L₁ Norm Solution of Overdetermined System of Linear Equations

Bijan BIDABAD

Keywords: L₁ norm, Regression, Algorithm, Computer program

Abstract

In this paper three algorithms for weighted median, simple linear and multiple m parameters L₁ norm regressions are introduced. The corresponding computer programs are also included.

Introduction

L₁ norm criterion is going to find its place in scientific analysis. Since it is not computationally comparable with other criteria such as L₂ norm, it needs more work to make it a hand tool. The closed form of the solution of the L₁ norm estimator has not been derived yet, and therefore, makes further inferences of the properties of this estimator difficult. Any attempt to give efficient computational algorithms which may introduce significant insight into the different characteristics of the problem is desirable. In this regard, Bidabad (1989a,b) gives a general procedure to solve L₁ norm linear regression problem. The proposed algorithms are based on a special descent method and use discrete differentiation technique. Primary designs of the algorithms have been discussed by Bidabad (1987a,b,88a,b). By manipulating the algorithms, more efficient ones were introduced by Bidabad (1989a,b) which has been shown to have better performance than other existing algorithms.

Consider the following regression model,

\[ y_i = \sum_{j=1}^{m} \beta_j x_{ij} + u_i \quad i = 1, \ldots, n \]  \hspace{1cm} (1)

where \( \beta_j, j = 1, \ldots, m \) are unknown population parameters to be estimated, \( y_i, x_{ij} \) and \( u_i \) are dependent, independent and random error variables respectively. We wish to estimate \( \beta_j \)'s by minimizing sum of absolute errors given by the following expression:

\[ S = \sum_{i=1}^{n} \left| u_i \right| = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{m} \beta_j x_{ji} \right| \]  \hspace{1cm} (2)

where \( \beta_j \) is the estimated value of \( \beta_j \). When \( m = 1 \) we are confronted with weighted median problem.

---

1 Research professor of economics, Monetary and Banking Research Academy, Central Bank of Iran, Pasdaran, Zarabkhaneh, Tchran, 16619, Iran. bijan_bidabad@msn.com

I should express my sincere thanks to Professor Yádolah Dodge from Neuchatel University of Switzerland who taught me many things about L₁ norm when he was my Ph.D. dissertation advisor.
Weighted median computation (restricted one parameter model)

Let us now consider a simple restricted linear model in which \( m = 1 \) namely,
\[
y_i = \beta_1 x_{i1} + u_i
\]
For the model given by (3), the L1 norm objective function \( S \) to be minimized will be,
\[
S = \sum_{i=1}^{n} \left| \frac{y_i - \beta_1 x_{i1}}{x_{i1}} \right| = \sum_{i=1}^{n} \left| \frac{y_i}{x_{i1}} - \beta \right|
\]

Two series of computations are necessary to compute weighted median. One sorting algorithm is essential to sort the ratio array \( \left( \frac{y_i}{x_{i1}} \right) \) and restoring the corresponding subscripts for second part of calculation to find the left and right weights \( \left( \frac{x_{i1}}{4} \right) \) sequences.

Efficient sorting algorithms exist for the first part of the computation. The algorithms 'quicksort' of Hoare (1961,62), 'quickersort' of Scowen (1965) and 'sort' of Singleton (1969) have desirable performances and efficiencies. For the second part of the computation there is no special purpose procedure, but Bloomfield and Steiger (1980) used the partial sorting of Chambers (1971) to give an efficient way to combine the two steps of sorting and finding the optimal observation. The superiority of this procedure is in sorting the smaller segments of the array rather than all its elements. With some modification, this procedure is used by Bidabad (1989a,b). The procedure can be stated as the following function.

**FUNCTION LWMED (n,ys,w,l)**

Step 0) Initialization.
- Real: ys(n), w(n).
- Integer: l(n), hi.
- Set: ii=0, shi=0, slo=0, sz=0, sp=0, sn=0.

Step 1) Compute left, middle and right sum of weights.
- Do loop for \( i=1,n \): \( w(i) = w(i) \); if \( ys(i)<0 \), then \( sn=sn+w(i) \), if \( ys(i)>0 \), then \( sp=sp+w(i) \), if \( ys(i)=0 \) then \( sz=sz+w(i) \); end do.
- If \( shi<slo \) then go to step 2.b, otherwise go to step 2.a.

Step 2) Assign subscripts for arrays.
- a. Let: \( shi=0 \).
- Do loop for \( i=1,n \): if \( ys(i)\leq0 \) go to continue, otherwise \( ii=ii+1 \), \( l(ii)=i \), continue, end do.
- Go to step 2.c.
- b. Let: \( slo=0 \).
- Do loop for \( i=1,n \): if \( ys(i)>0 \) go to continue, otherwise \( ii=ii+1 \), \( l(ii)=i \), continue, end do.
- c. Let: \( lo=1 \), \( hi=ii \).

Step 3) Check for solution.
- If \( hi>lo+1 \) then go to step 4, otherwise \( lwmed=l(lo) \).
- If \( lo=hi \) return, otherwise if \( ys(l(lo))\leq ys(l(hi)) \) go to step 3.a, otherwise \( lt=l(lo) \), \( l(lo)=l(hi) \), \( l(hi)=lt \), \( lwmed=l(lo) \).
- a. If \( shi+w(l(hi))>slo+w(l(lo)) \) then set \( lwmed=l(hi) \), otherwise return.

Step 4) Divide the string into two halves then sort.
- Set: \( mid=(lo+hi)/2 \), \( lop=lo+1 \), \( lt=l(mid) \), \( l(mid)=l(lop) \), \( l(lop)=lt \).
- a. If \( ys(l(lop))\leq ys(l(hi)) \) then go to step 4.b, otherwise \( lt=l(lop) \), \( l(lop)=l(hi) \), \( l(hi)=lt \).
- b. If \( ys(l(lo))\leq ys(l(hi)) \) then go to step 4.c, otherwise \( lt=l(lo) \), \( l(lo)=l(hi) \), \( l(hi)=lt \).
- c. If \( ys(l(lop))\leq ys(l(lo)) \) then go to step 5, otherwise \( lt=l(lop) \), \( l(lop)=l(lo) \), \( l(lo)=lt \).

Step 5) Compute the accumulation of weights.
- Let: \( lwmed=l(lo) \), \( i=lop \), \( j=hi \), \( xt=ys(lwmed) \), \( tlo=slo \), \( thi=shi \).
- a. Set: \( tlo=tlo+w(l(i)) \), \( i=i+1 \).
If $y_i(l(i)) < x_t$ then go to step 5.a, otherwise go to step 5.b.

b. Let: $\theta_i = \theta_i + w(l(j))$, $j = j - 1$.

If $y_i(l(j)) > x_t$ then go to step 5.b, otherwise if $j \leq i$ then go to step 6, otherwise $l_t = l(i)$, $l(i) = l(j)$, $l(j) = l_t$, go to step 5.a.

Step 6) Test for solution.

Let: $test = w(lwmed)$.

If $i > j$ then go to step 6.a, otherwise $test = test + w(l(i))$, $i = i + 1$, $j = j - 1$.

a. If $test \geq |\theta_t - lo|_t$ then return, otherwise, if $lo > \theta_t$ then step 6.b, otherwise $slo = lo + test$, $lo = i$, go to step 3.

b. Let: $sh_i = \theta_t + test$, $lo = lop$, $hi = j$.

Go to step 3.

END

Unrestricted simple linear regression

Let us now consider a simple unrestricted linear model in which $m = 2$ and $x_{1i} = 1$ for all $i = 1, \ldots, n$; namely,

$$y_i = \beta_1 + \beta_2 x_{2i} + u_i \quad (5)$$

For the model given in (5), the $L_1$ norm objective function $S$ to be minimized will be,

$$S = \sum_{i=1}^{n} |y_i - \beta_1 - \beta_2 x_{2i}| \quad (6)$$

PROGRAM BL1S

Step 0) Initialization.

Parameter: $n$.
Real: $y(n)$, $x2(n)$, $z(n)$, $w(n)$.
Integer: $l(n)$.
Set: $k1 = $ arbitrary, $k1r = 0$, $k1s = 0$, $iter = 0$.
Read $(y(i), x2(i), i = 1, n)$

Step 1) Compute weights and ratios.

Do loop for $i = 1, k1-1$: $w(i) = x2(i) - x2(k1)$, $w(i) = y(i) - y(k1) / w(i)$, end do.
Set: $w(k1) = 0$, $z(k1) = 0$.
Do loop for $i = k1 + 1, n$: $w(i) = x2(i) - x2(k1)$, $z(i) = (y(i) - y(k1)) / w(i)$ end do.
Set: $iter = iter + 1$.

Step 2) Compute weighted median.

Let: $lm = LWMED(n, z, w, l)$.

Step 3) Check for optimality.

Set: $k1s = k1r$, $k1r = k1$.
If $lm = k1s$ then go to step 4, otherwise $k1 = lm$.
Go to step 1.

Step 4) Compute the solution.

Let $b2 = z(lm)$, $b1 = y(k1) - b2 * x2(k1)$.
Print b1, b2, k1, lm, iter.
stop.

END

General linear model

For the general $m$ parameter model, the following algorithm is proposed.

PROGRAM BL1

Step 0) Initialization.
Parameter: \( n, m, m_1 = m - 1, m_2 = m - 2 \).
Real: \( y(n), x(n, m_1), y_{sk}(m_1), y_{w}(n), x_{kw}(n), w(n), y_{s}(n), x_{s}(n, m_1), b(m), \)
\( x(w(n, m_2), y_{sol}(m_1), x_{sol}(m_1, m_1) \).
Integer: \( l(n), kk(m_1) \).
Common: \(/c1/i1,i2\).
Read: \((y(i), (x(i,j), j=1, m_1), i=1, n)\).
Let: iter=0, kr=0, \( mm=1, (kk(j)=\text{arbitrary}, j=1, m_1) \).

**Step 1)** Refill working arrays.
Do loop for \( i=1, n \): \( y_{sk}(i)=y(i) \), do loop for \( j=1, m_1 \): \( x_{sk}(j)=x(i,j) \), end do, end do.

**Step 2)** Store weights and ratios for next iteration.
Do loop for \( i=1, n \): \( w(i)=x_{kw}(i), y_{w}(i)=y_{w}(i) \), do loop for \( j=1, m_1 \): \( x_{w}(i,j)=x_{w}(i,j) \), end do, end do.

**Step 3)** Compute the arguments for weighted median.

a. Set: \( jj=mm, k=kk(jj), y_{sk}=y_{sk}(j), i_1=1, i_2=k-1 \).
   Do loop for \( j=jj, m_1 \): \( x_{sk}(j)=x_{sk}(j,k) \), end do.

b. Do loop for \( j=jj, m_1 \): call COL1 \((x_{sk}(j), x(1,j))\) end do.
   Call COL2 \((y_{sk}, jj, w, y_{sk}, x(1,j))\).
   If \( i_2=n \) go to step 3.c; otherwise set: \( i_1=k+1, i_2=n \), go to step 3.b.

c. Set: \( w(k)=0 \).
   If \( jj=m_1 \) go to step 4; otherwise \( i_1=1, i_2=k-1 \), go to step 3.d.

b. Do loop for \( j=jj+1, m_1 \): call COL3 \((x_{sk}(j), x_{sk}(1,j))\) end do.
   If \( i=n \) go to step 3.e; otherwise set \( i_1=k+1, i_2=n \), go to step 3.d.

b. Do loop for \( i=jj, m_1 \): \( x_{w}(i,j)=x_{w}(i,j) \), end do; end do.
   Set: \( jj=jj+1, \) go to step 3.

**Step 4)** Compute the weighted median.
Set: \( y_{sk}=0, iter=iter+1, lm=LWMED(n, y_{sk}, w, l) \).

**Step 5)** Test for optimality.
If \( lm=kr \) go to step 5.b; otherwise \( i_{opt}=0 \) go to step 5.a.

a. If \( mm=m_1 \) set \( mm=1, kr=kk(mm), kk(mm)=lm, \) go to step 1; otherwise set \( mm=mm+1, kr=kk(mm), kk(mm)=lm, \) go to step 2.

b. Set: \( i_{opt}=i_{opt}+1, \)
   If \( i_{opt} \neq m_1 \) go to step 5.a, otherwise go to step 6.

**Step 6)** Compute the solution.
Set: \( b(m)=y_{sk}(lm) \).
Do loop for \( i=1, m_1 \): \( y_{sol}(i)=y(kk(i)) \); do loop for \( j=1, m_1 \): \( x_{sol}(i,j)=x(kk(i),j) \), end do; end do.
Set: \( jj=1 \).
a. Set: \( y_{sk}=y_{sol}(jj) \).
   Do loop for \( j=jj, m_1 \): \( x_{sk}(j)=x_{sol}(jj,j) \), end do.
   Do loop for \( i=jj, m_1 \): if \( i=jj \) go to continue; otherwise \( y_{sol}(i)=y_{sol}(i)-y_{sk} \), do loop for \( j=jj, m_1 \): \( x_{sol}(i,j)=x_{sol}(i,j)-x_{sk}(j) \), end do; set \( y_{sol}(i)=y_{sol}(i)/x_{sol}(i,j) \), continue, end do.

b. Do loop for \( i=jj, m_1 \): if \( i=jj \) go to continue, otherwise, do loop for \( j=jj+1, m_1 \):
   \( x_{sol}(i,j)=x_{sol}(i,j)/x_{sol}(i,j) \), end do; continue; end do.

c. If \( jj=m_2 \) go to step 6.d; otherwise go to step 6.a.

b. Do loop for \( i=1, m_2 \): \( k=m-i, s=y_{sol}(k) \); do loop for \( j=k, m_1 \), \( s=s-b(j+1)*x_{sol}(k,j) \) end do, \( b(k)=s \), end do.
Set: \( s=y(kk(1)) \).
Do loop for \( j=1, m_1 \): \( s=s-b(j+1)*x_{sol}(k(1),j) \), \( b(1)=s \), end do.
Print: \( ((b(j), j=1, m_1), (kk(j), j=1, m_1), lm, iter) \).
Stop.

END

The major portion of computation in this program is transformation of two dimensional arrays. Passing columns of these arrays to other subroutines which involves only one dimensional arrays saves the time of computation (see, Barrodale and Roberts (1974)). Subroutine COL1, COL2 and COL3 have been coded to do this task for subtraction, multiplication and division, and for only division respectively. Function LWMED which is used to compute weighted median has been introduced in section 2.1.

SUBROUTINE COL1(v1,v2)
Step 0) Initialization
   Real: v2(1).
   Common /c1/i1,i2.
Step 1) Subtraction.
   Do loop for i=i1,i2: v2(i)=v2(i)-v1, end do.
   Return.
END

SUBROUTINE COL2(ysk,jj,v1,ys,v2)
Step 0) Initialization.
   Real: v1(1),v2(1),ys(1).
   Common /c1/i1,i2.
Step 1) Compute weights and ratios.
   If jj<1 go to step 1.a.; otherwise do loop for i=i1,i2: v1(i)=v2(i),
   ys(i)=(ys(i)-ysk)/v2(i).
   Return.
   a. Do loop for i=i1,i2: v1(i)=v1(i)*v2(i), ys(i)=(ys(i)-ysk)/v2(i), end do.
   Return.
END

SUBROUTINE COL3(v1,v2)
Step 0) Initialization.
   Real: v1(1),v2(1),ys(1).
   Common /c1/i1,i2.
Step 1) Division.
   Do loop for i=i1,i2: v1(i)=v1(i)/v2(i), end do.
   Return.
END

Computer programs
FUNCTION LWMED(N,YS,W,L)
C N Number of observations (input).
C YS(I) The array to be sorted (yi/xi1) (input).
C W(N) The weight array (xi1) (input).
C L The index of location of weighted median in the unsorted arrays (output).
REAL YS(N),W(N)
INTEGER L(N),HI
II=0
SHI=0.
SLO=0.
SZ=0.
SP = 0.
SN = 0.
DO 4 I = 1, N
  W(I) = ABS(W(I))
  IF(YS(I)) 3 , 2 , 1
1  SP = SP + W(I)
  GO TO 4
2  SZ = SZ + W(I)
  GO TO 4
3  SN = SN + W(I)
4  CONTINUE
SHI = SP + SZ
SLO = SN + SZ
IF(SHI .LE. SLO) GO TO 6
SHI = 0.
DO 5 1 = 1, N
  IF(YS(I).LE.0.) GO TO 5
II = II + 1
L(II) = I
5  CONTINUE
GO TO 8
6  SLO = 0.0
DO 7 1 = 1, N
  IF(YS(I).GT.0.) GO TO 7
II = II + 1
L(II) = I
7  CONTINUE
8  LO = 1
HI = II
10  IF(HI.GT.LO+1) GO TO 30
  LWMED = L(LO)
  IF(LO.EQ.HI) RETURN
  IF(YS(L(LO)), LE, YS(L(HI))) GO TO 20
  LT = L(LO)
  L(LO) = L(HI)
  L(HI) = LT
  LWMED = L(LO)
20  IF(SHI+W(L(HI)), GT, SLO+W(L(LO))) LWMED = L(HI)
RETURN
30  MID = (LO+HI)/2
  LOP = LO+1
  LT = L(MID)
  L(MID) = L(LOP)
  L(LOP) = LT
  IF(YS(L(LOP)), LE, YS(L(HI))) GO TO 40
  LT = L(LOP)
  L(LOP) = L(HI)
  L(HI) = LT
40  IF(YS(L(LO)), LE, YS(L(HI))) GO TO 50
  LT = L(LO)
  L(LO) = L(HI)
  L(HI) = LT
50  IF(YS(L(LOP)), LE, YS(L(LO))) GO TO 60
  LT = L(LOP)
  L(LOP) = L(LO)
  L(LO) = LT
60  LWMED = L(LO)
I = LOP
J = HI
XT = YS(LWMED)
TLO=SLO
THI=SHI
70   TLO=TLO+W(L(I))
 I=I+1
 IF(YS(L(I)).LT.XT) GO TO 70
80   THI=THI+W(L(J))
 J=J-1
 IF(YS(L(J)).GT.XT) GO TO 80
 LT=L(I)
 L(I)=L(J)
 L(J)=LT
 GO TO 70
90   TEST=W(LWMED)
 IF(I.NE.J) GO TO 100
 TEST=TEST+W(L(I))
 I=I+1
 J=J-1
100  IF(TEST.GE.ABS(THI-TLO)) RETURN
 IF(TLO.GT.TH) GO TO 110
 SLO=TLO+TEST
 LO=I
 GO TO 10
110  SHI=THI+TEST
 LO=LOP
 HI=J
 GO TO 10
END

PROGRAM BL1S
C N Number of observation (input).
C Y(N) Dependent variable observations array (yi) (input).
C X2(N) Independent variable observations array (x2i) (input).
C Z(N) Working array for (yi/xi2).
C W(N) Working array for index of sorted array of (yi/xi1).
C L(N) Working array for weights (xi1).
PARAMETER (N=1000)
DIMENSION Y(N),X2(N),Z(N),W(N),L(N)
DO 10 I=1,N
10    READ(5,20) Y(I),X2(I)
20    FORMAT(2F10.3)
K1=N/2
K1R=0
K1S=0
30    DO 40 I=1,K1-1
 W(I)=X2(I)-X2(K1)
40    Z(I)=(Y(I)-Y(K1))/W(I)
 W(K1)=0.
 Z(K1)=0.
 DO 50 I=K1+1,N
 W(I)=X2(I)-X2(K1)
50    Z(I)=(Y(I)-Y(K1))/W(I)
 ITER=ITER+1
 LM=LWMED(N,Z,W,L)
 K1S=K1R
 K1R=K1
 IF (LM.EQ.K1S) GOTO 60
 K1=LM
 GOTO 30
60    B2=Z(LM)
B1=Y(K1)-B2*X2(K1)
PRINT 70,B1,B2
70  FORMAT(1X,'B1=',F13.5,3X,'B2=',F13.5)
STOP
END

PROGRAM BL1
C  N  Number of observation (input).
C  M  Number of parameters (ß) (input).
C  Y(N)  Dependent variable observations array (yi) (input).
C  X(N,M1)  Independent variable observations matrix (x2i,...,xmi) (input).
C  W(N)  Working array for index of sorted array of (yi/xi1).
C  L(N)  Working array for weights (xi1).
C  Other arrays are working arrays.
PARAMETER (N=1000,M=5,M1=M-1,M2=M-2)
DIMENSION Y(N),X(N,M1),XSK(M1),YW(N),XKW(N)
DIMENSION W(N),YS(N),XS(N,M1),B(M),XW(N,2:M1)
DIMENSION L(N),KK(M1),YSOL(M1),XSOL(M1,M1)
COMMON /C1/I1,I2
DO 10 I=1,N
10  READ(5,20) Y(I),(X(I,J),J=1,M1)
20  FORMAT(10F10.3)
ITER=0
KR=0
MM=1
DO 30 J=1,M1
30    KK(J)=J*N/M
40    DO 50 I=1,N
    YS(I)=Y(I)
    DO 50 J=MM,M1
    XS(I,J)=X(I,J)
    GO TO 80
60    DO 70 I=1,N
    W(I)=XKW(I)
    YS(I)=YW(I)
    DO 70 J=MM,M1
70    XS(I,J)=XW(I,J)
80    JJ=MM
90    K=KK(JJ)
    YSK=YS(K)
    DO 100 J=JJ,M1
100   XS(K,J)=XS(K,J)
    I1=1
    I2=K-1
110   DO 120 J=JJ,M1
120   CALL COL1(XSK(J),XS(1,J))
    CALL COL2(YSK,JJ,W,YS,XS(1,JJ))
    IF(I2.EQ.N) GO TO 130
    I1=K+1
    I2=N
    GO TO 110
130   W(K)=0.
    IF (JJ.EQ.M1) GO TO 190
    I1=1
    I2=K-1
140   DO 150 J=JJ+1,M1
150   CALL COL3(XS(1,J),XS(1,JJ))
    IF(I2.EQ.N) GO TO 160
    I1=K+1
    I2=N
GO TO 140
160 IF(JJ.NE.MM) GO TO 180
   DO 170 I=1,N
   XKW(I)=W(I)
   YW(I)=YS(I)
   DO 170 J=JJ+1,M1
170   XW(I,J)=XS(I,J)
180   JJ=JJ+1
   GO TO 90
190   YS(K)=0.
   ITER=ITER+1
   LM=LWMED(N,YS,W,L)
   IF(LM.EQ.KR) GO TO 220
   IOPT=0
200 IF(MM.EQ.M1) GO TO 210
   MM=MM+1
   KR=KK(MM)
   KK(MM)=LM
   GO TO 60
210   MM=1
   KR=KK(MM)
   KK(MM)=LM
   GO TO 40
220 IOPT=IOPT+1
   IF(IOPT.NE.M1) GO TO 200
   B(M)=YS(LM)
   DO 230 I=1,M1
   YSOL(I)=Y(KK(I))
   DO 230 J=1,M1
230   XSOL(I,J)=X(KK(I),J)
   JJ=1
240   YSK=YSOL(JJ)
   DO 250 J=JJ,M1
250   XSK(J)=XSOL(JJ,J)
   DO 270 I=JJ,M1
   IF(I.EQ.JJ) GO TO 270
   YSOL(I)=YSOL(I)-YSK
   DO 260 J=JJ,M1
260   XSOL(I,J)=XSOL(I,J)-XSK(J)
   YSOL(I)=YSOL(I)/XSOL(I,JJ)
270   CONTINUE
   DO 290 I=JJ,M1
   IF(I.EQ.JJ) GO TO 290
   DO 280 J=JJ+1,M1
280   XSOL(I,J)=XSOL(I,J)/XSOL(I,JJ)
290   CONTINUE
   IF(JJ.EQ.M2) GO TO 300
   JJ=JJ+1
   GO TO 240
300   DO 320 I=1,M2
   K=M-I
   S=YSOL(K)
   DO 310 J=K,M1
310   S=S-B(J+1)*XSOL(K,J)
320   B(K)=S
   S=Y(KK(1))
   DO 330 J=1,M1
330   S=S-B(J+1)*X(KK(1),J)
   B(1)=S
   PRINT 340,(B(J),J=1,M)
SUBROUTINE COL1(V1,V2)
DIMENSION V2(1)
COMMON /C1/I1,I2
DO 1 I=I1,I2
1 V2(I)=V2(I)-V1
RETURN
END

SUBROUTINE COL2(YSK,JJ,V1,YS,V2)
DIMENSION V1(1),V2(1),YS(1)
COMMON /C1/I1,I2
IF (JJ.NE.1) GO TO 2
DO 1 I=I1,I2
V1(I)=V2(I)
1 YS(I)=(YS(I)-YSK)/V2(I)
RETURN
2 DO 3 I=I1,I2
V1(I)=V1(I)*V2(I)
3 YS(I)=(YS(I)-YSK)/V2(I)
RETURN
END

SUBROUTINE COL3(V1,V2)
DIMENSION V1(1),V2(1)
COMMON /C1/I1,I2
DO 1 I=I1,I2
1 V1(I)=V1(I)/V2(I)
RETURN
END

References
- B. Bidabad (1987a) Least absolute error estimation. Submitted to the First International Conference on Statistical Data Analysis Based on the L_{1} norm and Related Methods, Neuchatel, Switzerland.
- B. Bidabad (1987b) Least absolute error estimation, part II. Submitted to the First International Conference on Statistical Data Analysis Based on the L_{1} norm and Related Methods, Neuchatel, Switzerland.
- B. Bidabad (1989a) Discrete and continuous L_{1} norm regressions, proposition of discrete approximation algorithms and continuous smoothing of concentration surface, Ph.D. thesis, Islamic Azad University, Tehran, Iran and Neuchatel University of Switzerland.
- B. Bidabad (1989b) Discrete and continuous L_{1} norm regressions, proposition of discrete approximation algorithms and continuous smoothing of concentration surface, Ph.D. thesis, Islamic Azad University, Tehran, Iran and Neuchatel University of Switzerland. Farsi translation.