Linear programming

1979

UCD Department of Agricultural Economics

WORKING PAPER SERIES

University of California, Davis
Department of Agricultural Economics

A SELF-DUAL PROBLEM FOR CHECKING THE RELIABILITY OF LINEAR AND QUADRATIC PROGRAMMING COMPUTER CODES

by

Q. Paris

Working Paper No. 79-10

A SELF-DUAL PROBLEM FOR CHECKING THE RELIABILITY OF LINEAR AND QUADRATIC PROGRAMMING COMPUTER CODES

I. Introduction. To date, the approach used in checking the reliability of computer subroutines for solving linear programming (LP) problems is based on a battery of tests problems whose solutions were obtained either by desk calculator or by other "well established" computer codes. This approach leaves much to be desired because it provides only a test of relative consistency. There remains always the possiblity that the same mistake is made by all the computer codes used to solve the given test problems and against whose solutions one wishes to verify the correctness of a newly developed computer subroutine.

In place of a relative consistency check one would obviously wish to use an absolute consistency criterion for judging the reliability of the computer code. Such an absolute consistency criterion must clearly be provided internally by the structure of the problem to be solved and whose solution one attempts to achieve by means of the computer code at hand. The great advantage of such a criterion -- if one can be found -- is that the programmer can dispense with the relative comparison of test problems' solutions obtained also by other means or methods.

With regard to linear programming, the absolute consistency criterion is provided by problems characterized by the property of self-duality. Very simply, problems of this type exhibit the unique feature that the primal solution is identical to the dual solution. Therefore, if one does not encounter such a correspondence in the solution of a self-dual linear programming problem, the inescapable conclusion is that the computer code employed is not reliable. Using this criterion, it was recently discovered that three out of five LP computer codes provide incorrect answers.

In what follows, we first describe the structure of a self-dual linear programming problem. The topic was first discussed by Duffin [4] whose paper seems to be the only reference on the subject. The self-dual problem discussed here, however, is more general than any of the examples presented by Duffin. It turns out that this problem is directly related to the least-squares problem restricted by linear inequalities. We then present a numerical example used to verify the reliability of five LP computer codes. We report in detail the performance of each computer code vis-a-vis the numerical example. We conclude the discussion with a conjecture as to the possible origin of the programs' errors and with suggestions for extending the self-duality criterion to other mathematical programming computer codes.

2. <u>A self-dual linear program</u>. Consider the following linear programming problem:

Maximize

$$d'\pi - b'\psi \tag{2.1}$$

subject to

$$Dx + \pi = d \tag{2.2}$$

$$D'\pi - A'\psi = 0 \tag{2.3}$$

$$-Ax \geq -b \tag{2.4}$$

$$\psi \geq 0, \text{ x and } \pi \text{ unrestricted,}$$

where D and A are matrices of dimensions p x n and d x n, respectively. The rank of such matrices is arbitrary. The vector $d\epsilon R^p$ while the vector $b\epsilon R^m$. The vector variable ψ is required to be nonnegative, while the vector variables x and π are free. This means that the implementation of problem (2.1) - (2.4) using a standard LP computer code requires that each of the vectors x and π be defined as the difference of two nonnegative vectors such as $x = x_1 - x_2$, $\pi = \pi_1 - \pi_2$, where all the components of x_1, x_2, π, and π_2 are nonnegative.

The principal characteristic of problem (2.1) - (2.4) is that it is self-dual, that is, its dual prossesses the identical structure of the primal and, therefore, the primal solution(s) is (are) identically equal to the dual solution(s).

To demonstrate this proposition we form the Lagrangean function of problem (2.1) - (2.4) using the symbols u, y and ϕ to indicate the Lagrange multiplier (dual variable) vectors associated with constraints (2.2), (2.3) and (2.4), respectively. Thus,

$$L = d'\pi - b'\psi + u'[d - \pi - Dx] + y'[A'\psi - D'\pi] + \phi'[Ax - b]. \tag{2.5}$$

It is well known that the minimization of (2.5) with respect to u, y and ϕ, subject to the appropriate constraints constitutes the dual of problem (2.1) - (2.4). It is also well known that the appropriate constraints correspond to the Kuhn-Tucker conditions represented by the derivatives of the Lagrangean function (2.5) with respect to the primal variables π, x, and ψ. Such conditions are:

$$\partial L/\partial\pi = d - u - Dy = 0 \tag{2.6}$$

$$\partial L/\partial x = -D'u + A'\phi = 0 \tag{2.7}$$

$$\partial L/\partial\psi = -b + Ay \leq 0 \tag{2.8}$$

$$\pi'(\partial L/\partial \pi) = \pi'd - \pi'u - \pi'Dy = 0 \tag{2.9}$$

$$x'(\partial L/\partial x) = -x'D'u + x'A'\phi = 0 \tag{2.10}$$

$$\psi'(\partial L/\partial \psi) = -\psi'b + \psi'Ay = 0. \tag{2.11}$$

It is clear that conditions (2.6), (2.7) and (2.8) are -- except for the symbols used to indicate the vector variables -- identical to the primal constraints (2.2), (2.3) and (2.4), respectively. Also, by using (2.9), (2.10) and (2.11) in (2.5), we can restate in a simplified fashion the objective function of the dual problem which becomes

$$\min L = d'u - b'\phi. \tag{2.12}$$

Hence, relation (2.12) subject to conditions (2.6), (2.7) and (2.8), $\phi \geq 0$, u and y unrestricted, constitutes the dual of problem (2.1) - (2.4). It is thus, obvious that -- if a solution exists -- we must have $\pi = u$, $x = y$ and $\psi = d$. This result proves our assertion that it is sufficient to inspect the numerical findings corresponding to the primal and dual solut-tions for deciding whether the correct result was obtained.

3. Inequality restricted least-squares problems.

The structure of the LP problem (2.1) - (2.4) is intimately related to that of the least-squares (LS) problem restricted by linear inequalities. Such a problem has always been considered only as a quadratic program (QP). Problems of this type were also discussed in this Journal by Stoer [7] who presented a solution algorithm closely related to Rosen's gradient projection method. It turns out, however, that least-squares problems of any sort can be solved by linear programming and, therefore, one can add the simplex algorithm to the list of least-squares solution procedures. To demonstrate the correspondence between the inequality restricted LS problem and LP structure given in (2.1) - (2.4) we follow Stoer [7] in defining the following least-squares primal problem:

Minimize

$$u'u/2 \tag{3.1}$$

subject to

$$Dx + u = d \tag{3.2}$$

$$Ax \leq b \tag{3.3}$$

$$x, u \text{ unrestricted.}$$

This quadratic programming problem possesses a dual representation which corresponds exactly to the linear programming structure (2.1) - (2.4). To demonstrate this proposition let us choose dual variables (Lagrange multipliers) π and ψ corresponding to constraints (3.2) and (3.3), respectively. The Lagrangean function of problem (3.1) - (3.3) is, then, defined as

$$L = u'u/2 + \pi'[d - Dx - u] + \psi'[Ax - b]. \tag{3.4}$$

The Kuhn-Tucker conditions associated with the solution of problem (3.1) - (3.3) are as follows:

$$\partial L/\partial u = u - \pi = 0 \tag{3.5}$$

$$\partial L/\partial x = -D'\pi + A'\psi = 0 \tag{3.6}$$

$$\partial L/\partial \pi = d - Dx - u = 0 \tag{3.7}$$

$$\partial L/\partial \psi = Ax - b \leq 0 \tag{3.8}$$

(3.9) $$u'(\partial L/\partial u) = \pi'\pi - \pi'u = 0$$

(3.10) $$x'(\partial L/\partial x) = -x'D'\pi + x'A'\psi = 0$$

(3.11) $$\pi'(\partial L/\partial \pi) = \pi'd - \pi'Dx - \pi'u = 0$$

(3.12) $$\psi'(\partial L/\partial \psi) = \psi'Ax - \psi'b = 0.$$

Notice that in (3.5) we re-encounter very explicitly the self-duality condition concerning the residual vector according to which $u = \pi$. Hence, using (3.5), the relations (3.6), (3.7) and (3.8) correspond exactly to the constraints (2.3), (2.2) and (2.4), respectively. To show that the dual objective function of problem (3.1) - (3.3) is indeed given by the objective function of the linear programming problem (2.1) it is sufficient to substitute, as convenient, the relations (3.5) - (3.12) into the Lagrangean function (3.4). When this is done, one obtains.

(3.13) $$\begin{aligned} \max L &= u'u/2 + \pi'd - \pi'Dx - \pi'u + \psi'Ax - \psi'b \\ &= \pi'd - \psi'b - \pi'u/2 && \text{by (3.5) and (3.10)} \\ &= \pi'd - \psi'b - \pi'(d - Dx)/2 && \text{by (3.7)} \\ &= \pi'd/2 - \psi'b + x'A'\psi/2 && \text{by (3.10)} \\ &= \pi'd/2 - \psi'b + \psi'b/2 && \text{by (3.12)} \\ &= \pi'd/2 - \psi'b/2, \end{aligned}$$

and the proposition is proved.

The peculiar dual structure of the least-squares problem provides a very strong consistency check as to whether the correct solution of a given LS problem was indeed obtained: not only one possesses the internal check given by solving the linear programming (2.1) - (2.4) where the entire primal solution vector must be equal to the entire dual vector; one can also verify the numerical results by solving the associated quadratic programming (3.1) - (3.3), where, also there, $u = \pi$. Hence, the self-duality of least-squares problems can further be used as an <u>absolute</u> consistency criterion for verifying <u>also</u> the reliability of quadratic programming computer codes.

4. <u>An application of the self-dual consistency criterion.</u> The data presented in Table 1 were used to implement the approach outlined in sections 2 and 3. The regression line was postulated to be linear in the logarithms of the data of Table 1. Hence, the various components of the least-squares problem are as follows:

$$D = \begin{pmatrix} 1.0 & 12.668639 & 11.559819 \\ 1.0 & 12.714335 & 11.572778 \\ 1.0 & 12.746089 & 11.600890 \\ 1.0 & 12.749475 & 11.547549 \\ 1.0 & 12.801257 & 11.589460 \\ 1.0 & 12.839963 & 11.730783 \\ 1.0 & 12.872704 & 11.618222 \\ 1.0 & 12.858117 & 11.573550 \\ 1.0 & 12.904324 & 11.614868 \\ 1.0 & 12.929516 & 11.625191 \end{pmatrix}, \quad d = \begin{pmatrix} 12.063345 \\ 12.094381 \\ 12.134920 \\ 12.125331 \\ 12.211025 \\ 12.234200 \\ 12.255458 \\ 12.234174 \\ 12.307659 \\ 12.335314 \end{pmatrix}$$

The inequality restrictions were chosen as

$$A = \begin{pmatrix} 0.0 & 1.0 & -3.0 \\ 0.0 & 1.0 & 1.0 \end{pmatrix}, \qquad b = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}.$$

The five linear programming computer codes tested by means of this numerical example are (a) Burroughs' "TEMPO", (b) Lawrence Berkeley Laboratory's (LBL) "GUMPS", (c) Berkeley Computer Center's (BCC) "ALPHAC",

(d) IBM's 1130 LP-MOSS, and (e) IBM's 370 MPSX. All these codes have compatible input data requirements.

Burroughs' "TEMPO". This is a computer code copyrighted by Burroughs Corporation since 1971 and revised in 1972, 1973, 1974 and 1975. It is available only through computer centers equipped with Burroughs 7700 or 6700 systems. The corporation makes available only the object deck. The User's Manual [2] does not specify the language of the original code. The Computer Center of the University of California at Davis, where "TEMPO" is available, owns a Burroughs 6700 system.

The results of the test are presented in Table 2, Column 1. Although the code's exit condition asserts that it has encountered an optimal solution, the primal and dual solution vectors are entirely different. Furthermore, it does not seem plausible to invoke rounding errors as the cause of the discrepancy. The inevitable conclusion is that "TEMPO" constitutes an unreliable LP computer code.

Lawrence Berkeley Laboratory's "GUMPS". This subroutine was written for use at the Lawrence Berkeley Laboratory which is equipped with 7600, 6600 and 6400 CDC hardware. Although the write-up of the program [8] explicitly states that the Laboratory has been "reluctant to provide a user's guide while the program is still undergoing many changes and additions -- and an occasional correction", this computer code has been in operation for more than three years. GUMPS test results obtained on the LBL's CDC computer network are presented in Table 2, Column 2. Numerically, they are identical to those obtained by TEMPO, indicating that also GUMPS gives incorrect answers.

Berkeley Computer Center's "ALPHAC". This code [1] was written for the Berkeley campus computer center in 1972 and revised in 1973 and 1975. The coding languages are Fortran IV and Compass compiled on a CDC 6400 series machine. ALPHAC'S findings, obtained at the Berkeley Computer Center, are presented in Table 2, Column 3. Again they indicate that the ALPHAC code is not reliable. The numerical results -- except for the sign of the y_i's -- are identical to those of TEMPO and GUMPS.

IBM 1130 LP-MOSS. This code [5] is copyrighted by IBM Corporation and is intended for use on the IBM 1130 system. The code is, to a large extent, compatible with the IBM MPS 360 and IBM MPSX 370 systems. The test was carried out on the 1130 machine at the Department of Agricultural and Resource Economics of the University of California, at Berkeley. The results are presented in Table 2, Column 4, and indicate that the LP-MOSS computer code passes the reliability test. Although only three decimal digits are printed by the output subroutine, it is clear that the entire primal solution vector is equal to the entire dual solution vector, as anticipated by the theory. Furthermore, the value of the optimal objective function is more than three times smaller than that obtained by TEMPO, GUMPS and ALPHAC.

IBM MPSX. This code [6] is supplied by IBM for its 370 system. One would expect to obtain results similar to those computed with the IBM 1130 LP-MOSS code. Indeed -- as it can be seen from Table 2, Column 5 -- the numerical information on the primal and dual solutions indicates that the correct optimal solution was achieved. Hence, also MPSX passes the reliability test.

Rand Quadratic Programming Code. This is a famous subroutine [3] since it is a conversion of an earlier Rand QP subroutine co-planned by Philip Wolfe, written for the US Air Force, and coded QPF4. It is completely written in Fortran IV, explicitly for the IBM 360/65. It can be easily adapted to other machines with minor modifications. For the present study it was implemented on the Burroughs 6700 system of the University of California, at Davis.

In sections 2 and 3 it was explained that the LP specification of least-squares problems is self-dual, a property exhibited also by their quadratic programming formulation. Hence, the same test problem was used to verify the reliability of the Rand QP computer code. The results are encouraging and are presented in Table 2, Column 6. The solution obtained corresponds to that computed by the IBM 1130 LP-MOSS subroutine. The fact that, once again, $u = \pi$, indicates that the Rand QP code has passed this reliability test.

5. A conjecture. The source decks of all programs used in this test were unavailable. Speculating as to the probable reasons for the unreliable behavior of subroutines TEMPO, GUMPS, and ALPHAC, therefore, becomes a guessing game. As a conjecture, one may advance the hypothesis that rounding errors have set in all the programs that misbehaved. If this were true, the logical action would be to increase the precision of the computations by, for example, instructing the program to operate in "double precision". This strategy, however, is much easier to formulate than to carry out. Essentially, it amounts to rewriting many sections of the (unavailable) codes.

Another possibility is to study the various tolerances employed in the programs and decide whether their present levels might be the cause of the problem. In Table 3 we have summarized the main tolerances' values employed by the six computer codes tested in this exercise. Except with regard to LP-MOSS subroutine there seems to be no great difference among the various codes. TEMPO, however, sets a tolerance for the reduced cost and the primal solution which is higher than all other subroutines.

In order to verify the hypothesis that the cause for the unreliability of TEMPO, GUMPS and ALPHAC codes is different from a problem of rounding-off errors, we performed the following experiment: the tolerance value of "reduced cost" was first increased to .01. This action was taken on the basis that the algorithm was inserting the wrong primal activities into the basis because the corresponding reduced costs were encountered to be significantly negative. When this was done the program approached the correct optimal value of the objective function but the primal solution was still very different from the dual solution. A second experiment was carried out setting the tolerance value of the "reduced cost" to .1. In this case, TEMPO was able to achieve the correct optimal primal solution and the optimal value of the objective function but, quite naturally, almost all the dual variables exhibited a misleading zero value. Of course, a tolerance value of .1 is absurd and the exercise was carried out to make sure that the main troubles are algorithmic rather than generated by rounding errors.

6. Conclusion. It may come as a surprise that three out of five LP computer codes were found unreliable. The surprise, however, will soon

disappear if one realizes that a self-dual linear programming problem of some generality was formulated only very recently. Without such a specification with its own stringent consistency conditions, it is not easy to realize that an "optimal" solution like that generated by either TEMPO or ALPHAC or GUMPS is not optimal. The output printouts appear to be completely plausible, and even when tested against each other, the two computer codes cannot be detected in error.

Since none of the source programs were available to the author, little can be said about the causes of unreliability of TEMPO, GUMPS, and ALPHAC. It is, however, possible to conjecture that the trouble area for the three codes must be associated with the treatment of slack and artificial variables. This conclusion is obvious if one studies the incorrect solutions: the two inequality constraints which in the correct optimal solution should be binding, are actually satisfied by slack variables.

The importance and the widespread use of linear and quadratic programming models for representing engineering, biological and economic problems, warrants a thorough testing of the computer codes in circulation. This paper has offered a very powerful test of consistency which ought to be added to the traditional battery of test problems.

8/27/79:cfg

REFERENCES

[1] BERKELEY COMPUTER CENTER, ALPHAC, A Linear Programming System, University of California, Berkeley, CA, 1972, 1973, 1975.

[2] BURROUGHS CORPORATION, B7700/B6700 Systems TEMPO, Mathematical Programming System, User's Manual 1073665, Detroit, MI, 1975.

[3] L. CUTLER AND D. S. PASS, A Computer Program For Quadratic Mathematical Models to be Used for Aircraft Design and Other Applications Involving Linear Constraints, R-516-PR, Rand Corporation, Santa Monica, CA, 1971.

[4] R. J. DUFFIN, Infinite Programs, in Linear Inequalities and Related Systems, H. W. Kuhn and A. W. Tucker, Editors, Princeton University Press, Study 38, Princeton, NJ, 1956, pp. 157-170.

[5] IBM CORPORATION, 1130 Linear Programming - Mathematical Optimization Subroutine System, (1130LP-MOSS) (1130-CO-16X), 1967.

[6] IBM CORPORATION, Mathematical Programming System, MPSX User's Manual, 1967.

[7] J. STOER, "On the Numerical Solution of Constrained Least-Squares Problems", Siam J. Numer. Analysis, 8 (1971), pp. 382-411.

[8] LAWRENCE BERKELEY LABORATORY, User's Manual GUMPS, Berkeley, CA, 1977.

TABLE 1

THE DATA

Dependent Variable	Regressor #1	Regressor #2
173,398	317,629	104,801
178,864	332,480	106,168
186,264	343,207	109,195
184,486	344,371	103,523
200,993	362,673	107,954
205,730	376,986	124,341
210,125	389,533	111,104
205,700	383,892	106,250
221,385	402,047	110,732
227,593	412,304	111,881

TABLE 2

TEST RESULTS OF SIX COMPUTER CODES

	Burroughs "TEMPO" 1	LBL "GUMPS" 2	BCC "ALPHAC" 3	IBM 1130LP-MOSS 4	IBM MPSX 5	Rand QP 6
Exit Condition	Optimal	Optimal	Optimal	Optimal	Optimal	Optimal
Value of Obj. Function	.01679	.01679	.01679	.005	.00525	.00525
PRIMAL SOLUTION						
x_1	12.19959	12.19960	12.19959	-.307	-.30757	-.30757
x_2	.00000	.00000	.00000	.750	.75000	.75000
x_3	.00000	.00000	.00000	.249	.25000	.25000
π_1	-.13625	-.13625	-.13625	-.020	-.02052	-.02052
π_2	-.10521	-.10521	-.10521	-.026	-.02700	-.02700
π_3	-.06467	-.06467	-.06467	-.017	-.01730	-.01730
π_4	-.07426	-.07426	-.07426	-.016	-.01610	-.01610
π_5	.01143	.01143	.01143	.020	.02028	.02028
π_6	.03473	.03473	.03473	-.020	-.02078	-.02078
π_7	.05587	.05587	.05587	.003	.00394	.00394
π_8	.03458	.03458	.03458	.004	.00477	.00477
π_9	.10807	.10807	.10807	.033	.03326	.03326
π_{10}	.13572	.13572	.13572	.039	.03945	.03945
ψ_1	.01219	.01219	.01219	.003	.00355	--
ψ_2	.05829	.05829	.05829	.001	.01154	--
DUAL SOLUTION						
y_1	-.30757	-.30757	.30757	-.307	-.30757	--
y_2	.75000	.75000	.75000	.750	.75000	--
y_3	.25000	.25000	.25000	.250	.25000	--
u_1	-.02052	-.02052	-.02052	-.020	-.02052	-.02052
u_2	-.02700	-.02700	-.02700	-.026	-.02700	-.02700
u_3	-.01730	-.01730	-.01730	-.017	-.01730	-.01730
u_4	-.01610	-.01610	-.01610	-.016	-.01610	-.01610
u_5	.02028	.02028	.02028	.020	.02028	.02028
u_6	-.02078	-.02078	-.02078	-.020	-.02078	-.02078
u_7	.00394	.00394	.00394	.003	.00394	.00394
u_8	.00477	.00477	.00477	.004	.00477	.00477
u_9	.03326	.03326	.03326	.033	.03326	.03326
u_{10}	.03945	.03945	.03945	.039	.03945	.03945
ϕ_1	-.00030	.00000	.00000	.003	.00355	.00355
ϕ_2	.00000	.00000	.00000	.011	.01154	.01154

TABLE 3

TOLERANCES FOR SIX COMPUTER CODES

Tolerance On	Burroughs "TEMPO"	LBL "GUMPS"	BCC "ALPHAC"	IBM 1130LP-MOSS	IBM MPSX	Rand QP
	1	2	3	4	5	6
Reduced Cost	.0001	.000001	.000001	--	.00001	.00001
Primal Solution	.0001	.000001	.000001	.0001	.00001	.00001
Pivot	.00001	.000001	.00001	.0005	.000001	.00001
Check on DJ Coeff.1/	.001	--	--	--	--	--
Screen2/	--	--	--	.001	--	--

1/ Tolerance for computational error based upon the reduced cost (DJ) values computed by forward and backward transformations.

2/ After the input data has been scaled, any number whose magnitude is below this tolerance is ignored.