






LIBRARY
OF THE
MASSACHUSETTS INSTITUTE
OF TECHNOLOGY



Digitized by the Internet Archive
in 2011 with funding from
Boston Library Consortium Member Libraries

**working paper
department
of economics**

The Calculation of Ordinary Least Squares

by

R.E. Hall

MASS. INST. OF TECH.

JUN 18 1968

DEWEY LIBRARY

Number 2 - August 17, 1967

**massachusetts
institute of
technology**

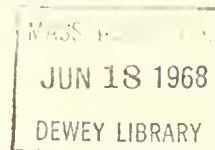
**50 memorial drive
cambridge, mass. 02139**



The Calculation of Ordinary Least Squares

by

R.E. Hall



Number 2 - August 17, 1967

Econometric Working Paper # 2

THE CALCULATION OF ORDINARY LEAST SQUARES ESTIMATES

Although the great bulk of econometric estimates are made by the method of ordinary least squares, very few computer programs for the purpose use algorithms which meet reasonable standards of computational accuracy. In an illuminating study, (3), James Longley presents results he obtained from a variety of regression programs for a typical econometric model. The only program which gave acceptable results was that of the National Bureau of Standards; the algorithm of their program appears in the present study as Algorithm III.

There is a widespread and tenacious misconception among econometricians that the source of difficulty in ordinary least squares calculations lies in the inversion of the matrix of sums of cross-products. Actually, however, the inversion is almost never a source of trouble; furthermore, it has been shown that for positive definite matrices (cross product matrices are always positive definite) the simplest inversion algorithm is also the best (see e.g., Wilkinson[4]).

The principal trouble arises not in inverting the cross-products matrix, but, rather in forming it in the first place. If there is a high degree of collinearity among the independent variables, the inverse matrix, and hence the estimated coefficients, will be extremely sensitive to small changes in the original matrix of sums of cross-products. Even an exact inverse of a cross-products matrix which is insufficiently precise will not be good enough. The reader should consult Wilkinson (4) for an example of a matrix in which the effect on the inverse of a change in one part in 10^8 in the original matrix is 30 times greater than the error introduced by a conventional inversion algorithm.

One method for overcoming this difficulty which has substantial merit is to carry enough digits in the calculation of the cross-products matrix so that the errors in the estimates will be sufficiently small. In machines with double-precision floating-point operations whose speed is close to that of the corresponding single-precision operations, and with problems for which 60 or 70 binary digits is adequate, a fast, rough method such as Algorithm I, below, is probably the optimal method. But where fast double-precision operations are not available, or where 60 or 70 bits are not enough, other algorithms are necessary. As we shall show, there is available a spectrum of least squares algorithms, ranging from the fastest and least accurate (Algorithm I) to the slowest and most accurate (Algorithm IV). As far as is known, Algorithms II and IV have not previously appeared in the literature.

Algorithm I. Direct calculation.

This algorithm involves the straightforward calculation of the algebraic formula for ordinary least squares:

$$(1) \quad b = (X'X)^{-1}X'y \quad ,$$

where X is the $T \times N$ matrix of right-hand variables, y is the $T \times 1$ vector of observed values, and b is the $N \times 1$ vector of estimated coefficients.

The computational steps are

1. Form the lower triangle of $X'X$, the matrix of sums of cross-products, using G2YMLT*.
2. Invert $X'X$, using YINV.
3. Form $X'y$, using GGGMLT.
4. Calculate $b = (X'X)^{-1}X'y$, using GGGMLT.

*The matrix routines referred to here are described in (2).

Algorithm II. One-step orthogonalization.

This algorithm is a generalization of the classical method of transforming the matrix of right-hand variables by subtracting the mean of each variable from the variable before forming the matrix of sums of cross-products.

Before stating this method, we derive a formula which is the basis not only for this algorithm but also for Algorithms III and IV. Suppose S is a nonsingular matrix of order N . We define

$$(2) \quad \tilde{X} = XS^{-1}.$$

Now suppose we calculate the least squares regression coefficients for y and the transformed variables \tilde{X} :

$$(3) \quad \tilde{b} = (\tilde{X}'\tilde{X})^{-1}\tilde{X}'y.$$

By substituting $X = \tilde{X}S$, the reader should be able to verify the following relation between \tilde{b} and b :

$$(4) \quad \tilde{b} = Sb, \text{ or}$$

$$(5) \quad b = S^{-1}\tilde{b}.$$

Furthermore, the inverse matrices are related by

$$(6) \quad (X'X)^{-1} = (S^{-1})(\tilde{X}'\tilde{X})^{-1}(S^{-1})'.$$

The point of these formulas is that by an astute choice of the matrix S , we can deal with a new regression problem which demands much less accuracy in calculating the matrix of sums of cross-products. That is, we should choose S so that the new variables \tilde{X} are much less collinear than the old variables X .

Algorithm II is based on the following assumptions about the economic time series which enter X :

(i) The variables in X have the following decomposition:

$$x_{tj} = \alpha_j d_t + u_{tj} \quad ;$$

d_t is a trend term common to all variables,

α_j is a scale factor (with $\alpha_1 = 1$), and

u_{tj} is a disturbance, not necessarily random.

(ii) The disturbance vectors are approximately orthogonal; i.e.,

$\lim_{T \rightarrow \infty} \frac{1}{T} U'U$ is approximately diagonal, where U is the matrix of disturbances.

(iii) There is one variable, which we number 1, which is almost purely trend (u_{t1} is small relative to the other u's).

If these assumptions hold, there is a simple procedure which makes X roughly orthogonal. We let

$$(7) \quad \tilde{X}_{tj} = X_{tj} - \hat{\alpha}_j X_{t1} \quad \text{for } j = 2, \dots, N \quad \text{and} \quad \tilde{X}_{t1} = X_{t1}$$

where $\hat{\alpha}_j$ is an estimate of α_j .

Then by assumption (ii), the columns of X are approximately orthogonal, as we desired. Assumption (iii) implies that the estimates $\hat{\alpha}_j$ can be obtained by ordinary least squares:

$$(8) \quad \hat{\alpha}_j = \frac{\sum_{t=1}^T X_{tj} X_{t1}}{\sum_{t=1}^T X_{t1}^2} .$$

The transformation can then be written

$$(9) \quad S^{-1} = \begin{bmatrix} 1 & -\hat{\alpha}_2 & -\hat{\alpha}_3 & \dots & -\hat{\alpha}_N \\ & 1 & 0 & & \\ & & \cdot & & \\ & & & \cdot & \\ 0 & & & & 1 \end{bmatrix} ;$$

it amounts to replacing each variable by the residuals from the regression of that variable on the first variable, if the first variable, X_1 , is taken to be the constant vector,

$$X_1 = \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ 1 \end{bmatrix} ,$$

Algorithm II is the same as the classical method of subtracting means. The generalization of the classical method presented here has two advantages: First, if all of the variables of a problem in fact share the same trend, this algorithm is more accurate than the classical one. Second, and more important, this method can be used on equations which do not have constants. The classical method cannot be used on such problems; as a result, computer programs based on it which give satisfactory results for equation with constants may break down if run on the same problem with the constant suppressed.

Summary of steps for Algorithm II:

1. Calculate the vector $\hat{\alpha}$ and form \tilde{X} , using the special routine ORTHOG*.
2. Carry out all of the steps of Algorithm I to obtain \tilde{b} and $(\tilde{X}'\tilde{X})^{-1}$.

* Special routines are described in the appendix.

3. Calculate b and $(X'X)^{-1}$, using the special routine UNTRAN.

Algorithm III. Gram-Schmidt Orthonormalization.

This method is based on the Gram-Schmidt orthonormalization algorithm, which simultaneously calculates \tilde{X} and S^{-1} from X so that $\tilde{X} = XS^{-1}$ and

$$(10) \quad \tilde{X}'\tilde{X} = I \quad .$$

In this case the intermediate estimate \tilde{b} is given by the simple formula

$$(11) \quad \tilde{b} = \tilde{X}'y \quad .$$

The Gram-Schmidt process operates recursively on the columns of X , replacing each column in turn by the residuals from the regression of that column on all of the previously transformed columns. Each column is also normalized so that it has unit sum of squares. The upper-triangular matrix S^{-1} is formed simultaneously by applying the same transformation to the columns of a matrix which is initially set equal to the identity matrix.

Steps in Algorithm III:

1. Calculate \tilde{X} and S^{-1} from X , using the special routine ORTHOS.
2. Calculate $\tilde{b} = \tilde{X}'y$, using GGGMLT.
3. Calculate $b = S^{-1}\tilde{b}$, using TGGMLT.
4. Calculate $(X'X)^{-1}$ as $(S^{-1})(S^{-1})'$, using T2YMLT.

Algorithm IV. Two-pass orthonormalization.

This method is closely related to the method of Algorithm III; however, it is somewhat more accurate and is about half again slower. Its principal advantage is that it requires only two passes over the data matrix X, while Algorithm III requires N passes. Thus Algorithm IV is particularly suited to problems in which T is so large that X must reside in secondary storage.

The basic principal of Algorithm IV is that equations (2) to (5) hold for any nonsingular matrix S, not just for an orthonormalizing matrix. Furthermore, the transformation

$$\tilde{X} = XS^{-1}$$

can be carried out with fairly high accuracy. Thus even if S is only a poor approximation to the true orthonormalizing matrix as long as it reduces the collinearity in X this method will yield more accurate results.

To calculate the matrix S in one pass through the data, we use the following method: If A is a positive definite matrix, then Choleski's method (see [2]) can be used to calculate an upper-triangular matrix S such that

$$(12) \quad A = S'S \quad .$$

If A is taken to be the matrix of sums of cross-products, then we can find an S such that

$$(13) \quad X'X = S'S \quad .$$

Then, theoretically,

$$\begin{aligned}
 (14) \quad \tilde{X}'\tilde{X} &= (S^{-1})'X'X(S^{-1}) \\
 &= (S^{-1})'S'S(S^{-1}) \\
 &= I \quad ,
 \end{aligned}$$

or the inverse of the matrix generated by Choleski's factorization method orthonormalizes the matrix X.

Since the matrix X'X is generally formed inaccurately, the matrix S will not, in fact, orthonormalize X. As we have indicated above, this is not a problem in itself; we simply calculate \tilde{b} as $(\tilde{X}'\tilde{X})^{-1}\tilde{X}'y$ instead of dropping the calculation and inversion of $\tilde{X}'\tilde{X}$, as we did in Algorithm III.

Steps in Algorithm IV:

- | | |
|---|--|
| <ol style="list-style-type: none"> 1. Form X'X, using G2YMLT. 2. Calculate S, using YFACT. 3. Calculate S⁻¹, using TINV. 4. Calculate \tilde{X}, using GTGMLT. 5. Calculate \tilde{b}, using the steps of Algorithm I. 6. Calculate b using TGGMLT. 7. Calculate (X'X)⁻¹ using TGGMLT. | <div style="font-size: 4em; vertical-align: middle;">}</div> <p style="margin-left: 10px;">Done by ORTHON.</p> |
|---|--|

Table I presents a comparison of the speeds of the four algorithms. In the first column we give a rough formula for the number of multiplications required -- an equal number of additions is also required. The formulas were obtained by dropping all terms which did not involve either T or the highest power of N. Greater accuracy would be spurious since actual relative timings will depend

on the particular machine and program in use. In the second column we give the limits as T becomes large of the speeds of Algorithms II, III, and IV relative to Algorithm.I. Finally in column 3 we give the relative speeds for the typical case N = 7 and T = 70.

Table I

Algorithm	General Formula	lim T ∞ , relative to I.	N = 7 T = 70
I	$\frac{TN^2}{2} + TN + \frac{N^3}{2}$	1	1.0
II	$\frac{TN^2}{2} + 3TN + \frac{N^3}{2}$	$\frac{N+2}{N+6}$.71
III	$TN^2 + TN + \frac{N^3}{3}$	$\frac{1}{2}$.59
IV	$\frac{3TN^2}{2} + TN + \frac{5N^3}{6}$	$\frac{1}{3}$.40

Appendix. FORTRAN subroutines to implement the algorithms.

1. REGCL1, REGCL2, REGCL3, and REGCL4.

These routine calculate ordinary least squares estimates using Algorithms I, II, III, and IV respectively. They all have the same calling sequence:

```
CALL REGCL1(NOB,NOIN,NOVAR,Y,Z,X,YFIT,V,S,D) .
```

NOB Number of observations.

NOIN Number of instrumental variables. If NOIN = 0, ordinarily least squares estimates are claculated. Algorithms for instrumental estimates are described in a separate paper.

NOVAR Number of right-hand variables.

Y Left-hand variable. Length = NOB.

Z Matrix of right-hand variables. Length = NOB^*NOVAR .

X Matrix of instrumental variables; must appear in calling sequence even if NOIN=0. Length = NOB^*NOIN .

YFIT Returns fitted values; also used for intermediate results in instrumental estimates. Length = $\max(NOB,NOIN^*NOB)$.

V Contains inverse of cross-products matrix on return. Uded for intermediate results. Length = $\max(NOVAR^{**}2,NOIN^*[NOIN+1])$.

S Inverse of S-matrix. Length = $NOVAR^{**}2$, upper triangle only.

D Estimated coefficients. Used for intermediate results. Length = $\max(NOVAR,NOIN)$.

2. ORTHOG

This routine carries out the one-step orthogonalization described under Algorithm II.

Calling sequence: CALL ORTHOG(NOVAR,NOB,X,S)

NOVAR Number of variables.

NOB Number of observations.

X Data matrix to be transformed. Length = NOVAR*NOB.

S First row of transformation matrix. Length = NOVAR.

3. UNTRAN.

This routine calculates b and V from b, V, and S^{-1} for Algorithm II.

Calling sequence: CALL UNTRAN(NOVAR,B,V,S).

B Vector of estimates to be transformed. Length = NOVAR.

V Inverse of the cross-products matrix to be transformed.

Both upper and lower triangles are transformed. Length
= NOVAR**2.

Other arguments are the same as for ORTHOG.

4. ORTHOS.

This routine carries out the Gram-Schmidt orthonormalization using the method presented by Davis and Rabinowitz in (1), with the modification that the normalization of each column is carried out as a separate step after the orthogonalization. This modification has been found to be crucially important.

Calling sequence: CALL ORTHOS(NOVAR,NOB,X,V,S)

NOVAR Number of variables.

NOB Number of observations.

X Data matrix to be transformed. Length = NOVAR*NOB.

V Vector of length NOVAR used for intermediate results.

S Upper triangular matrix of the transformation
(i.e., S^{-1} in the notation of this paper); Length = NOVAR**2.

5. ORTHON.

This routine calls G2YMLT, YFACT, and TINV to carry out the data transformation of Algorithm IV.

Calling sequence: CALL ORTHON(NOVAR,NOB,X,V,S).

All of the arguments are the same as for ORTHOS, except that V is a lower triangular matrix which may interlace S; that is, CALL ORTHON(NOVAR,NOB,X,V,V[NOVAR+1]) is usually the way that it is used.

Date Due

~~DEC 21 '76~~

JUL 20 '76

SEP 08 '76

~~JAN 12 '77~~

AUG 04 '79

~~JUN 14 '79~~

~~JUL 17 1980~~

1985

AUG 23 1985

MIT LIBRARIES



3 9080 003 958 771

MIT LIBRARIES



3 9080 003 927 651

MIT LIBRARIES



3 9080 003 958 664

MIT LIBRARIES



3 9080 003 958 714

MIT LIBRARIES



3 9080 003 958 623

MIT LIBRARIES



3 9080 003 958 763

MIT LIBRARIES



3 9080 003 927 693

MIT LIBRARIES



3 9080 003 958 839

MIT LIBRARIES

DUPL



3 9080 003 927 669

MIT LIBRARIES



3 9080 003 927 677

[.3.]

en e

