ANL--82-73

DE83 004400

Distribution Category:
Mathematics and Computers
(UC-32)

ANL-82-73

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439

**RANDOMLY GENERATED TEST PROBLEMS
FOR POSITIVE DEFINITE QUADRATIC PROGRAMMING***

by

Melanie L. Lenard** and Michael Minkoff

Mathematics and Computer Science Division

October 1982

* This work was supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-ENG-38.

** Permanent Address: Boston University, School of Management, Boston, Massachusetts 02215

TABLE OF CONTENTS

Abstract.	5
1. Introduction.	6
2. General Description of Test Problems.	8
3. The Objective Function.	10
4. The Constraints	11
5. Sample Use of Test Problems	14
6. Conclusions	16
References.	18
Acknowledgements.	19

Abstract

A procedure is described for randomly generating positive definite quadratic programming test problems. The test problems are constructed in the form of linear least squares problems subject to linear constraints. The probability measure for the problems so generated is invariant under orthogonal transformations. The procedure allows the user to specify the size of the least squares problem (number of unknown parameters, number of observations, and number of constraints); the relative magnitude of the residuals; the condition number of the Hessian matrix of the objective function; and the structure of the feasible region (number of equality constraints and the number of inequalities which will be active at the feasible starting point and at the optimal solution). An example is given illustrating how these problems can be used to evaluate the performance of a software package.

1. INTRODUCTION

The process of evaluating optimization software has been the subject of considerable study in the past several years. Interest in this subject has increased as a consequence of the development of a wide range of algorithms and related software packages and the proliferation of often conflicting claims of relative superiority for various algorithms and software packages. The subject has gained relevance as sophisticated optimization techniques have come to be employed in production environments, leading to a growing need for assistance in software package selection. Furthermore, evaluating mathematical software is an inherently complex process, involving such factors as user-friendliness, robustness, reliability, accuracy, and speed [5].

Approaches to testing optimization software have generally fallen into three categories [14]:

- (1) battery testing -- in which software is tested on a set of specific problems;
- (2) random problem testing -- in which software is tested on a set of problems which are randomly generated;
- (3) performance profile testing -- in which software is tested for a single specific capability, e.g. descending through a curved valley.

In each of these categories of testing, the objective is to represent the spectrum of problems that one is likely to encounter in actual use of the optimization software. Studies in these areas include, but are not limited to, the work in [3,6,7,8,9,11,12,13,15,16].

In this paper we will develop quadratic programming problems which will combine aspects of the random and profile approach. We shall impose specific structure on the problem by specifying, for example, the condition number or size of the active constraint set, before allowing the remaining data to be randomly generated. We will deal with quadratic programming (QP) problems since they provide a minimal extension of linear programming problems yet they involve some of the difficulties in generating nonlinear programming problems. These QP problems have the advantage of requiring only a limited number

of parameters to specify the problem, i.e. the constraint normals and the coefficients for the linear and quadratic terms in the objective function. Thus the issue of representing general nonlinear functions is avoided. Further, since only linear constraints occur in these problems, they provide a convenient basis for testing active set strategies, e.g. [11], without requiring the treatment of the more complex issue of active set strategies for nonlinear constraints.

We further restrict the class of test problems to those which have a strictly convex objective function, i.e. where the constant Hessian matrix is positive definite. Thus each problem has a unique optimum point. Moreover, these positive definite QP problems are equivalent to linearly constrained linear least-squares problems [10]. From this alternative viewpoint, which we will use below in problem generation, the problem is one in data-fitting, where the objective is finding the best solution in the least-squares sense of $p \geq n$ linear equations in n variables subject to m linear constraints.

When generating random problems, care must be taken to avoid imposing structure on the problem unintentionally. For example, as is observed in [20], if we generate coefficients for the linear constraints using a uniform distribution, some angles occur more frequently than they should. This structure can be removed by generating the coefficients from any spherically symmetric distribution (such as the normal distribution). Also, if we generate a Hessian matrix using normally distributed variates, it is very likely that we will obtain a well-conditioned problem [1]. This difficulty can be overcome by generating the matrix in terms of its singular value decomposition.

In Section 2 we describe the class of test problems to be constructed. Section 3 gives further details concerning the objective function while Section 4 deals with the constraints. In Section 5 we illustrate the use of this problem generator in evaluating a software package. Finally, Section 6 provides concluding remarks about this approach to testing optimization software.

2. GENERAL DESCRIPTION OF TEST PROBLEMS

The positive definite quadratic programming test problems are generated in the form of linearly constrained linear least squares problems; that is,

$$\begin{aligned} & \min \|Cx-d\|^2 \\ & \text{subject to:} \\ & Ax \rho b \end{aligned} \tag{2.1}$$

where A is an $m \times n$ matrix, b is an m -vector, C is a $p \times n$ matrix, d is a p -vector, $\|\cdot\|$ denotes the usual Euclidean norm, and ρ is a vector of relational operators chosen from the set $\{\leq, =, \geq\}$. Thus, generating one test problem is equivalent to generating values for every element of A , b , C , d , and ρ .

The problem (2.1) is equivalent to the following QP problem

$$\begin{aligned} & \min x'Qx + p'x + r \\ & \text{subject to:} \\ & Ax \rho b \end{aligned} \tag{2.2}$$

where $Q = C'C$, $p = -2d'C$, $r = d'd$, and $(\cdot)'$ denotes transpose.

The problems are generated to have a known optimal solution. In particular, letting x^* denote the n -vector of optimal values for x and letting λ^* denote the k_1 -vector of optimal values for the Lagrange multipliers corresponding to the k_1 constraints ($1 \leq k_1 \leq n$) which are active (binding) at the optimal solution, we arbitrarily choose $x_i^* = 1$, $i=1, \dots, n$; and $|\lambda_j^*| = 1$, $j = 1, \dots, k_1$. This choice of x^* and λ^* is special in that it represents a very well (in fact, perfectly) scaled solution. Other choices are of course possible, and we will show how they could be incorporated when we examine the details of test problem generation in the subsequent two sections.

The generated problems also have one known feasible point, x^0 , which can be used as a starting point. We arbitrarily choose the origin; that is, $x_i^0 = 0$, $i=1, \dots, n$.

The user is expected to specify:

- 1) the number of unknown parameters (n);

- 2) the number of observations (p);
- 3) the condition number of C (denoted by t); (Note: the condition number of the Hessian of the equivalent QP problem is t^2 .)
- 4) the ratio of the final objective function value to the initial objective function value (denoted by γ^2 , $0 < \gamma^2 < 1$);
- 5) the number of constraints of various kinds, namely,
 - a) the number of equality constraints μ_0 ,
 - b) the number of inequality constraints which are
 - i) active at the starting point x^0 (μ_1)
 - ii) active at the solution point x^* (μ_2)
 - iii) active at both x^0 and x^* (μ_3)
(note: $\mu_3 \leq \min(\mu_1, \mu_2)$)
 - iv) not active at x^0 nor x^* (μ_4).

In addition to the constraints specified by the user, one more constraint is generated. This last constraint is chosen to insure that x^* is indeed the optimal solution; that is, so that x^* and λ^* satisfy the Kuhn-Tucker necessary and sufficient conditions for optimality. Thus, the total number of constraints is

$$m = \mu_0 + \mu_1 + \mu_2 + \mu_4 + 1. \quad (2.3)$$

Further, so that the number of active constraints at x^0 (denoted by k_0) or at x^* (denoted by k_1) will not exceed n, the values specified for μ_0 , μ_1 , and μ_2 must satisfy

$$\begin{aligned} k_0 &= \mu_0 + \mu_1 \leq n \\ k_1 &= \mu_0 + \mu_2 + 1 \leq n. \end{aligned}$$

In summary then, the problems are generated to allow the user to control the size of the problem, the condition number of C, and details of the constraint structure. As a consequence, these test problems are particularly well-suited to evaluating an algorithm in terms of its numerical stability and its ability to handle constraint structures of various kinds.

3. THE OBJECTIVE FUNCTION

The objective function for the linear least squares problem is determined by the matrix C and the vector d in (2.1). Following Birkhoff and Gulati [1], we generate C as follows: $C = PDQ$, where P and Q are randomly generated ($p \times p$ and $n \times n$, respectively) orthogonal matrices and D is an $p \times n$ diagonal matrix ($D_{ij} = 0$, if $i \neq j$).

Birkhoff and Gulati show that the probability measure for the matrix so generated is invariant under orthogonal transformations. This can be interpreted to mean, for example, that the ellipsoids which represent the level surfaces of the objective function in the QP are randomly oriented.

To generate the requisite orthogonal matrices, we follow the procedure given by Stewart, in which, conceptually, we find the orthogonal matrix in the QR decomposition of some square matrix N , each of whose elements is chosen from the standard normal distribution. Computationally, however, the matrix N is not actually formed; as a consequence, only $\frac{1}{2} n^2$ standard normal variates are needed.

The condition number of C is determined by the diagonal elements of D , which are chosen as follows:

$$\begin{aligned} D_{11} &= 1/t' \\ D_{ii} &= (t')^{u_i}, \quad i = 2, \dots, n-1; \\ D_{nn} &= t'; \end{aligned} \tag{3.1}$$

where t' is the square root of the desired condition number t and each u_i is a uniform variate on the interval $(-1, +1)$. As a consequence, the singular values of C are distributed on $(1/t', t')$ according to a log uniform distribution.

Next, we must choose the n -vector d . This is equivalent to choosing the vector of residual errors in the least squares problem at the optimal solution x^* . Denoting the residuals by r^* , we set $d = Cx^* + r^*$, where $r^* = \beta z$, z is an n -vector whose components are standard normal variates, and β is a scalar. (Thus, each component of the residual error vector is normally distributed with mean zero and variance β^2 .) The magnitude of β is chosen so that the sum of squared residuals at the optimal solution will be equal to the user-specified fraction of the original objective function value; that is, so that

$$\beta^2 \|z\|^2 = \|r^*\|^2 = \gamma^2 \|Cx^0 - d\|^2. \quad (3.2)$$

Since $d = Cx^* + \beta z$, (3.2) is a quadratic equation in β , whose solutions are:

$$\beta = \eta \{v' Cz \pm [(v' Cz)^2 + \|Cv\|^2 / \eta]^{1/2}\}, \quad (3.3)$$

where $v = x^* - x^0$ and $\eta = \gamma^2 / [(1 - \gamma^2) \|z\|^2]$.

The sign of the square root in (3.3) will be determined in conjunction with the construction of the constraints active at x^* (see Section 4 for details). Also, in constructing those constraints, it will be necessary to apply a scale factor to the objective function, as described in Section 4. The link between the objective function and the constraints is necessitated by the Kuhn-Tucker optimality conditions.

We note in passing that the unconstrained least squares solution will be

$$\hat{x} = x^* + (C'C)^{-1} C'r^*. \quad (3.4)$$

Further, at $\hat{x}$, the vector of residual errors will be zero.

4. THE CONSTRAINTS

The specified m constraints are generated, one at a time, in the form

$$a_j' x \rho_j b_j, \quad j=1, \dots, m; \quad (4.1)$$

where a_j is an n -vector, ρ_j is a relational operator from the set $\{\leq, =, \geq\}$, b_j is a scalar.

Each vector a_j represents a constraint normal and is chosen from the uniform distribution over the unit sphere by setting $a_j = z / \|z\|$, where z is an n -vector whose elements are standard normal variates.

There are five different kinds of constraints specified by the user and these will be discussed in turn. Although our current implementation is for a

specific x^0 (starting point) and x^* (solution), the constraint generation will be described for a general x^0 and x^* . For convenience in what follows, we let $v = x^* - x^0$.

(1) Equality constraints

In this case, a_j must be transformed to a vector orthogonal to the vector v . First we let

$$\hat{a} = (I - vv' / \|v\|^2) a_j, \quad (4.2)$$

where I is an $n \times n$ identity matrix. Then, normalizing, we have $a_j = \hat{a} / \|\hat{a}\|$. We set $b_j = a_j' x^0$, and of course ρ_j is " $=$ ".

(2) Inequality constraints active at x^0 (not at x^*)

Set $b_j = a_j' x^0$, and then choose ρ_j so that x^* is feasible: if $a_j' x^* \geq b_j$, $\rho_j = ">"$; otherwise, $\rho_j = "<"$.

(3) Inequality constraints active at x^* (not at x^0)

Set $b_j = a_j' x^*$, and then choose ρ_j so that x^0 is feasible: if $a_j' x^0 \geq b_j$, $\rho_j = ">"$; otherwise, $\rho_j = "<"$.

(4) Inequality constraints active at both x^0 and x^*

This is similar to the case of equality constraints, except that $\rho_j \neq "="$. So, we choose, arbitrarily, $\rho_j = "<"$.

(5) Inequality constraints inactive at x^0 and x^*

We define these constraints in terms of their distance from either x^0 or x^* , selected randomly. The constraint is then defined to have a slack variable at the chosen point whose value is the absolute value of a standard normal variate. Choose u , a uniform variate on the interval $(0,1)$, and y , a standard normal variate. Then, if $u \leq 0.5$, set $b_j = a_j' x^0 + |y|$, and choose ρ_j so that x^* is feasible (see (2) above); Otherwise, set $b_j = a_j' x^* + |y|$, and choose ρ_j so that x^0 is feasible (see (3) above).

Finally we come to the construction of the last of the m constraints in the problem, which we will denote by $a_m' x \rho_m b_m$. The vector a_m must be chosen to satisfy the Kuhn-Tucker optimality conditions. Introducing a scale factor α^2 , so that the objective function is now

$$\alpha^2 \|Cx-d\|^2, \quad (4.3)$$

and letting

$$g = \sum_{j \in J} \lambda_j^* a_j \quad \text{and} \quad h = C'r^*, \quad (4.4)$$

the optimality conditions are

$$\lambda_m^* a_m = -g + \alpha^2 h, \quad (4.5)$$

where J is the index set of all constraints satisfied as equalities at x^* (those in categories (1), (3), and (4), above) not including this last one, and λ_j is the optimal value of the corresponding Lagrange multiplier. In particular, we adopt the convention $\lambda_j = -1$, if $\rho_j = ">"$; and $\lambda_j = 1$, otherwise. In order that x^0 will be feasible with respect to this last constraint, we require that a_m be oriented so that $\lambda_m^* v'a_m \geq 0$. We know that $v'g \geq 0$ (because x^0 is feasible for all the previously defined constraints). To make $v'h > 0$, we choose the sign of the square root in (3.3) to be $\text{sign}(v'Cz)$.

[Remark: First, however, we must be sure that $v'h \neq 0$. We check this when generating the objective function. If the randomly generated residual error vector z yields $v'h=0$ (an event with probability zero), a new z is generated.]

It remains to choose an appropriate value of α^2 . In the case where $v'g > 0$ and $k_1 > 1$, we can choose $\alpha^2 = k_1 v'g / (k_1 - 1) v'h$. As a consequence, $\lambda_m^* v'a_m = v'g / (k_1 - 1)$. Thus, the amount of slack in the m th constraint at x^0 is the average of the amount of slack at x^0 for the other $(k_1 - 1)$ constraints active at x^* . (Alternatively, the choice of α^2 may be used to make $\|a_m\| = 1$ (or nearly so) or to achieve some particular value of $\cos(v, a_m)$.)

The formula for α^2 must be modified if $k_1 = 1$ or $v'g = 0$, the latter being an event which occurs with certainty when $\mu_2 = \mu_3$ for example, or which

can occur by chance (again an event with probability zero). In this situation, we use the formula $\alpha^2 = 0.1 \|v\|/\|h\|$, which scales the gradient at x^* to have norm equal to $0.1 \|v\|$.

To complete the construction of this last constraint, set $\rho_m = "<"$, (implying $\lambda_m^* = 1$) and set $b_m = a_m' x^*$.

5. SAMPLE USE OF TEST PROBLEMS

To demonstrate the utility of the test problem generator, we used these problems to evaluate a software package which implements an algorithm for the linearly constrained linear least squares problem. The algorithm is that proposed by Stoer and Schittkowski [17] and implemented by Crane et al. [4]. The algorithm is characterized by the use of matrix decomposition and updating procedures which are designed to maintain a high degree of numerical stability.

Experiments were performed on an IBM 3033 under CMS. All programming was done in FORTRAN, compiled by the FORTRAN-H compiler. The data for the test problems were generated and all calculations were performed in double-precision arithmetic.

Uniformly distributed random numbers were generated by the multiplicative congruential method as implemented in the routine "RANF" from the Argonne AMD library [2]. Normally distributed random numbers were generated using the routine "GGNQF" from International Mathematical and Statistical Library (IMSL). The seeds for both random number generators were initialized only once before generating all the results given in Table 1. All "zero" tolerances in the test problem generator were set to 10^{-14} .

The test problems were specified to have varying: (1) condition numbers for the matrix of observations (ranging from 10^2 to 10^6); (2) numbers of inequality constraints active at the solution (ranging from 1 to n); and (3) numbers of inactive constraints (ranging from none to $4n$). In all cases, the starting point (the origin) was unconstrained and there were no equality constraints. The number of variables (n) was 10; the number of observations (p) was 20; and γ^2 was 0.1.

Variations in the first parameter (condition number) are intended to test numerical stability; variations in the second parameter (active constraints at the solution) are intended to test the degree to which constraints improve or degrade performance; and variations in the last parameter (inactive constraints) are intended to test the algorithm's ability to find the correct active set.

The measures of performance recorded were:

- (1) number of iterations (that is, number of changes in the active set);
- (2) accuracy of the solution.

For the latter, we computed

$$A = \frac{1}{n} \sum_{i=1}^n L(|\hat{x}_i - x_i^*|) / n, \quad (5.1)$$

where

$$L(x) = \begin{cases} 16 & \text{if } x = 0 \\ -\log_{10}(x) & \text{otherwise} \end{cases}.$$

$\hat{x}_i$ is the computed optimum, and x_i^* is the correct optimal solution. The quantity A may be interpreted as the average number of significant digits of accuracy in the components of the solution vector.

The experiment consisted of generating and solving ten problems of each type. The mean and standard deviation of each measure of performance was calculated and reported. The results of the experiment are reported in Table 1.

The results show the loss of accuracy as the condition number increases. Although 15 digits of accuracy are possible, only 6 digits of accuracy are obtained when the condition number is 10^6 . However, as predicted by Stewart, the presence of active constraints clearly improves accuracy, because of the high probability that the problem will be well-conditioned when projected into the subspace defined by the active constraints. In the presence of relatively larger numbers of inactive constraints, the number of iterations increases. We also observe an increase in the number of iterations as the condition number increases from 10^2 to 10^4 although no such increase is apparent as the condition number increases from 10^4 to 10^6 .

6. CONCLUSIONS

We have described a procedure for generating positive definite quadratic programming test problems which allows the user to control the condition number of the Hessian matrix and to specify the constraint structure. A distinctive feature of the procedure is that the distributions of the constraint normals and the principal axes of the ellipsoids representing the level surfaces of the objective function are randomly oriented.

By using the test problems so generated to evaluate several aspects of the performance of a particular software package, we have shown how this type of testing can be a valuable approach to obtaining information about the strengths and weaknesses of such a package.

A number of extensions of this work can be suggested immediately. These include: generating one or more infeasible starting points, scaling the solution vector, introducing sparsity, controlling the condition number of the constraint matrix, generating constraints in conjunction with the Hessian, and generating indefinite and semi-definite problems.

More generally, of course, there should be some attempt to identify the significant aspects of "real-world" QP problems, so that they can be represented fairly in the generation of random test problems. Further, the generation of QP problems may be helpful in understanding how to generate more general nonlinear optimization problems. Finally, attention needs to be given to the design of experiments in software evaluation so that the magnitude of testing effort can be controlled.

TABLE 1

Experimental Results ($n=10$, $p=20$, $\gamma^2=0.1$)

Condition Number	No. of Active Constraints		Number of Inactive Constraints		
			None	2n	4n
			MEAN (STD DEV)	MEAN (STD DEV)	MEAN (STD DEV)
10^2	1	Iter.	1.0 (0.0)	15.4 (5.80)	16.6 (10.95)
		Accur.	13.92 (0.55)	13.81 (0.48)	13.74 (0.53)
	n/2	Iter.	6.0 (0.0)	16.8 (4.13)	27.8 (7.51)
		Accur.	15.28 (0.33)	15.20 (0.39)	15.01 (0.30)
	n	Iter.	10.0 (0.0)	23.4 (5.66)	28.0 (8.84)
		Accur.	15.24 (0.51)	15.13 (0.35)	15.34 (0.31)
10^4	1	Iter.	1.0 (0.0)	25.4 (6.10)	35.8 (8.65)
		Accur.	9.99 (0.54)	9.84 (0.53)	10.07 (0.56)
	n/2	Iter.	6.0 (0.0)	28.0 (4.81)	37.8 (7.15)
		Accur.	13.98 (1.4)	13.69 (1.4)	13.83 (1.1)
	n	Iter.	10.0 (0.0)	30.0 (6.67)	40.4 (9.32)
		Accur.	15.10 (0.48)	15.13 (0.47)	15.03 (0.34)
10^6	1	Iter.	1.0 (0.0)	29.6 (5.82)	39.6 (7.89)
		Accur.	6.20 (0.56)	6.09 (0.55)	5.96 (0.67)
	n/2	Iter.	6.0 (0.0)	28.8 (5.90)	35.0 (4.92)
		Accur.	12.61 (2.7)	12.16 (1.8)	11.46 (2.1)
	n	Iter.	10.0 (0.0)	29.4 (6.80)	39.4 (7.83)
		Accur.	14.81 (0.74)	15.06 (0.49)	15.12 (0.67)

REFERENCES

- [1] Birkhoff, G. and Gulati, S., "Isotropic distributions of test matrices," *Journal of Applied Mathematics and Physics (ZAMP)* 30 (1979), 148-158.
- [2] Clark, N. W., "Applied Mathematics Division System/360 library subroutine RANF: ANL G552S," Argonne National Laboratory, Argonne, Ill., 1967.
- [3] Colville, A. R., "A comparative study of nonlinear programming codes," in: H. W. Kuhn (Ed.), *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton University Press, Princeton, N.J., (1970), 487-501.
- [4] Crane, R. L., Garbow, B. S., Hillstrom, K. E., and Minkoff, M., "LCLSQ: An implementation of an algorithm for linearly constrained linear least-squares problems," Mathematics and Computers Report ANL-80-116, Argonne National Laboratory, Argonne, Ill., November, 1980.
- [5] Crowder, H. P., Dembo, R. S., and Mulvey, J. M., "Reporting computational experiments in mathematical programming," *Math. Programming* 15 (1978), 316-329.
- [6] Fletcher, R., "A general quadratic programming algorithm," *Journal of the Institute of Mathematics and Its Applications* 7 (1971), 76-91.
- [7] Hanson, R. J. and Haskell, K. H., "Algorithm 5XX. Two algorithms for the linearly constrained least squares problem," *ACM Transactions on Math. Software*, to appear (1982).
- [8] Hock, W. and Schittkowski, K., *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer-Verlag, Berlin, 1981.
- [9] Lasdon, L. S., Waren, A. D., Jain, A., and Ratner, M., "Design and testing of a generalized reduced gradient code for nonlinear programming," *ACM Transactions on Math. Software* 4 (1978), 34-50.
- [10] Lawson, C. L. and Hanson, R. J., *Solving Least Squares Problems*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1974.
- [11] Lenard, M. L., "A computational study of active set strategies in nonlinear programming with linear constraints," *Math. Programming* 16 (1979), 81-87.

- [12] Lyness, J. N., "A benchmark experiment for minimization algorithms," *Math. of Comp.* 33 (1979), 249-264.
- [13] Michaels, W. M. and O'Neill, R. P., "A mathematical programming generator MPGENR," *ACM Transactions on Math. Software* 6 (1980), 31-44.
- [14] Minkoff, M. "Methods for evaluating nonlinear programming software," in: O. L. Mangasarian, R. R. Meyer and S. M. Robinson (Eds.), *Nonlinear Programming* 4, Academic Press, New York, (1981), 519-548.
- [15] Rosen, J. B. and Suzuki, S., "Construction of nonlinear programming test problems," *Comm. ACM* 8 (1965), 113.
- [16] Sandgren, E. and Ragsdell, K. M., "On some experiments which delimit the utility of nonlinear programming methods for engineering design," in: A. G. Buckley and J.-L. Goffin (Eds.), *Mathematical Programming Study* 16, North-Holland, Amsterdam (1982), 118-136.
- [17] Schittkowski, K. and Stoer, J., "A factorization method for constrained least squares problems with data changes: Part I: Theory," *Numer. Math.*, 31 (1979), 431-463.
- [18] Stewart, G. W., "The efficient generation of random orthogonal matrices with an application to condition estimators," *SIAM Jour. on Numerical Analysis* 17 (1980), 403-409.
- [19] Stewart, G. W., "Constrained definite Hessians tend to be well-conditioned," *Math. Prog.* 21 (1981), 235-238.
- [20] van Dam, W. B. and Telgen, J., "Randomly generated polytopes for testing mathematical programming algorithms," Report 7929/0, Erasmus Univ., Rotterdam, 1979.

ACKNOWLEDGEMENTS

One of us (MLL) gratefully acknowledges the assistance provided by the Mathematics and Computer Science Division of Argonne National Laboratory.