Bangladesh Journal of Multidisciplinary Scientific Research
Vol. 1, No. 1
April-June; 2019
Published by Centre for Research on Islamic Banking & Finance and Business

L_1 Norm Solution of Overdetermined System of Linear Equations

Bijan Bidabad
B.A., M.Sc., Ph.D., Post-Doc.
Professor
Economics and Chief Islamic Banking Advisor
Bank Melli, Iran
E-mail:bijan@bidabad.com

Abstract
In this paper, three algorithms for weighted median, simple linear, and multiple m parameters L_1 norm regressions are introduced. The corresponding computer programs are also included.

Keywords: L_1 norm, Regression, Algorithm, Computer program

1. Introduction

L_1 norm criterion is going to find its place in scientific analysis. Since it is not computationally comparable with other criteria such as L_2 norm, it needs more work to make it a hand tool. The closed form of the solution of the L_1 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 the L_1 norm linear regression problem. The proposed algorithms are based on a special descent method and use a 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,

$$y_i = \sum_{j=1}^{m} \beta_j x_{ij} + u_i \qquad i=1,\ldots,n \tag{1}$$

where β_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 β_j's by minimizing the sum of absolute errors given by the following expression:

$$S = \sum_{i=1}^{n} |u_i\hat{}| = \sum_{i=1}^{n} \left| y_i - \sum_{j=1}^{m} \beta_j\hat{} x_{ji} \right| \tag{2}$$

where $\beta_j\hat{}$ is the estimated value of β_j. When m=1, we are confronted with a weighted median problem.

2. Weighted median computation (restricted one parameter model)

Let us now consider a simple restricted linear model in which m=1 namely,

$$y_i = \beta_1 x_{i1} + u_i \tag{3}$$

For the model given by (3), the L1 norm objective function S to be minimized will be,

$$S = \sum^{n} | y_i - \beta_1 x_{i1} | = \sum^{n} |x_{i1}| \, |y_i/x_{i1} - \beta| \tag{4}$$

i=1 i=1

Two series of computations are necessary to compute the weighted median. One sorting algorithm is essential to sort the ratio array (y_i/x_{i1}) and restoring the corresponding subscripts for the second part of the calculation to find the left and right weights ($|x_{i1}|$) 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, the 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)≤0 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))≤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))≤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))≤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))≤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 ys(l(i))<xt then go to step 5.a, otherwise go to step 5.b.
  b. Let: thi=thi+w(l(j)), j=j-1.
    If ys(l(j))>xt then go to step 5.b, otherwise if j≤i then go to step 6, otherwise lt=l(i), l(i)=l(j), l(j)=lt, 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≥ |thi-tlo| then return, otherwise, if tlo>thi then step 6.b, otherwise slo=tlo+test, lo=i, go to step 3.
b. Let: shi=thi+test, lo=lop, hi=j.
Go to step 3.
END

3. 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,...,n; namely,

$$y_i = \beta_1 + \beta_2 x_{i2} + u_i \quad (5)$$

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

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

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), z(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

4. General linear model

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

PROGRAM BL1

Step 0) Initialization.
Parameter: n, m, m1=m-1, m2=m-2.

Real: y(n), x(n,m1), xsk(m1), yw(n), xkw(n), w(n), ys(n), xs(n,m1), b(m), xw(n,2:m1), ysol(m1), xsol(m1,m1).

Integer: l(n), kk(m1).

Common: /c1/i1,i2.

Read: (y(i),(x(i,j),j=1,m1),i=1,n).

Let: iter=0, kr=0, mm=1, (kk(j)=arbitrary,j=1,m1).

Step 1) Refill working arrays.

Do loop for i=1,n: ys(i)=y(i), do loop for j=1,m1: xs(i,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)=xkw(i), ys(i)=yw(i), do loop for j=1,m1: xs(i,j)=xw(i,j), end do, end do.

Step 3) Compute the arguments for weighted median.

a. Set: jj=mm, k=kk(jj), ysk=ys(k), i1=1, i2=k-1.
Do loop for j=jj,m1: xsk(j)=xs(k,j), end do.

b. Do loop for j=jj,m1: call COL1(xsk(j),xs(1,j)) end do.
Call COL2(ysk,jj,w,ys,xs(1,jj)).
If i2=n go to step 3.c; otherwise set: i1=k+1, i2=n, go to step 3.b.

c. Set: w(k)=0.
If jj=m1 go to step 4; otherwise i1=1, i2=k-1, go to step 3.d.

d. Do loop for j=jj+1,m1: call COL3(xs(1,j),xs(1,jj)), end do.
If i=n go to step 3.e; otherwise i1=k+1, i2=n, go to step 3.d.

e. If jj≠mm jj=jj+1, go to step 3, otherwise do loop for i=1,n: xkw(i)=w(i), yw(i)=ys(i); do loop for j=jj+1,m1: xw(i,j)=xs(i,j), end do; end do.
Set: jj=jj+1, go to step 3.

Step 4) Compute the weighted median.

Set: ys(k)=0, iter=iter+1, lm=LWMED(n,ys,w,l).

Step 5) Test for optimality.

If lm=kr go to step 5.b; otherwise iopt=0 go to step 5.a.

a. If mm=m1 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: iopt=iopt+1.
If iopt≠m1 go to step 5.a, otherwise go to step 6.

Step 6) Compute the solution.

Set: b(m)=ys(lm).
Do loop for i=1,m1: ysol(i)=y(kk(i)); do loop for j=1,m1: xsol(i,j)=x(kk(i),j), end do; end do.
Set: jj=1.

a. Set: ysk=ysol(jj).
Do loop for j=jj,m1: xsx(j)=xsol(jj,j), end do.
Do loop for i=jj,m1: if i=jj go to continue; otherwise ysol(i)=ysol(i)-ysk, do loop for j=jj,m1: xsol(i,j)=xsol(i,j)- xsk(j), end do; set ysol(i)=ysol(i)/xsol(i,jj), continue, end do.

b. Do loop for i=jj,m1: if i=jj go to continue, otherwise, do loop for j=jj+1,m1: xsol(i,j)=xsol(i,j)/xsol(i,jj), end do; continue; end do.

c. If jj=m2 go to step 6.d; otherwise go to step 6.a.

d. Do loop for i=1,m2: k=m-i, s=ysol(k); do loop for j=k,m1, s=s- b(j+1)*xsol(k,j) end do, b(k)=s, end do. Set: s=y(kk(1)).
Do loop for j=1,m1: s=s-b(j+1)*x(kk(1),j), b(1)=s, end do.
Print: ((b(j),j=1,m),(kk(j),j=1,m1),lm,iter).

Stop.

END

The major portion of computation in this program is the transformation of two-dimensional arrays. Passing columns of these arrays to other subroutines which involve 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 the 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

5. 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 (outpot).
      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 I=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 I=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
     IF(J.LE.I) GO TO 90
     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.THI)GO TO 110
     SLO=TLO+TEST
     LO=I
     GO TO 10
110  SHI=THI+TEST
     LO=LOP
     HI=J
     GO TO 10
     END

     PROGRAM BLIS
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)
      KIS=KIR
      KIR=K1
      IF (LM.EQ.KIS) 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=1,M1
50    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   XSK(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)
340   FORMAT(1X,F13.5)
      PRINT 350,(KK(J),J=1,M1),LM,ITER
350   FORMAT(1X,I13)
      STOP
      END

      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

Barrodale, F.D.K. Roberts (1974) Algorithm 478: Solution of an overdetermined system of equations in the L_1 norm. Commun. ACM, 17, 319- 320.

Bijan Bidabad (1987a) Least absolute error estimation. The First International Conference on Statistical Data Analysis Based on the L_1 norm and Related Methods, Neuchatel, Switzerland. http://www.bidabad.com/doc/lae-I.pdf

Bijan 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. http://www.bidabad.com/doc/lae-II.pdf

Bijan Bidabad (1988a) A proposed algorithm for least absolute error estimation. Proc. of the Third Seminar of Mathematical Analysis. Shiraz Univ., 24-34, Shiraz, Iran.

Bijan Bidabad (1988b) A proposed algorithm for least absolute error estimation, part II. Proc. of the Third Seminar of Mathematical Analysis, Shiraz Univ., 35-50, Shiraz, Iran.

Bijan 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 Univ., Tehran, Iran. http://www.bidabad.com/doc/L1-norm-thesis-en.pdf

Bijan 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 Univ., Tehran, Iran. Farsi translation. http://www.bidabad.com/doc/L1-norm-thesis-fa.pdf

Bijan Bidabad (2005). L_1 norm based computational algorithms. http://www.bidabad.com/doc/l1-article6.pdf

Bijan Bidabad (2005). L_1 norm solution of overdetermined system of linear equations. http://www.bidabad.com/doc/l1-article5.pdf

Bijan Bidabad (2005). L_1 norm based data analysis and related methods. http://www.bidabad.com/doc/l1-articl1.pdf

Bijan Bidabad (2005). New algorithms for the L_1 norm regression. http://www.bidabad.com/doc/l1-article2.pdf

Bijan Bidabad (2005). Comparative study of the L_1 norm regression algorithms. http://www.bidabad.com/doc/l1-articl3.pdf

Bijan Bidabad (2005). Continuous L_1 norm estimation of Lorenz curve. http://www.bidabad.com/doc/l1-articl4.pdf

Bijan Bidabad (1993). Estimating Lorenz curve for Iran by using continuous L_1 norm estimation, Economics and Management Journal, Islamic Azad University, No. 19, winter 1993, pp. 83-101. http://www.bidabad.com/doc/iraninc-l1.pdf

Bijan Bidabad (2005). Continuous L_1 norm estimation of Lorenz curve when probability density function is known.

Bijan Bidabad (2005). USA Income distribution counter-business-cyclical trend (Estimating Lorenz curve using continuous L_1 norm estimation). First meeting of the Society for the Study of Economic Inequality (ECINEQ), Palma de Mallorca, Spain, July 20-22, 2005.

http://www.uib.es/congres/ecopub/ecineq/general.html

http://www.uib.es/congres/ecopub/ecineq/papers/039Bidabab.pdf

http://www.bidabad.com/doc/estimating-lorenz-us.pdf

Bijan Bidabad, Hamid Shahrestani. (2008) An implied inequality index using L_1 norm estimation of Lorenz curve. Global Conference on Business and Finance Proceedings. Mercedes Jalbert, managing editor, ISSN 1931-0285 CD, ISSN 1941-9589 Online, Volume 3, Number 2, 2008, The Institute for Business and Finance Research, Ramada Plaza Herradura, San Jose, Costa Rica, May 28-31, 2008, pp. 148-163. Global Journal of Business Research, Vol. 4, No. 1, 2010, pp.29-45.

http://www.bidabad.com/doc/L1-Implied-inequality-index-4.pdf

http://www.theibfr.com/archive/ISSN-1941-9589-V3-N2-2008.pdf

http://www.bidabad.com/doc/SSRN-id1631861.pdf

P. Bloomfield, W. Steiger (1980) Least absolute deviations curve fitting. SIAM J. Sci. Stat. Com.1,290-301.

J. Chambers (1971) Algorithm 410: partial sorting. Commun. ACM, 14, 357- 358.

C.A.R. Hoare (1961) Algorithm 63 partition; 64, quicksort; and 65, find., Comm. ACM, 4, July, 321-322.

C.A.R. Hoare (1962) Quicksort. Comput. J., 5, 10-15.

S. Scowen (1965) Algorithm 271 Quickersort. Commun. ACM, 8, 669-670.
R.S. Singleton (1969) Algorithm 347 Sort. Comm. ACM, 12, 185-186.