanın, ortamın gerektirdiği durumlarda sistemdeki denklemlerin yerlerini değiştirecek yapıda düzenlenme zorunluluğunu taşımasıdır. Bunun için mantıksal anlamda, **pivot satırları** seçiyoruz. Satırları $1, 2, ..., n-1$ pivot sırasına göre kullanmak yerine, $p_1, p_2, ..., p_{n-1}$ sırasıyla kullanacağımızı kabul edelim. Bu durumda, ilk adımda p_1-inci satırın bir katı diğer satırlardan çıkarılacaktır. $(p_1, p_2, ..., p_n)$, $(1, 2, ..., n)$ nin bir permütasyonu olacak şekilde p_n girdisini sunarsak, p_n bir pivot satırı olmayacaktır, fakat p_1-inci satırın katlarının $p_2, p_3, ..., p_n$. satırlardan çıkarılacağını söyleyebiliriz. İkinci adımda p_2-inci satırın katları $p_3, p_4, ..., p_n$. satırlardan çıkarılacak ve bu şekilde devam edecektir.

Ölçekli Satır Pivotlu Gauss Elemesi

$$Ax = b$$

sistemini çözmek için **ölçekli satır pivotlu Gauss elemesi** algoritması iki kısım içermektedir: Bir **ayrıştırma evresi** (**ileri-yönde eleme** de denir) ve (**güncelleme** ve **geri-yönde yerleştirme** yi içeren) bir **çözüm evresi.**

Ayrıştırma evresi sadece A ya uygulanır ve permütasyon dizisi p den üretilen matris P olmak üzere, PA nın LU ayrışımını oluşturmak için tasarlanır. (PA, A nın satırlarının yeniden sıralanmasıyla elde edilir.) Düzenlenmiş lineer sistem

$$PAx = Pb$$

dir.

$PA = LU$ ayrışımı, aşağıda açıklanacak olan değiştirilmiş Gauss elemesinden elde edilir. Çözüm aşamasında $Lz = Pb$ ve $Ux = z$ denklemlerini göz önüne almaktayız. Önce, b sağ tarafı P ye göre yeniden düzenlenir ve sonuç tekrar b ye yüklenir; yani $b \leftarrow L^{-1}b$ alınır. L birim alt üçgensel olduğundan, bu işlemler ileri-yönde yerleştirmeyi tamamlamış olur. Bu sürece b yi **yenileme** denir. Daha sonra, $Ux = b$ den $x_n, x_{n-1}, ..., x_1$ i çözmek için, geri-yönde yerleştirme yapılır.
Ayrıştırma evresine, herbir satırın **ölçeğini** hesaplayarak başlarız.

$$s_i = \max_{1 \leq j \leq n} \{|a_{i1}|, |a_{i2}|, ..., |a_{in}|\} \qquad (1 \leq i \leq n)$$

Bu değerler algoritmada bir s dizisine kaydedilir.

Ayrıştırma evresine başlarken; $\left|a_{i1}\right|/s_i$ nin en büyük olduğu satırı pivot satırı olarak seçeriz.

Seçilen indis p_1 ile gösterilir ve permütasyon dizisinin ilk elemanı olur. Bu durumda, $1 \leq i \leq n$ için $\left|a_{p_1 1}\right|/s_{p_1} \geq \left|a_{i1}\right|/s_i$ dir.

p_1 belirlendikten sonra, A nın ilk kolonunda sıfırları üretmek üzere, p_1-inci satırın uygun katları diğer satırlardan çıkarılır. Kuşkusuz, p_1-inci satır ayrıştırma sürecinin sonraki aşamalarında değişmez kalacaktır.

Oluşturulan p_i indislerini kayıt altında tutmak için, $\left(p_1, p_2, \ldots, p_n\right)$ sıralama vektörünü $\left(1, 2, \ldots, n\right)$ olarak başlatıyoruz. Daha sonra $\left|a_{p_j 1}\right|/s_{p_j}$ nin en büyük olduğu j indisini seçip, p sıralama dizisinde p_1 ve p_j nin yerlerini değiştiriyoruz.

İlk eleme aşaması, $2 \leq i \leq n$ için p_1-inci satırın $\left(a_{p_i 1}/a_{p_1 1}\right)$ katını p_i-yinci satırlardan çıkarma işlemlerini içerecektir.

Genel süreci açıklamak için, k-yıncı kolonda sıfırlar oluşturmaya hazırlandığımızı farzedelim. En büyük değeri bulmak için $\left|a_{p_i k}\right|/s_{p_i}$ $\left(k \leq i \leq n\right)$ sayılarını tarıyoruz. Eğer, j bu oranların en büyük olduğu ilk indis ise, bu durumda p dizisinde p_k ile p_j nin yerini değiştirip, daha sonra p_k-yıncı satırın $\left(a_{p_i k}/a_{p_k k}\right)$ katını $k+1 \leq i \leq n$ için p_i-yinci satırlardan çıkarırız.

Şimdi bu sürecin nasıl çalıştığını

$$A = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix} \quad \xleftarrow{1.\text{pivot}} \quad s = \begin{bmatrix} 6 \\ 8 \\ 3 \end{bmatrix}$$

matrisinde gösterelim. Başlangıçta $p = (1, 2, 3)$ ve $s = (6, 8, 3)$ olur. İlk pivot satırı seçmek için $\{2/6, 1/8, 3/3\}$ oranlarına bakıyoruz. En büyük oran $j = 3$ e karşılık geldiğinden, 3. satır ilk pivot satır alınır. O halde, p_1 ile p_3 ü yer değiştirip, $p = (3, 2, 1)$ elde ederiz. Şimdi ilk kolonda sıfırları elde etmek için, 3. satırın katları 1. ve 2. satırlardan çıkarılır.

Sonuç:

$$A = \begin{bmatrix} \boxed{\frac{2}{3}} & \frac{13}{3} & -\frac{20}{3} \\ \boxed{\frac{1}{3}} & -\frac{16}{3} & \frac{23}{3} \\ 3 & -2 & 1 \end{bmatrix} \xleftarrow{\text{2.pivot}} \qquad s = \begin{bmatrix} 6 \\ 8 \\ 3 \end{bmatrix}$$

a_{11} ve a_{21} deki *kutulanmış* girdiler çarpanlardır.
İkinci adımda, pivot satırın seçimi $\left|a_{p_2 2}\right| / s_{p_2}$ ve $\left|a_{p_3 2}\right| / s_{p_3}$ oranlarına bakılarak yapılır. İlk oran $(16/3)/8$ ve ikincisi $(13/3)/6$ dır. O halde $j = 3$ olup, p_2 ve p_3 yer değiştirilerek, $p = (3, 1, 2)$; p_2-inci (1.) satırın $\frac{-16}{3}\frac{3}{13} = -\frac{16}{13}$ katı p_3-üncü (2.) den çıkarılır.

Sonuç: $p = (3, 1, 2)$ ve

$$A = \begin{bmatrix} \boxed{\frac{2}{3}} & \frac{13}{3} & -\frac{20}{3} \\ \boxed{\frac{1}{3}} & \boxed{-\frac{16}{13}} & -\frac{7}{13} \\ 3 & -2 & 1 \end{bmatrix}$$

Son çarpan a_{22} ye yerleştirilir.

Eğer orjinal A matrisinin satırları p sıralama dizisinin son haline göre yer değiştirirse, bu durumda A nın bir LU-ayrışımını elde etmiş olmalıyız. Bu durumda

$$PA = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{1}{3} & -\frac{16}{13} & 1 \end{bmatrix} \begin{bmatrix} 3 & -2 & 1 \\ 0 & \frac{13}{3} & -\frac{20}{3} \\ 0 & 0 & -\frac{7}{3} \end{bmatrix} = \begin{bmatrix} 3 & -2 & 1 \\ 2 & 3 & -6 \\ 1 & -6 & 8 \end{bmatrix}$$

olup, burada

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad A = \begin{bmatrix} 2 & 3 & -6 \\ 1 & -6 & 8 \\ 3 & -2 & 1 \end{bmatrix}$$

Permütasyon matrisi P, sıralama dizisi p den $(P)_{ij} = \delta_{p_i,j}$ ile elde edilir. Diğer bir deyişle P, I birim matrisinin satırlarının p deki girdilere göre sıralanmasından oluşur.

Teorem ($PA = LU$ ile Çözüm)

Eğer $PA = LU$ ayrışımı ölçekli satır pivotlu Gauss algoritması ile üretilirse, bu durumda $Ax = b$ nin çözümü, önce $Lz = Pb$ ve sonra da $Ux = z$ çözülerek elde edilir. Benzer şekilde, $y^T A = c^T$ un çözümü, önce $U^T z = c$ ve sonra $L^T Py = z$ çözülerek elde edilir.

İspat

Problem 4.3.47 (s.185)