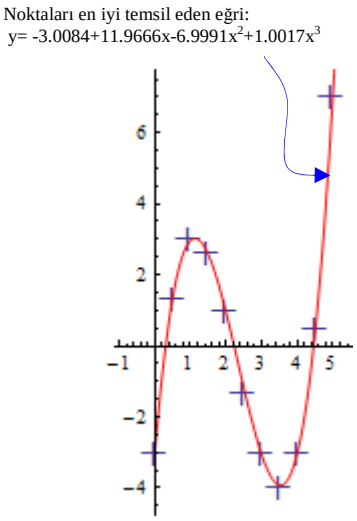
ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
Mühendislik Mimarlık Fakültesi
İnşaat Mühendisliği Bölümü
E-Posta: ogu.ahmet.topcu@gmail.com
Web: http://mmf2.ogu.edu.tr/atopcu

Bilgisayar Destekli

Nümerik Analiz

Ders notları 2014

Ahmet TOPÇU



$$A_{n\times m}X_m=b_n$$

$$n\neq m$$

20

PROGRAMLAR: Denklem sayısı bilinmeyen sayısından farklı doğrusal denklem sistemleri

- n≥m durumu: Dengeleme hesabı
  En küçük kareler metodu ile eğri(polinom) uydurma
  QR çarpanlara ayırma metodu

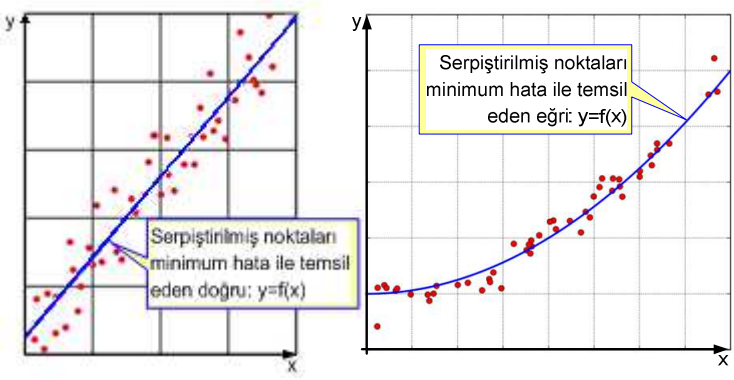
- n≤m durumu: Sağ ters matris ve çekirdek (GAUSS-JORDAN tekniği)

20. PROGRAMLAR

En küçük kareler metodu ile en uygun eğri(polinom) uydurma -LeastSquares

En uygun eğri uydurma problemi(curve fitting); gözlem, ölçüm, deney veya istatistiksel verilerin değerlendirilmesi ve yorumlanmasını basitleştirmek için kullanılan bir yoldur.

Verilmiş (x_1,y_1), (x_2,y_2), …, (x_m,y_m) değerleri x-y eksen takımında gösterildiğinde, notaların bir doğru veya eğri üzerinde olmadığı, x-y düzleminde serpiştirildiği görülür. Bunun nedeni verilerin hata içirmesidir.

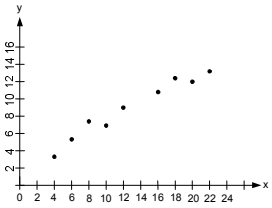


Amaç, bu noktaları en az hata ile temsil eden bir doğru veya eğrinin denkleminin bulunmasıdır. Yukarıda, solda verilen noktaların bir doğru ile temsil edilebileceği görülmektedir. Bir doğrunun genel denklemi $y=a_0+a_1x$ dir. Bilinmeyen 2 dir: a_0 ve a_1. Bu doğrunun denklemini, yani a_0 ve a_1 parametrelerini belirlemek, için sadece iki noktanın koordinatlarını bilmek yeter. Fakat çok sayıda, m tane koordinatları bilinen nokta vardır. Hepsinin de doğrunun denklemini sağlaması imkânsızdır. Noktaların koordinatları doğru denkleminde yerine konduğunda denklemi sağlamayacak, bir miktar hata olacaktır.

LeastSquares programı; x_i, y_i değerleri bilinen m noktayı temsil edecek $y=a_0+a_1x+a_2x^2+…+a_nx^n$ polinomunun a_0, a_1, ,a_2, …, a_n katsayılarını GAUSS'un en küçük kareler metodundaki normal denklemleri kullanarak minimum hata ile hesaplar. Teorik bağıntılar ve sayısal örnekler için için bak: Bölüm 9.

Örnek 1:

Nokta no→	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
x_i	4	6	8	10	12	14	16	18	20	22
y_i	3.3	5.3	7.4	6.9	9.0	8.6	10.8	12.4	12.0	13.2



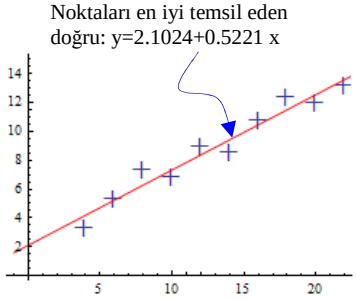
Noktalarını minimum hata ile temsil eden $y=a_0+a_1x$ doğrusunun a_0 ve a_1 katsayılarını LeastSquares programı ile bulunuz. El çözümü için bakınız: Bölüm 9, örnek 3.

Nokta sayısı m=10, polinomun derecesi n=1 (doğru denklemi) dir. n, m ve noktaların koordinatları LeastSquares ana programına aşağıdaki gibi verilmelidir:

```
n = 1:  'Polinomun derecesi
m = 10: 'Nokta sayısı
'x ordinatları
DATA 4,6,8,10,12,14,16,18,20,22
'y ordinatları
DATA 3.3,5.3,7.4,6.9,9.0,8.6,10.8,12.4,12.0,13.2
```

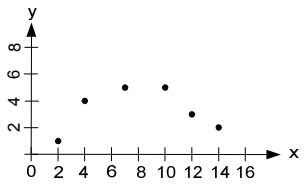
LeastSquares programının sonucu

```
C:\NANALIZ\Basic\QBasic.EXE
 1 .derece polinomun katsayıları(LeastSquares):
 2.10242424242424  .522121212121212

 1 . derece polinom:
y(x)=+ 2.10242424242424 + .522121212121212 X

Hataların karelerinin toplamı (Ypolinom-Yveri)^2= 4.66751515151515
```



Örnek 2:

Nokta no→	1	2	3	4	5	6
x_i	2	4	7	10	12	14
y_i	1	4	5	5	3	2



Noktalarını minimum hata ile temsil eden $y=a_0+a_1x+a_2x^2$ eğrisinin a_0, a_1 ve a_2 katsayılarını LeastSquares programı ile bulunuz. El çözümü için bakınız: Bölüm 9, örnek 4.

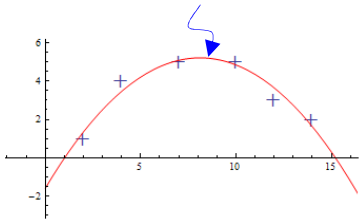
Nokta sayısı m=6, polinomun derecesi n=2 (parabol denklemi) dir. n, m ve noktaların koordinatları LeastSquares ana programına aşağıdaki gibi verilmelidir:

```
n = 2: 'Polinomun derecesi
m = 6: 'Nokta sayısı
DATA 2,4,7,10,12,14: 'x ordinatları
DATA 1,4,5,5,3,2: 'y ordinatları
```

LeastSquares programının sonucu

```
C:\NANALIZ\Basic\QBasic.EXE
 2 .derece polinomun katsayıları(LeastSquares):
-1.59881002289659   1.67906360518989  -.103499521644254

 2 . derece polinom:
y(x)=- 1.59881002289659 + 1.67906360518989 X- .103499521644254 X^ 2

Hataların karelerinin toplamı (Ypolinom-Yveri)^2= 1.00129532286326
```

Noktaları en iyi temsil eden eğri:
$y= -1.5988+1.6791x-0.1035x^2$



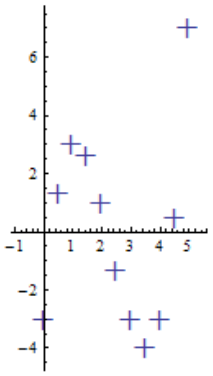
Örnek 3:

Nokta no→	1	2	3	4	5	6	7	8	9	10	11
x_i	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4	4.5	5.0
y_i	-3.0	1.3	3.0	2.6	1.0	-1.3	-3.0	-4.0	-3.0	0.5	7

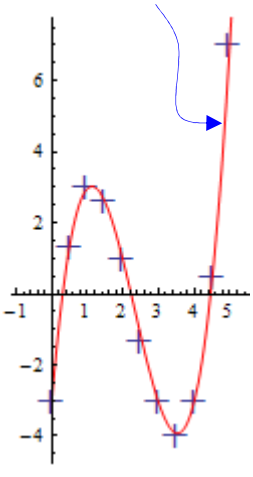
Noktalarını minimum hata ile temsil eden $y=a_0+a_1x+a_2x^2+a_3x^3$ eğrisinin a_0, a_1, a_2 ve a_3 katsayılarını LeastSquares programı ile bulunuz.

Nokta sayısı m=11, polinomun derecesi n=3 tür. n, m ve noktaların koordinatları LeastSquares ana programına aşağıdaki gibi verilmelidir:

```
n = 3: 'Polinomun derecesi
m = 11: 'Nokta sayısı
'x ordinatları
DATA 0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5
'y ordinatları
DATA -3,1.3,3,2.6,1,-1.3,-3,-4,-3,0.5,7
```



Noktaları en iyi temsil eden eğri:
$y= -3.0084+11.9666x-6.9991x^2+1.0017x^3$



LeastSquares programının sonucu

```
C:\NANALIZ\Basic\QBasic.EXE
 3 .derece polinomun katsayıları(LeastSquares):
-3.00839160839161   11.966588966589  -6.9990675990676   1.0017094017094

 3 . derece polinom:
y(x)=- 3.00839160839161 + 11.966588966589 X- 6.9990675990676 X^ 2 +
 1.0017094017094 X^ 3

Hataların karelerinin toplamı (Ypolinom-Yveri)^2= 4.75524475524493D-02
```

```
'---------------Ana program LeastSquares------------------------
' Ahmet TOPÇU, Eskişehir, 2010
' Koordinatları bilinen m notayı en iyi temsil eden ve hataların
' karelerinin toplamı en küçük olan y=a0+a1x+a2x^2+...+anx^n
' polinomunun a0, a1, a2,...,an katsayıları hesaplanır.

' Metod:
'  En küçük kareler metodu(Least Squares)-dengeleme hesabı

' Veri:
'  n: bulunacak polinomun derecesi. 0<n<m olmalıdır.
'  m: nokta sayısı. m>1 olmalıdır.
'  x(m): x1,x2,...,xm ordinatlarının depolandığı vektör
'  y(m): y1,y2,...,ym ordinatlarının depolandığı vektör

' Çıktı:
'  x(n+1): a0, a1, ..., an katsayılarının depolandığı vektör
'  hTh>=0: hataların karelerinin toplamı
'  hTh<0 : hata oluştu, hesap yapılmadı

' Çağrılan alt program: LeastSquares
'-----------------------------------------------------------------

DEFINT I-N
DEFDBL A-H, O-Z
DECLARE SUB LeastSquares (n, m, x(), y(), hTh)

n = 1: 'Polinomun derecesi
m = 10: 'Nokta sayısı
'x ordinatları
DATA 4,6,8,10,12,14,16,18,20,22
'y ordinatları
DATA 3.3,5.3,7.4,6.9,9.0,8.6,10.8,12.4,12.0,13.2

'n = 2: 'Polinomun derecesi
'm = 6: 'Nokta sayısı
'  DATA 2,4,7,10,12,14: 'x ordinatları
'  DATA 1,4,5,5,3,2: 'y ordinatları

'n = 3: 'Polinomun derecesi
'm = 11: 'Nokta sayısı
'x ordinatları
'  DATA 0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5
'y ordinatları
'  DATA -3,1.3,3,2.6,1,-1.3,-3,-4,-3,0.5,7

  CLS : ' ekranı temizle
  DIM x(m), y(m)
  FOR i = 1 TO m
   READ x(i)
  NEXT i
  FOR i = 1 TO m
   READ y(i)
  NEXT i
  CALL LeastSquares(n, m, x(), y(), hTh)
  IF hTh < 0 THEN
   PRINT "n veya m hatalı veya sayı taşması (LeastSquares)!"
   END
  END IF

' Çıktıları ekrana yaz
  PRINT n; ".derece polinomun katsayıları(LeastSquares):"
  FOR i = 1 TO n + 1
   PRINT x(i);
  NEXT

  PRINT
  PRINT
  PRINT n; ". derece polinom:"
  PRINT "y(x)=";
  FOR i = 1 TO n + 1
   a = x(i)
   IF a < 0 THEN a$ = "-" ELSE a$ = "+"
   a = ABS(a)
   k = i - 1
   IF k < 1 THEN PRINT a$; a;
   IF k = 1 THEN PRINT a$; a; "X";
   IF k > 1 THEN PRINT a$; a; "X^"; k;
  NEXT i
  PRINT
  PRINT
  PRINT "Hataların karelerinin toplamı (Ypolinom-Yveri)^2="; hTh
  PRINT

END ' LeastSquares ana sonu
```

LeastSquares ana programı

Örnek 1 in verileri

Örnek 2 nin verileri (' işareti bu satırların devre dışı olduğu anlamındadır)

Örnek 3 ün verileri (' işareti bu satırların devre dışı olduğu anlamındadır)

xi ve yi değerleri için bellekte yer ayır

xi değerlerini oku

yi değerlerini oku

LeastSquares alt programını çağır, polinomun katsayılarını hesapla

Hesaplanan katsayıları yaz

Polinomu yaz

LeastSquares alt programı

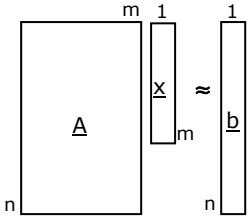
```
SUB LeastSquares (n, m, x(), y(), hTh)
'-------------------------------------------------------------
' Koordinatlari x(1),y(1),...,x(m),y(m) olarak verilmiş m
' noktayı en iyi temsil eden y(x)=a0+a1*x+a2*x^2+...+an*x^n
' polinom eğrisinin a0,a1,a2,...,an katsayıları en küçük
' kareler metodu ve ortogonal fonksiyonlar kullanılarak
' hesaplanır.

' Veri:
'  m: nokta sayısı. m>1 olmalıdır.
'  n: polinomun derecesi. 0<n<m olmalıdır.
'  x(m): x1,x2,...,xm ordinatlarının depolandığı vektör
'  y(m): y1,y2,...,ym ordinatlarının depolandığı vektör

' Çıktı:
'  x(n+1): a0, a1, ..., an katsayılarının depolandığı vektör
'  hTh>=0: hataların karelerinin toplamı
'  hTh<0 : n veya m hatalı veya sayı taşması, hesap yapılmadı

' Çağrılan alt program: Yok
'-------------------------------------------------------------
  hTh = -1
  IF m < 2 OR n <= 0 THEN EXIT SUB
  IF n >= m THEN n = m - 1
  m1 = m + 1
  n1 = n + 1
  DIM p(m1, m), a(n + 1): 'yardımcı matrisler
' huge: yuvarlama hatalarını ve sayı taşmasını önlemek için
' en büyük sayı varsayımı
  huge = 1D+30
  FOR i = 1 TO m1
   FOR j = 1 TO m
   p(i, j) = 0
   NEXT j
  NEXT i

  p(1, 1) = 1
  xsum = 0
  ysum = 0
  FOR j = 1 TO m
   xsum = xsum + x(j)
  NEXT j
  p(2, 2) = 1
  p(2, 1) = -xsum / m

  FOR i = 3 TO n1
   ii = i - 1
   ix = i - 2
   xsum = 0
   ysum = 0
   usum = 0
   zsum = 0
   FOR k = 1 TO m
    sumy = 0
    sumx = 0
    FOR L = 1 TO i
     sumy = sumy + p(ii, L) * x(k) ^ (L - 1)
     sumx = sumx + p(ix, L) * x(k) ^ (L - 1)
    NEXT L
    xsum = xsum + x(k) * sumy * sumy
    ysum = ysum + sumy * sumy
    usum = usum + x(k) * sumy * sumx
    zsum = zsum + sumx * sumx
   NEXT k
   IF ABS(ysum) > huge OR ABS(zsum) > huge THEN EXIT SUB
   b = xsum / ysum
   g = -usum / zsum
   p(i, 1) = -b * p(ii, 1) + g * p(ix, 1)
   FOR j = 2 TO i
    jj = j - 1
    p(i, j) = p(ii, jj) - b * p(ii, j) + g * p(ix, j)
   NEXT j
  NEXT i

  FOR i = 1 TO n1
   xsum = 0
   ysum = 0
   FOR k = 1 TO m
    sumz = 0
    FOR L = 1 TO i
     sumz = sumz + p(i, L) * x(k) ^ (L - 1)
    NEXT L
    xsum = xsum + y(k) * sumz
    ysum = ysum + sumz * sumz
   NEXT k
   IF ABS(ysum) > huge THEN EXIT SUB
   p(m1, i) = xsum / ysum
  NEXT i

  FOR i = 1 TO n1
   a(i) = 0
   FOR j = 1 TO i
    a(j) = a(j) + p(i, j) * p(m1, i)
   NEXT j
  NEXT i

' hataların karelerinin toplamını hesapla
   hTh = 0
   FOR i = 1 TO m
    x = x(i)
    ypolinom = 0
    FOR k = 1 TO n1
     ypolinom = ypolinom + a(k) * x ^ (k - 1)
    NEXT k
    hTh = hTh + (y(i) - ypolinom) ^ 2
   NEXT i
' Polinomun katsayılarını x vektörüne kopyala
   FOR i = 1 TO n1
    x(i) = a(i)
   NEXT i

END SUB ' LeastSquares
```

Dengeleme hesabı: QR çarpanlara ayırma metodu[1] - QR:

Dengeleme hesabında karşılaşılan

$$\underline{A}_{nxm}\underline{x}_m = \underline{b}_m$$

doğrusal denklem sisteminin denklem sayısı bilinmeyen sayısından fazladır, n>m. $\underline{A}_{nxm}$ kolon düzenli, yani Rank $\underline{A}$=m dir. Denklem sistemi şematik olarak



şeklindedir. Denklemler uyumlu olmadığından denklemi sağlayan tam doğru bir çözüm bulmak mümkün değildir. $\underline{x}$ çözümü minimum hata içerecek şekilde hesaplanır. Aşağıda Qbasic kodu verilen **QR** alt programı bu amaca yöneliktir.

Örnek 1:

Bölüm 9, örnek 2 de el çözümü verilmiş olan dengeleme probleminin

$$\underline{A}\underline{x} \approx \underline{b} \rightarrow \begin{bmatrix} 1 & \\ & 1 \\ 1 & \\ & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} h_3 \\ h_4 \end{bmatrix} \approx \begin{bmatrix} 161.678 \\ 159.277 \\ 161.685 \\ 159.265 \\ 2.410 \end{bmatrix}$$ denklem sisteminde n=5 denklem ve m=2 bilinmeyen vardır,

n>m dir. n, m, $\underline{A}$ ve $\underline{b}$ nin değerleri

```
DATA 5,2 : ' n ve m nin değeri
' a matrisi:
DATA 1,0
DATA 0,1
DATA 1,0
DATA 0,1
DATA 1,-1
' b vektörü:
DATA 161.678,159.277,161.685,159.265,2.410
```

satırları ile **QR** programına verilerek $\underline{x} = \begin{bmatrix} h_3 \\ h_4 \end{bmatrix}$ bilinmeyen vektörü hesaplanmıştır.

QR programının sonucu

```
C:\NANALIZ\Basic\QBasic.EXE

Denklem sisteminin çözümü (QR):
 161.681375   159.271125

Hataların karelerinin toplamı= 9.66250000001067D-05
```

$$\rightarrow \underline{x} = \begin{bmatrix} h_3 \\ h_4 \end{bmatrix} = \begin{bmatrix} 161.681375 \\ 159.271125 \end{bmatrix}$$

Hataların karelerinin toplamı=0.000097

[1] Teori ve sayısal örnekler için bak: Bölüm 9

Örnek 2:

QR programı en uygun eğri uydurma problemlerinin çözümü için de kullanılabilir. Bölüm 9, örnek 4 de el çözümü verilmiş olan en uygun eğrinin bulunması probleminin

$$\underline{A}x \approx \underline{b} \rightarrow \left[\begin{array}{ccc} 1 & 2 & 4 \\ 1 & 4 & 16 \\ 1 & 7 & 49 \\ 1 & 10 & 100 \\ 1 & 12 & 144 \\ \hline 1 & 14 & 196 \end{array}\right] \left[\begin{array}{c} a_0 \\ a_1 \\ a_2 \end{array}\right] \approx \left[\begin{array}{c} 1 \\ 4 \\ 5 \\ 5 \\ 3 \\ 2 \end{array}\right]$$

denklem sisteminde n=6 denklem ve m=3 bilinmeyen vardır, n>m dir. n, m, $\underline{A}$ ve $\underline{b}$ nin değerleri

```
DATA 6,3: ' n ve m nin değeri
' a matrisi:
DATA 1,2,4
DATA 1,4,16
DATA 1,7,49
DATA 1,10,100
DATA 1,12,144
DATA 1,14,196
' b matrisi:
DATA 1,4,5,5,3,2
```

satırları ile **QR** programına verilerek $y=a_0+a_1x+a_2x^2$ eğrisinin a_0, a_1 ve a_2 katsayıları hesaplanmıştır.

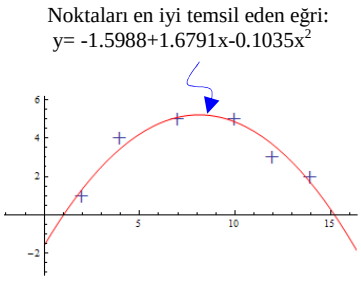
QR programının sonucu

```
C:\NANALIZ\Basic\QBasic.EXE
Denklem sisteminin çözümü (QR):
-1.59881002289657  1.67906360518989  -.103499521644254

Hataların karelerinin toplamı= 1.00129532286326
```

$a_0 = -1.5988$

$a_1 = 1.6791$

$a_2 = -0.1035$



Noktaları en küçük hata ile temsil eden eğrinin denklemi:

$y= -1.5988+1.6791x-0.1035x^2$

olur.

```
'------------------- Ana program QR-----------------------------------
' Dr. Ahmet TOPÇU, ESKİŞEHİR, 2010
' Denklem sayısı bilinmeyen sayısından büyük olan doğrusal
' a(n,m)x(m)=b(n) denklem sistemi hataların karelerinin toplamı
' minimum  olacak şekilde QR çarpanlara ayırma metodu ile çözülür.
' n>=m ve a nın kolonları doğrusal bağımsız olmalıdır, yani rank(a)=m.

' Bu program en uygun eğri uydurma problemleri için de kullanılabilir.

' Kullanılan metot: QR çarpanlara ayırma yöntemi
' Çağrılan alt programlar: QR
'-----------------------------------------------------------------------
DATA 5,2 : ' n ve m nin değeri
' a matrisi:
DATA 1,0
DATA 0,1
DATA 1,0
DATA 0,1
DATA 1,-1
' b vektörü:
DATA 161.678,159.277,161.685,159.265,2.410

'DATA 6,3: ' n ve m nin değeri
' a matrisi:
'DATA 1,2,4
'DATA 1,4,16
'DATA 1,7,49
'DATA 1,10,100
'DATA 1,12,144
'DATA 1,14,196
' b matrisi:
'DATA 1,4,5,5,3,2

DEFINT I-N
DEFDBL A-H, O-Z

DECLARE SUB QR (a(), b(), n, m, x(), hTh)

  CLS

  READ n, m
  DIM a(n, m), x(m), b(n)

' A yı oku:
  FOR i = 1 TO n
   FOR j = 1 TO m
    READ a(i, j)
   NEXT j
  NEXT i
' b yi oku:
  FOR i = 1 TO n
   READ b(i)
  NEXT i

  CALL QR(a(), b(), n, m, x(), hTh)

  IF hTh < 0 THEN
   PRINT "Çözüm bulunamadı(QR) !"
  END
  END IF

  PRINT "Denklem sisteminin çözümü (QR):"
  FOR i = 1 TO m
   PRINT x(i);
  NEXT i
  PRINT
  PRINT
  IF n > m THEN PRINT "Hataların karelerinin toplamı="; hTh

END  ' QR ana
```

QR alt programı

```
SUB QR (a(), b(), n, m, x(), hTh)
'--------------------------------------------------------------------------
' Solves the overdetermined linear system a(n,m)x(m)=b(n) using
' the QR factorization with Householder transformation, where n>=m.
' The solution x(m) is found in least squares sens, so that Euclidean
' norm ||Ax-b|| will be minimum.
' Method used:
'   orthogonalization: Householder transformation
'   Decomposition: QR algorithm
'   Coefficient matrix a(n,m) will be transform to A=QR, where Q is an
'   othogonal matrix and R is an upper triangular non-singular matrix.
' Constraints: n>=m and rank(a)=m
' This program can be used for curve fitting problems olso.
' input a(n,m): coefficient matrix
'   b(n): right side vector with length n
'      n: number of equations
'      m: number of uknowns
' output:
'   a(n,m): A and R matricies.
'   x(m)  : solution vector.
'   hTh<0 : the condition n>=m not held or R is singular, so no solution
'   hTh>=0: sum of squares of errors, solution found
' Subroutines called: none
' Fortran code: www.stat.berkeley.edu/~stark/code/index.htm
' Original file name: qr.f
'--------------------------------------------------------------------------
   hTh = -1
' Check if the system is not overdetermined
   IF n < m THEN EXIT SUB
   Zero = 1E-30: 'assumed smallest number.
   FOR j = 1 TO m
' Find constants for Housholder rotation and diagonal entry
    sq = 0
    FOR i = j TO n
     sq = sq + a(i, j) ^ 2
    NEXT i
' Check if column is linear dependent
    IF sq < Zero THEN EXIT SUB
    Sign = SQR(sq)
    IF a(j, j) >= 0 THEN Qv1 = Sign ELSE Qv1 = -Sign
    Qv1 = -Qv1
    u1 = a(j, j) - Qv1
    a(j, j) = Qv1
    j1 = j + 1
' Rotate remaining columns of sub-matrix
    FOR jj = j1 TO m
     Dot = u1 * a(j, jj)
     FOR i = j1 TO n
      Dot = Dot + a(i, jj) * a(i, j)
     NEXT i
     Con = Dot / ABS(Qv1 * u1)
     FOR i = j1 TO n
      a(i, jj) = a(i, jj) - Con * a(i, j)
     NEXT i
      a(j, jj) = a(j, jj) - Con * u1
    NEXT jj
' Rotate b vector
    Dot = u1 * b(j)
    FOR i = j1 TO n
     Dot = Dot + b(i) * a(i, j)
    NEXT i
    Con = Dot / ABS(Qv1 * u1)
    b(j) = b(j) - Con * u1
    FOR i = j1 TO n
     b(i) = b(i) - Con * a(i, j)
    NEXT i
   NEXT j
' Solve triangular system by back-substitution
   FOR i = 1 TO m
    x(i) = 0
   NEXT i
   FOR ii = 1 TO m
    i = m - ii + 1
    Sum = b(i)
    FOR j = i + 1 TO m
     Sum = Sum - a(i, j) * x(j)
    NEXT j
' Check if trianguler system is singular
    IF ABS(a(i, i)) < Zero THEN EXIT SUB
    x(i) = Sum / a(i, i)
   NEXT ii
' Find residual in overdetermined case
   hTh = 0
   FOR i = m + 1 TO n
    hTh = hTh + b(i) ^ 2
   NEXT i

END SUB ' end of QR
```

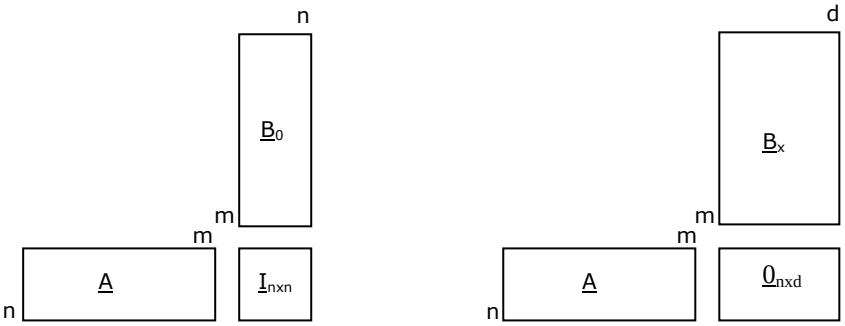
Sağ ters matris ve çekirdek hesabı-BoBx[1]:

Satır düzenli $\underline{A}_{nxm}$ matrisinde n≤m olmak üzere

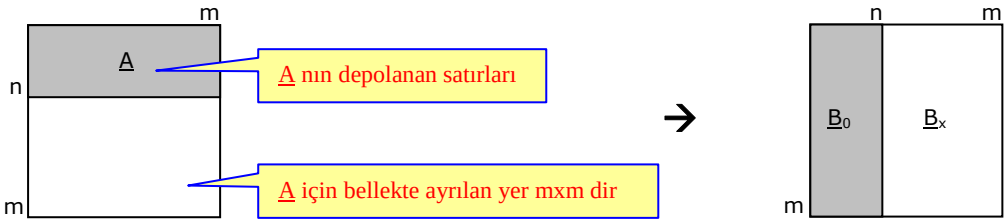
$\underline{A}\ \underline{B}_x = \underline{0}$

$\underline{A}\ \underline{B}_0 = \underline{I}$

Koşullarını sağlayan $\underline{B}_x$ matrisine $\underline{A}$ nın çekirdeği(Kern, null space), $\underline{B}_o$ matrisine de $\underline{A}$ nın sağ ters matrisi(right inverse) denir. Denklem sayısı bilinmeyen sayısından az olan denklem sistemlerinin homojen ve inhomojen çözümünün doğrudan bulunmasını sağlar. $\underline{B}_x$ homojen çözüme, $\underline{B}_o$ inhomojen çözüme karşılık gelir. n<m durumunda d=m-n kolon doğrusal bağımlıdır. $\underline{B}_o$ mxn boyutlu, $\underline{B}_x$ mxd boyutludur. Şematik olarak:



BoBx alt programı bu amaca yöneliktir. $\underline{A}$ matrisi için çağıran programda mxm boyutunda yer ayrılmış ve $\underline{A}$ nın n satırı depolanmış olmalıdır. $\underline{B}_o$ ve $\underline{B}_x$ matrisleri $\underline{A}$ nın üzerine depolanır. İlk n kolon $\underline{B}_o$ sonraki d=m-n kolon $\underline{B}_x$ dir.



n=m durumunda $\underline{B}_o=\underline{A}^{-1}$ dir, bu durumda $\underline{B}_x$ tanımsızdır.

iHata=0 olarak dönerse $\underline{B}_o$ ve $\underline{B}_x$ hesaplanmıştır, aksi halde hesap yarıda kesilmiştir.

Örnek:

$$\underline{A}_{8x10} = \begin{bmatrix} 0 & 0 & -0.7071 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -0.7071 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -0.7071 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0.7071 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.7071 & 0 & 0 & 0 & -0.7071 & -1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0 & 0 & 1 & 0.7071 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.7071 & 0 & 0 & 0 & -0.7071 & -1 \\ 0 & 0 & 0 & 0 & -0.7071 & -1 & 0 & 0 & -0.7071 & 0 \end{bmatrix}, \quad \underline{B}_o = ?, \quad \underline{B}_x = ?$$

[1] Teori ve sayısal örnekler için bak: bölüm 10

BoBx programının sonucu

```
C:\Basic\QBasic.EXE
Doğrusal bağımlı kolonların numarası(BoBx):
 9  10

Çekirdek Bx in kolonları(BoBx):
-.7071 -.7071  1 -.7071  1 -.7071  0  0  0  0
 0  0  0  0  0 -.7071  1 -.7071  1 -.7071

Sağ ters matris Bo ın kolonları(BoBx):
 0  0  0 -1  0  0  0  0  0 -1
 0  0 -1.41422712487626  1  0  1  0 -1 -1.41422712487626  2
 0 -1  0  0  0  0  0 -1  0  0
 1  0 -1.41422712487626  1  0  1  0 -1 -1.41422712487626  2
 0  0  0  0  0  0  0 -1  0  0
 0  0  0  0  0  1  0  0 -1.41422712487626  1
 0  0  0  0  0  0  0  0  0 -1
 0  0  0  0  0  0  0  0 -1.41422712487626  1
```

Çözüm:

$$\underline{B}_x = \begin{bmatrix} -0.7071 & 0 \\ -0.7071 & 0 \\ 1 & 0 \\ -0.7071 & 0 \\ 1 & 0 \\ -0.7071 & -0.7071 \\ 0 & 1 \\ 0 & -0.7071 \\ 0 & 1 \\ 0 & -0.7071 \end{bmatrix} \qquad \underline{B}_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.4142 & 0 & -1.4141 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1.4142 & 0 & -1.4141 & 0 & -1.4142 & 0 & -1.4142 \\ -1 & 2 & 0 & 2 & 0 & 1 & -1 & 1 \end{bmatrix}$$

Ana program BoBx

```
' ---------------------Ana program BoBx---------------------------
' Dr. ahmet TOPÇU, Osmangazi Üniversitesi, ESKİŞEHİR, 1995
' a(n,m) matrisinin sağ ters matris ve çekirdek hesabı

' Veri:
'  a(n,m): satır düzenli matris. mxm boyutlu tanımlanmalıdır
'          a nın elemanları ilk n satır ve m kolona depolanmalıdır.
'     n,m: matrisin satır ve kolon sayısı, n<=m olmalıdır.

' Çıktı:
'  Bo(m,n): a nın ilk n kolonunda depoludur.
'  Bx(m,d): a nın son d kolonunda depoludur. d=m-n dir.
'  n=m durumunda Bx tanımsızdır, hesaplanmaz ve Bo a nın tersidir.

' Çağrılan alt programlar: BoBx
'-----------------------------------------------
  DATA 8, 10: ' a nın satır ve kolon sayısı
' Matrix a:
  DATA  0, 0,-.7071,    -1,     0, 0,    0,  0, 0,     0
  DATA -1, 0,-.7071,     0,     0, 0,    0,  0, 0,     0
  DATA  0,-1,    0,      0,-.7071, 0,    0,  0, 0,     0
  DATA  1, 0,    0,      0, .7071, 0,    0,  0, 0,     0
  DATA  0, 1, .7071,     0,     0, 0,-.7071,-1, 0,     0
  DATA  0, 0, .7071,     0,     0, 1, .7071, 0, 0,     0
  DATA  0, 0,     0,     1, .7071, 0,     0, 0,-.7071,-1
  DATA  0, 0,     0,     0,-.7071,-1,     0, 0,-.7071, 0

  DEFINT I-N
  DEFDBL A-H, O-Z
  DECLARE SUB BoBx (n, m, a(), iSp(), iHata)

  CLS
  READ n, m
  DIM a(m, m)
  DIM iSp(m)

  FOR i = 1 TO n
   FOR j = 1 TO m
    READ a(i, j)
   NEXT j
  NEXT i

  CALL BoBx(n, m, a(), iSp(), iHata)
```

A nın satır ve kolon sayısı

A nın satırları

A nın satır ve kolon sayısını oku

A için bellekte mxm boyutlu yer ayır

A yı oku

BoBx alt programını çağır

Devamı var

```
  IF iHata = 1 THEN
   PRINT "Matrisin satırları düzensiz(BoBx)"
   END
  END IF

   IF m > n THEN
    PRINT "Doğrusal bağımlı kolonların numarası(BoBx):"
    FOR i = n + 1 TO m
     PRINT iSp(i);
    NEXT i
    PRINT
    PRINT

    PRINT "Çekirdek Bx in kolonları(BoBx):"
    FOR i = n + 1 TO m
     FOR j = 1 TO m
      PRINT a(j, i);
     NEXT j
     PRINT
    NEXT i
   END IF
   PRINT

  PRINT "Sağ ters matris Bo ın kolonları(BoBx):"
  FOR i = 1 TO n
   FOR j = 1 TO m
    PRINT a(j, i);
   NEXT j
   PRINT
  NEXT i
  PRINT

END 'BoBx ana
```

Devamı

Doğrusal bağımlı kolonların numaralarını yaz

Bx in kolonlarını yaz

Bo ın kolonlarını yaz

BoBx alt programı

```
SUB BoBx (n, m, a(), iSp(), iHata)
' Sağ ters matris ve çekirdek hesabı
'-------------------------------------------------------------------------
' Dr. ahmet TOPÇU, Osmangazi Üniversitesi, ESKİŞEHİR, 1995
' a(n,m) matrisinin Bo sağ ters matrisi ve Bx çekirdeği GAUSS-JORDAN
' tekniği ile a*Bo=I ve a*Bx=0 koşulları sağlanacak şekilde hesaplanır.
' a(n,m) çağıran programda mxm boyutlu olak tanımlanmış ve n tane satırı
' depolanmış olmalıdır.

' A nın satır düzenli olduğu ve n<=m olduğu varsayılmaktadır.

' BoBx çağrıldıktan sonra a nın ilk n kolonu Bo(m,n) sağ ters matrisini,
' son d=m-n kolonu Bx(m,d) çekirdeğini içerir.
' n=m durumunda sağ ters matris a nın tersine eşittir, Bx tanımsızdır.

' Satırlar doğrusal bağımlı ise hesap kesilir iHata<>0 döner, Bo ve Bx
' hesaplanmaz.
' iSp(m) vektörü çağıran programda tanımlanmış olmalıdır.
' iSp de kolon numaraları saklanır, son d kolon doğrusal bağımlılardır.

' Çağrılan alt program: yok
'-------------------------------------------------------------------------
  iHata = 0
' Machep
  Eps = 1
  DO
   Eps = Eps / 2
   s = 1 + Eps
  LOOP UNTIL s <= 1
  Eps = 2 * Eps
' Zero: Sıfır varsayılacak değer
  IF m - n < 0 THEN iHata = 1: EXIT SUB
' Norm
  zero = 0
  FOR j = 1 TO m
   iSp(j) = j
   FOR i = 1 TO n
    Norm = ABS(a(i, j))
    IF Norm > zero THEN zero = Norm
   NEXT i
  NEXT j
  zero = zero * Eps

  FOR i = 1 TO n
' satırda pivot ara
  Pivot = 0
   FOR k = i TO m
    at9 = ABS(a(i, k))
    IF at9 > Pivot THEN
     iV = k: ' pivot column
     Pivot = at9: ' pivot eleman
    END IF
   NEXT k
```

Devamı var

Devamı

```
' Kolon doğrusal bağımlı mı kontrol
   IF Pivot < zero THEN iHata = 1: EXIT SUB
    IF iV > i THEN
' i ve iSv nolu kolonları değiş
    iSv = iSp(i)
    iSp(i) = iSp(iV)
    iSp(iV) = iSv
    at9 = -1 / a(i, iV)
    a(i, iV) = -1
    FOR k = 1 TO n
     agrz = a(k, i)
     a(k, i) = a(k, iV) * at9
     a(k, iV) = agrz
    NEXT k
   ELSE
    at9 = -1 / a(i, i)
    a(i, i) = -1
    FOR k = 1 TO n
     a(k, i) = a(k, i) * at9
    NEXT k
   END IF

' Transformasyon matrisi ile çarp
   FOR k = 1 TO m
    IF k <> i THEN
     at9 = a(i, k)
     a(i, k) = 0
     FOR j = 1 TO n
      a(j, k) = a(j, k) + a(j, i) * at9
     NEXT j
    END IF
   NEXT k
  NEXT i

' İşaret değiş, nxm boyutlu alanı m*m boyuta dönüştür
   FOR i = 1 TO n
    FOR j = n + 1 TO m
     a(i, j) = -a(i, j)
    NEXT j
   NEXT i
   FOR i = n + 1 TO m
    FOR j = 1 TO m
     a(i, j) = 0
    NEXT j
    a(i, i) = 1
   NEXT i

' Satırlara yer değiştir
  FOR j = 1 TO m
   iSv = iSp(j)
   IF j <> iSv THEN
    FOR k = j + 1 TO m
     IF iSp(k) = j THEN EXIT FOR
    NEXT k
    iSp(k) = iSv
    FOR i = 1 TO m
     at9 = a(j, i)
     a(j, i) = a(k, i)
     a(k, i) = at9
    NEXT i
   END IF
  NEXT j

END SUB  'BoBx
```