4.5 Sequential Solution of a Banded Least-Squares Problem

A. Purpose

Let a linear least-squares problem be denoted by

$$A\mathbf{x} \simeq \mathbf{b}$$

where A is a given $m \times n$ matrix with $m \geq n$, $\mathbf{b}$ is a given m-vector, and it is required to find an n-vector, $\mathbf{x}$, that is an approximate solution to this equation in the least-squares sense. The given data for this problem can be regarded as the composite matrix, $[A : \mathbf{b}]$.

Assume further that the matrix A is *banded*, in the sense that there is an integer, NB $\leq n$, such that in any row of A all nonzero elements occur within a set of NB contiguous positions. This set of subroutines is intended for the case in which NB is significantly smaller than n so there is the possibility of achieving some significant saving of storage and execution time by taking advantage of this band property.

Subroutine SBACC can be used to preprocess the data matrix, $[A : \mathbf{b}]$, sequentially. Subroutine SBSOL can be used to solve the problem after the transformation accomplished by SBACC. Subroutine SBSOL can also be used to compute a covariance matrix for the solution vector.

B. Usage

B.1 Sequential processing of data

The user must make a sequence of calls to SBACC, sending a block of rows of $[A : \mathbf{b}]$ with each call. On the first call the user sets NB, the bandwidth, which will remain unchanged after that. Also, on the first call the user must set IR = 1, and JT = 1. After that IR will be updated by SBACC.

Along with each block the user sets two integers, MT and JT, indicating that MT new rows are being provided, within these rows of A only the NB columns beginning with column JT are being provided, and that the elements in all other columns in these rows of A are zero. The user puts this block of data in the array G(,), beginning with row IR of G(,). The data corresponding to column JT of A goes into the first column of G(,). Data from the vector $\mathbf{b}$ of the problem goes into column NB + 1 of G(,).

The data blocks must be ordered so that from one call to the next, the value of the column index JT either remains the same or increases.

B.1.a Program Prototype, Single Precision

INTEGER LDG, NB, IR, MT, JT, JTPREV, IERR2

REAL G(LDG, ≥NB+1)

When starting a new problem set LDG, NB, and IR. On the initial call and all subsequent calls for the same problem, set MT and JT and store MT rows of data into G(,) beginning at row IR.

CALL SBACC(G, LDG, NB, IR, MT, JT, JTPREV, IERR2)

Following the call the contents of G(,) will have been altered, reflecting the processing of the new data. The values of JTPREV and IR may have been changed. IERR2 will be set.

B.1.b Argument Definitions

G(,) [inout] The working array. With MT, JT, and IR defined below, on each call to SBACC the user places columns JT to JT + NB − 1 of MT rows of A into G(IR:IR+MT−1, 1:NB) and the corresponding elements of $\mathbf{b}$ into G(IR:IR+MT−1, NB+1). It is implied that within these MT rows of A all elements not in columns JT through JT + NB − 1 are zero.

LDG [in] The leading dimensioning parameter for G(,). LDG must be large enough so that on each call to SBACC one has IR + MT − 1 ≤ LDG.

MT can be set by the user on each call. Let $MT_{\max}$ denote the largest value the user will assign to MT. If the user keeps JT $\leq n -$ NB $+ 1$ (see discussion in Section C) then the largest value of IR will be $n + 2$, so it would suffice to set LDG $= n + 1 + MT_{\max}$.

If the user permits JT to be as large as n, then LDG must be at least n + NB + $MT_{\max}$.

NB [in] Set by user on the initial call for a problem. Must not be changed during the processing for one problem. NB indicates the bandwidth of the data matrix A. In any row of A all nonzero elements must appear in some set of NB consecutive columns.

IR [inout] Index of first row of G(,) into which the user is to place new data. The variable IR must be set by the user to the value 1 on the initial call to SBACC and must not be altered after that by the user on successive calls for the same problem. IR will be updated in SBACC by (effectively) setting IR = JT + min(NB + 1, MT + max(IR − JT, 0)).

Let IR_{in} denote the value of IR on entry to SBACC and IR_{out} denote its value on return. These quantities will satisfy

$$IR_{in} \leq IR_{out} \leq \text{JT} + \text{NB} + 1.$$

MT [in] Set by user to indicate the number of new rows of data being introduced by the current call to SBACC. SBACC will return immediately if MT ≤ 0. MT will not be altered by SBACC.

JT [in] Set by user to indicate the column of the new sub-block of A that the user is storing in the first column of the work array, G(,). The user must set JT = 1 on the initial call to SBACC and must either leave JT the same or increase it on each successive call for the same problem. Too large an increase between successive calls can cause the problem to be structurally singular. See Section E for more information on this. JT will not be altered by SBACC.

JTPREV [inout] Need not be set before the first call to SBACC for a problem. Must not be altered by the user on later calls for the same problem. JTPREV is used within SBACC to mark the row of G(,) at which the method of packing data changes. Quantities in G(,) at and below row JTPREV are subject to potential change during the processing of additional data, whereas quantities above row JTPREV are not.

On return, SBACC sets JTPREV = JT. JTPREV, as set by SBACC, is needed as input when calling SBSOL.

IERR2 [inout] Error status indicator. Need not be set before the initial call for a problem. Will be set by SBACC to zero if no errors are detected and to nonzero values, as described in Section E, if error conditions are detected.

B.1.c Modifications for Double Precision

For double precision usage change the REAL statement to DOUBLE PRECISION and change the subroutine name SBACC to DBACC.

B.2 Computation of Solution Vector

B.2.a Program Prototype, Single Precision

INTEGER MODE, LDG, NB, IR, JTPREV, N, IERR3

REAL G(LDG, ≥NB+1), **X**(≥ N), **RNORM**

Set MODE and N. If MODE is set to 2 or 3 then also store a vector, **p**, into X(). G(,), NB, IR, and JTPREV should have values resulting from previous calls to SBACC.

CALL SBSOL(MODE, G, LDG, NB, IR, JTPREV, X, N, RNORM, IERR3)

On return, X() and RNORM will contain computed results and IERR3 will be set. No other quantities in the argument list will be modified.

B.2.b Argument Definitions

MODE [in] The previous processing of data by SBACC will have left a representation of an upper triangular matrix, R, and a vector, $\mathbf{y}$, in the array, G(,). See Section D for the interpretation of these quantities. The user selects the desired solution process by setting MODE to 1, 2, or 3.

- =1 Solve $R\mathbf{x} = \mathbf{y}$, where R and $\mathbf{y}$ are contained in G(,) as the result of previous calls to SBACC. The solution vector, $\mathbf{x}$, will be stored in X(). This gives the solution to the least-squares problem, $A\mathbf{x} \simeq \mathbf{b}$.
- =2 Solve $R^t\mathbf{x} = \mathbf{p}$, where R is the matrix residing in G(,) as the result of previous calls to SBACC, and $\mathbf{p}$ is a vector placed in X() by the user. The solution vector, $\mathbf{x}$, will replace $\mathbf{p}$ in X().
- =3 Solve $R\mathbf{x} = \mathbf{p}$, where R is the matrix residing in G(,) as the result of previous calls to SBACC, and $\mathbf{p}$ is a vector placed in X() by the user. The solution vector, $\mathbf{x}$, will replace $\mathbf{p}$ in X().

G(,),LDG,NB,IR,JTPREV [in] These arguments must contain values as they were defined upon the return from a preceding call to SBACC.

X() [inout] On input, with MODE = 2 or 3, this array must contain the N-dimensional right-side vector of the system to be solved. On return, with MODE = 1, 2, or 3, this array will contain the N-dimensional solution vector of the appropriate system that has been solved.

N [in] Set by user to specify the dimensionality of the desired solution vector. This causes the subroutine SBSOL to use only the leading N×N submatrix of the triangular matrix currently represented in the array G(,). An error is reported if this submatrix is singular.

RNORM [out] If MODE = 1, RNORM is set by the subroutine to the norm of the residual vector for the least-squares problem, *i.e.*, $\|\mathbf{b} - A\mathbf{x}\|$. This number is computed as

$$\left[\sum_{I=\text{N}+1}^{\text{IR}-1} \text{G}(I, \text{NB} + 1)^2 \right]^{1/2}$$

If MODE = 2 or 3, RNORM is set to zero.

IERR3 [out] Error status indicator set by SBSOL. Zero means no errors detected. If a diagonal element of the leading N×N submatrix represented in G(,) is zero, an error message will be issued and IERR3 will be set to the index of the first zero diagonal element. In this latter case the solution vector X() will not be computed.

B.2.c Changes for Double Precision

For double precision usage change the REAL statement to DOUBLE PRECISION and change the subroutine name SBSOL to DBSOL.

C. Examples and Remarks

C.1 Computation of the covariance matrix for the solution vector

The features provided by MODE = 2 and 3 are primarily intended to support the computation of the unscaled covariance matrix, C, for the least-squares problem. This matrix, C, is defined by $C = (A^tA)^{-1}$. However, since $R^tR = A^tA$ (See Section D), C is also given by $C = (R^tR)^{-1}$, and thus C satisfies the equation, $R^tRC = I$, where I is the $n \times n$ identity matrix. It follows that the matrix, C, can be computed in two steps, first solving

$$R^tZ = I$$

for Z, and then solving

$$RC = Z$$

for C. Using SBSOL or DBSOL the matrices Z and C can be computed one column at a time.

This matrix, C, must be multiplied by an estimate of the variance of the data error to obtain the solution covariance matrix. This variance can be estimated using N, MTOTAL, and RNORM as follows:

DOF = MTOTAL − N

VAR = RNORM**2 / DOF

where MTOTAL is the total number of rows of A introduced into the problem.

C.2 Demonstration problem

As a demonstration problem we compute the continuous piecewise linear function that best fits a sample of 91 values of the sine function in the least-squares sense. Sample values are values of the sine function at one degree steps from zero to 90 degrees. The piecewise linear function will be parameterized by a set of ten values, y_i, $i = 1, 10$, which will be the values of the piecewise linear function at the arguments 0, 10, 20, ..., 90 degrees. The fitted function will be defined by linear interpolation between adjacent pairs of these points.

Note that because the sine curve is concave down throughout this interval we expect the knots, y_i, each to lie above the sine curve, with the linearly interpolated segment between each adjacent pair of knots passing below the sine curve.

Program DRDBACC illustrates the use of DBACC and DBSOL to compute the ten values of y_i for this problem. It also computes the (formal) 10×10 covariance matrix for these quantities. The results are shown in ODDBACC.

Note from the listing of residuals that the residuals at $x = 0, 10, 20, ...$ degrees are positive, while the residuals at $x = 5, 15, 25, ...$ degrees are negative, as expected.

This problem is not really statistical since the given data are essentially exact. Thus the computed SIGFAC and covariance matrix do not really have a statistical interpretation, but merely serve to illustrate how to use these subroutines to compute these quantities. Note in particular that the covariance matrix is symmetric (to some level of accuracy), as it should be, even though this method computes each column of the covariance matrix independently.

For an example of more general usage of DBACC see a listing of the MATH77 library subroutine, DC2FIT, Chapter 11.4.

C.3 Data blocks having fewer than NB contiguous columns of nonzeros

This subroutine requires the value of NB to be constant throughout the processing of one problem. If the data block being sent to SBACC in one call has all of its nonzeros in fewer than NB contiguous columns the user must pad out the block to a width of NB columns by including zeros. The padding columns may be either on the left or the right or both.

For example, suppose NB = 4 and one is introducing a row, or block of rows, of A having nonzeros only in columns 5, 6, and 7. One may either set JT = 4 and send columns 4, 5, 6, and 7, with column 4 containing zeros; or else one may set JT = 5 and send columns 5, 6, 7, and 8, with column 8 containing zeros.

In choosing between these alternatives there are two points to consider. JT must be nondecreasing on successive calls to SBACC. Sending padding columns that are beyond the last actual column of A causes the algorithm to use more rows of storage than would otherwise be necessary.

D. Functional Description

Subroutine SBACC uses Householder orthogonal transformations to process the given data, producing an equivalent least-squares problem of the form

$$\begin{bmatrix} R \\ 0 \end{bmatrix} \simeq \begin{bmatrix} \mathbf{y} \\ \alpha \end{bmatrix}$$

where R is an $n \times n$ upper triangular matrix with a bandwidth of NB, $\mathbf{y}$ is an n-vector, and α is a scalar quantity. These quantities are related to the data, $[A : \mathbf{b}]$, by the relations, $R^t R = A^t A$, $R^t \mathbf{y} = A^t \mathbf{b}$, and $\mathbf{y}^t \mathbf{y} + \alpha^2 = \mathbf{b}^t \mathbf{b}$.

The solution, $\mathbf{x}$, for this problem is also the solution for the given least-squares problem, $A\mathbf{x} \simeq \mathbf{b}$, and can be computed by solving the triangular system, $R\mathbf{x} = \mathbf{y}$. This latter system is solved for $\mathbf{x}$ when SBSOL is called with MODE = 1.

The residual vector for the transformed least-squares problem is

$$\begin{bmatrix} \mathbf{0} \\ \alpha \end{bmatrix}.$$

The norm of this residual vector is $|\alpha|$ and this is also equal to $\|\mathbf{b} - A\mathbf{x}\|$. After n linearly independent rows of A have been accumulated, where n is the number of columns of A, if a solution is requested with N = n, then the value $|\alpha|$ will be returned as RNORM. If a solution is requested with N < n, components of $\mathbf{y}$ beyond y_N will also be used in computing RNORM.

Subroutine SBACC dynamically partitions the array G(,) into three segments by groups of rows. These segments are rows 1 through JTPREV − 1, rows JTPREV through IR − 1, and rows IR through LDG.

The first segment holds rows of $[R : \mathbf{y}]$ for which processing is completed. In this segment element $r_{i,j}$ is stored in G$(i, j - i + 1)$.

The second segment, consisting of at most NB + 1 rows, holds rows of

$$\begin{bmatrix} R & : & \mathbf{y} \\ 0 & : & \alpha \end{bmatrix}$$

that may be changed by data yet to be received. In this segment element $r_{i,j}$ is stored in G$(i, j - \text{JTPREV} + 1)$.

The third segment is used to receive new rows of $[A : \mathbf{b}]$.

The subroutine SBSOL alters only the contents of X() and RNORM. Thus it is permissible to alternate between accumulating data using SBACC and obtaining a current solution or covariance matrix using SBSOL. It is also permissible to use SBSOL with N < n. The subroutine SBSOL can compute a solution vector of length N whenever the portion of the A matrix introduced to that point has the property that its first N columns are linearly independent.

References

1. Charles L. Lawson and Richard J. Hanson, **Solving Least-Squares Problems**, Prentice-Hall, Englewood Cliffs, N. J. (1974) 340 pages.

E. Error Procedures and Restrictions

In response to detected error conditions, SBACC will set IERR2 nonzero and issue an error message using the error message routines of Chapter 19.2. On errors numbered 1 through 4, SBACC will return. Generally the calling program should abandon the problem processing in these cases. When IERR2 = 3 it is conceivable, but not likely, that the user might wish to continue the processing, in which case the user must reset IERR2 to zero. If SBACC is reentered with IERR2 ≠ 0, (and IR > 1) it will set IERR2 = 5 and execute a STOP.

IERR2 Explanation

1 MT > LDG − IR + 1. To correct this problem either use a larger dimension LDG, or a smaller block size MT.

2 JT < JTPREV. This occurs if the data blocks are out of order, in the sense that the current JT is smaller than JT on the previous call.

3 JT > min(JTPREV + NB, IR). This occurs when the difference between the current JT and JT on the previous call is so large that the matrix would be structurally singular. This condition does not, however, preclude the processing of the data to triangular form, and thus SBACC will do the processing, unless there is a storage limitation, in which case IERR2 will be set to 4.
This condition is likely to be due to a program usage error. If not, then the problem needs either more data in preceding blocks or a mathematical model with fewer or differently defined free parameters. Alternatively, see pp. 218–219 of [1] for ideas on stabilizing such a structurally singular problem.

4 The condition described above for IERR2 = 3 holds and there is a storage limitation indicated by MT > LDG − JT + 1. The storage problem could be relieved by increasing LDG or decreasing MT. However, the more fundamental problem indicated by IERR2 = 3 must still be dealt with.

5 SBACC has been entered with IERR2 ≠ 0. SBACC will execute a STOP since the calling program has ignored a nonzero setting of IERR2 on the previous call.

The following conditions are not tested in SBACC and violation will have unpredictable effects: IR, and JT

must be set to 1 on the first call to SBACC. JTPREV and IR must not subsequently be altered by the user during the processing for one problem.

SBACC will return immediately if MT ≤ 0. This is not regarded as an error condition.

If SBSOL encounters a zero diagonal term in the N×N matrix R, IERR3 will be set to the index of the (first) zero term and an error message will be issued. The solution X() will not be computed in this case.

F. Supporting Information

The source language is ANSI Fortran 77.

These subroutines are adaptations to the JPL MATH77 library of the algorithms and subroutines BNDACC and BNDSOL that were developed by C. L. Lawson and R. J. Hanson at JPL in 1972 and described in detail in [1].

Entry	Required Files
DBACC	DBACC, DHTCC, DNRM2, ERFIN, ERMSG, IERM1, IERV1
DBSOL	DBSOL, ERFIN, ERMSG, IERM1, IERV1
SBACC	ERFIN, ERMSG, IERM1, IERV1, SBACC, SHTCC, SNRM2
SBSOL	ERFIN, ERMSG, IERM1, IERV1, SBSOL

DRDBACC

```
c      program DRDBACC
c>> 1996-05-28 DRDBACC  Krogh  Moved formats up.
c>> 1994-10-19 DRDBACC  Krogh  Changes to use M77CON
c>> 1987-12-09 DRDBACC  Lawson  Initial Code.
c--D replaces "?": DR?BACC, ?BACC, ?BSOL
c      Demonstration driver for DBACC & DBSOL
c      C. L. Lawson & S. Y. Chiu, JPL, July 1987, Sept 1987.
c     ------------------------------------------------------------------
      integer NX, LDG, NB, ISCALE
      integer I, IERR2, IERR3, IG, IR, J, JT, JTPREV, MT, MTOTAL
      parameter(NX = 10, LDG = 24, NB = 2, ISCALE = 6)
      double precision XTAB(NX), G(LDG,3), YFIT(NX), C(NX,NX)
      double precision DELX, DOF, DTOR, RDUMMY, RNORM, SIGFAC, VFAC
      double precision X, YF, YTRUE
      double precision ZERO, XINC, XLIMIT
      double precision ONE, C45, HALF
      parameter(ZERO = 0.0D0, XINC = 1.0D0, XLIMIT = 89.5D0)
      parameter(ONE = 1.0D0, C45 = 45.0D0, HALF = 0.5D0)
      data XTAB / 00.0D0, 10.0D0, 20.0D0, 30.0D0, 40.0D0,
     *            50.0D0, 60.0D0, 70.0D0, 80.0D0, 90.0D0 /
c
 1000 format(' Calling DBACC with JT =',i3,', MT =',i3)
 1003 format(1x/'      X        Y         YFIT    R=Y-YFIT'/1X)
 1004 format(1X, F6.1, 3F10.5)
 1006 format(1X,10(F7.2,1X))
c     ------------------------------------------------------------------
      write(*,'(1x,a//1x,a/1x,a/1x)')
     *'         Demonstration driver for DBACC and DBSOL.',
     *'Compute least-squares fit of a continuous piecewise linear',
     *'      function to the sine function on 0 to 90 degrees.'
      DTOR = atan(ONE)/C45
      MTOTAL = 0
      IR = 1
      MT = 0
      JT = 1
      IG = 0
      X = -XINC
      DELX = XTAB(JT+1) - XTAB(JT)
c
   20 if( X .lt. XLIMIT)then
         X = X + XINC
```

```
         MTOTAL = MTOTAL + 1
      if(X .gt. XTAB(JT+1))then
            write(*,1000) JT, MT
            call DBACC(G,LDG,NB,IR,MT,JT,JTPREV, IERR2)
            IG = IR-1
            MT = 0
            JT = min(JT+1, NX-1)
            DELX = XTAB(JT+1) - XTAB(JT)
      end if
         IG = IG + 1
         MT = MT + 1
         G(IG,1) = (XTAB(JT+1) - X) / DELX
         G(IG,2) = (X - XTAB(JT)) / DELX
         G(IG,3) = sin(x * DTOR)
      go to 20
      end if

      if(MT .gt. 0)then
         write(*,1000) JT, MT
         call DBACC(G,LDG,NB,IR,MT,JT,JTPREV, IERR2)
      end if

      call DBSOL(1,G,LDG,NB,IR,JTPREV,YFIT,NX,RNORM, IERR3)
c                        The following statement does a type conversion.
      DOF = MTOTAL - NX
      SIGFAC = RNORM / SQRT(DOF)
      write(*,'(1x/1x, a, i4, a, f10.5, a, f10.5)')
     *  'MTOTAL =', MTOTAL, ',  RNORM =', RNORM, ',  SIGFAC =', SIGFAC
      write(*,1003)
      do 30 I=1,NX
         YTRUE = sin(XTAB(I)*DTOR)
         write(*,1004) XTAB(I),YTRUE,YFIT(I),YFIT(I) - YTRUE
      if( I .ne. NX)then
            X = HALF*(XTAB(I+1) + XTAB(I))
            YF = HALF*(YFIT(I+1) + YFIT(I))
            YTRUE = sin(X*DTOR)
            write(*,1004) X,YTRUE,YF,YF - YTRUE
      end if
   30 continue
c
c     Compute unscaled covariance matrix in C(,).
c
      do 50 J = 1,NX
        do 40 I = 1,NX
            C(I,J) = ZERO
   40   continue
        C(J,J) = ONE
        call DBSOL(2,G,LDG,NB,IR,JTPREV,C(1,J),NX,RDUMMY,IERR3)
        call DBSOL(3,G,LDG,NB,IR,JTPREV,C(1,J),NX,RDUMMY,IERR3)
   50 continue
c
      write(*,'(1x/10x,a,i1/1x)')
     *  'Covariance matrix scaled up by 10**',ISCALE
c
      VFAC = (10**ISCALE) * SIGFAC**2
      do 60 I = 1,NX
        print 1006,(VFAC*C(I,J),J=1,NX)
   60 continue
c
```

```
stop
end
```

ODDBACC

Demonstration driver for DBACC and DBSOL.

Compute least-squares fit of a continuous piecewise linear
function to the sine function on 0 to 90 degrees.

```
Calling DBACC with JT =  1, MT = 11
Calling DBACC with JT =  2, MT = 10
Calling DBACC with JT =  3, MT = 10
Calling DBACC with JT =  4, MT = 10
Calling DBACC with JT =  5, MT = 10
Calling DBACC with JT =  6, MT = 10
Calling DBACC with JT =  7, MT = 10
Calling DBACC with JT =  8, MT = 10
Calling DBACC with JT =  9, MT = 10

MTOTAL =  91,  RNORM =   0.00818,  SIGFAC =   0.00091
```

X	Y	YFIT	R=Y-YFIT
0.0	0.00000	0.00009	0.00009
5.0	0.08716	0.08708	−0.00008
10.0	0.17365	0.17406	0.00041
15.0	0.25882	0.25847	−0.00034
20.0	0.34202	0.34289	0.00087
25.0	0.42262	0.42207	−0.00055
30.0	0.50000	0.50126	0.00126
35.0	0.57358	0.57283	−0.00075
40.0	0.64279	0.64441	0.00162
45.0	0.70711	0.70619	−0.00092
50.0	0.76604	0.76797	0.00193
55.0	0.81915	0.81809	−0.00106
60.0	0.86603	0.86821	0.00219
65.0	0.90631	0.90512	−0.00119
70.0	0.93969	0.94203	0.00234
75.0	0.96593	0.96471	−0.00122
80.0	0.98481	0.98738	0.00257
85.0	0.99619	0.99476	−0.00143
90.0	1.00000	1.00215	0.00215

Covariance matrix scaled up by 10**6

0.24	−0.06	0.02	−0.00	0.00	−0.00	0.00	−0.00	0.00	−0.00
−0.06	0.15	−0.04	0.01	−0.00	0.00	−0.00	0.00	−0.00	0.00
0.02	−0.04	0.14	−0.04	0.01	−0.00	0.00	−0.00	0.00	−0.00
−0.00	0.01	−0.04	0.14	−0.04	0.01	−0.00	0.00	−0.00	0.00
0.00	−0.00	0.01	−0.04	0.14	−0.04	0.01	−0.00	0.00	−0.00
−0.00	0.00	−0.00	0.01	−0.04	0.14	−0.04	0.01	−0.00	0.00
0.00	−0.00	0.00	−0.00	0.01	−0.04	0.14	−0.04	0.01	−0.00
−0.00	0.00	−0.00	0.00	−0.00	0.01	−0.04	0.14	−0.04	0.02
0.00	−0.00	0.00	−0.00	0.00	−0.00	0.01	−0.04	0.15	−0.06
−0.00	0.00	−0.00	0.00	−0.00	0.00	−0.00	0.02	−0.06	0.24