[7] Dowell, E. H. and Ventres, C. S., "Derivation of Aerodynamic Kernel Functions," *AIAA Journal*, Vol. 11, Nov. 1973, pp. 1586-1588.

[8] Landahl, M. T. and Stark, V. J. E., "Numerical Lifting Surface Theory—Problems and Progress," *AIAA Journal*, Vol. 6, Nov. 1968, pp. 2049-2060.

Robust Code for Constrained Optimization

Donald A. Pierre*
Montana State University, Bozeman, Mon.

Introduction

EASON and Fenton[1] and Pappas[3] have evaluated the performance of eighteen optimization codes for solving nonlinear programming problems. All codes were applied to ten problems having inequality constraints, and the results were used to rank the codes in terms of generality and efficiency. Of the eighteen codes, five used information concerning derivatives or derivative approximations; the best of these five codes solved only half of the problems under a specified set of test conditions. As a result, the gradient-based codes ranked low in terms of performance.

Among the recent advances in gradient-based methods have been the development of a variety of augmented Lagrangian methods (also called multiplier methods) which were originally introduced[4,5] for equality constrained problems, but have since been generalized in various ways[7] to account for inequality constraints as well. One of these generalizations[6] led to the development of an algorithm and a FORTRAN code LPNLP[7] which has proved to be very efficient in solving a variety of test problems and application problems; a method using gradient approximations with LPNLP has been evaluated in Ref. 8. In contrast to conventional gradient-based penalty function methods, the augmented Lagrangian methods do not require in general that penalty weights approach infinity as the solution progresses.

The purpose of this Note is to present the results of applying LPNLP to the ten problems cited in Ref. 1. The results show that LPNLP compares favorably with the best of the codes tested in Refs. 1 and 3.

Of interest are problems of the form

$$\text{maximize } f(x) \tag{1}$$

subject to

$$p_i(x) = a_i, \quad i = 1,2,\ldots,NE < n \tag{2}$$

$$q_j(x) \leq b_j, \quad j = 1,2,\ldots,NI \tag{3}$$

and

$$c_k \leq x_k \leq d_k, \quad k = 1,2,\ldots,n \tag{4}$$

where x is an n vector $(x_1, x_2, \ldots, x_n)$; the a_i's, b_j's, c_k's, and d_k's are real constants and are supplied in the data set; and f, p_i's, and q_j's are functions that are assigned in subroutine FXNS. In the analytical gradient (AG) case, the first derivatives of these functions are assigned in subroutine GRAD; but in the discrete gradient (DG) case, subroutine GRAD is standardized[8] and need not be modified from one problem to the next. Because the ten problems in Ref. 1 are minimization problems, LPNLP makes use of the relationship $\min f = -\max(-f)$.

Received Jan. 17, 1977; revision received March 17, 1977.

Index categories: Analytical and Numerical Methods; Performance; Structural Design.

*Professor, Electrical Engineering Department.

Test Conditions and Results

The LPNLP code has two major phases: an unconstrained maximization phase and a multiplier/weight update phase. Four distinct modes of unconstrained maximization are available in LPNLP. Only one of these modes, the Davidon-Fletcher-Power mode with controlled reset (DFP/RESET mode), was used here. From previous experience[7,8] this mode has been the most consistent, but usually not the fastest. The reset rate parameter NSRCH was assigned the value $2n+1$ where n is the number of search variables.

Other control parameters in the code pertain to convergence tests and penalty weights. For all problem solutions, solution parameters ϵ_1, ϵ_2, and ϵ_3 were assigned values of 0.0001, 0.01, and 0.01, respectively. These settings resulted in excess of six place accuracy in the maximum value of f at convergence. In all but one case, penalty weights were assigned initial values of unity, were updated by a factor of four, and were held to a maximum of sixteen. All Lagrange multipliers were initialized at zero. For the discrete gradient approximation, a delta factor[8] of 10^{-8} was applied. The LPNLP code in Refs. 7 and 8 was used as given, with all print statements retained, but with output control flags set so that only initial conditions and terminal conditions were printed. Results were obtained on a time-shared XDX Sigma-7 computer with an automatic double-precision option (equivalent to 15 significant decimal digits of accuracy).

Table 1 lists the analytical gradient (AG) and discrete gradient (DG) results in terms of number of function evaluations. For problems 9 and 10, the problem statements were taken directly from FORTAN code in Ref. 2 (except that the objective functions were the negative of those in Ref. 2), and analytical gradients were not readily available. The NG column in Table 1 gives the number of calls to GRAD, NF gives the number of calls to FXNS, and the number of "equivalent" function evaluations NEF is calculated, NEF = n NG + NF. Note that 5 of 8 NEF/AG values are identical to corresponding NF/DG values, which suggests that solution paths under the two solution modes are very similar.

A summary of timing results is given in Table 2. The normalized time values in Table 2 were calculated by dividing the actual CPU times, obtained by loading and running executable load modules to convergence, by the corresponding CPU time for Colville's timing program.[9] Three of these times, for problems 1 (with 5 variables), 3 (5 variables), and 10 (4 variables), are smaller than corresponding times obtained by any of the eighteen codes tested in Refs. 1 and 3. Problem 4 is the only other problem with 4 variables; the remaining problems have either two or three variables.

In comparing AG timing values to corresponding DG values, the AG values have a slight advantage. This is more than offset, however, by the coding advantage that results from using a standard code for subroutine GRAD in the discrete gradient case. The timing results for problems 9 and 10 could be improved significantly if the numerous trigonometric terms used in FXNS were evaluated in the calling program and made available to FXNS by a labeled common block. For consistency with earlier results, however, this was not implemented.

Timing results merit closer examination. In a time-shared computer environment, the same executable load module can be run at two different times and require different CPU times; a difference of 5% in CPU times is not unusual. Furthermore, the compiling and linking times are not included in timing results. To illustrate the effect of these operations, problem 3 was run in several ways. First, the total normalized time required to compile a short calling program, to link all relevant compiled subroutines, and to run the program, was 0.1724 units. Next, the normalized time required to compile a short calling program, to link the same subroutines, and to simply exit from the calling program (prior to calling on LPNLP to solve the problem), was 0.1457 units – the difference between this value and the preceding one is 0.0267

Table 1 Function and gradient calls required for convergence

Problem number	NG	Analytical gradient NF	Analytical gradient NEF	Discrete gradient NF
1	39	122	317	317
2	32	105	201	201
3	44	133	353	387
4	70	209	489	506
5	35	105	175	177
6[a]	26	92	144	144
7	18	55	109	109
8	10	24	44	44
9[b]		(not run)		335
10		(not run)		111

[a]Reciprocals of x and x_2 occur in problem 6, and the maximum of the objective function can be made arbitrarily large by assigning $x_1 = \epsilon$ and $x_2 = -\epsilon^2$ with $0 < \epsilon << 1$. To avoid this, weight w_2 was held at 100 and w_3 at 1000, and the lower bounds on both x_1 and x_2 were increased from 0 to 0.1.

[b]The solution to problem 9 terminated before gradient convergence criteria were satisfied, but accuracy in the final value of f was within 6 significant figures of the optimum.

Table 2 Normalized CPU times for convergence

Problem number	Analytical gradient	Discrete gradient
1	0.0410	0.0471
2	0.0243	0.0248
3	0.0335	0.0370
4	0.0304	0.0339
5	0.0201	0.0211
6	0.0203	0.0222
7	0.0220	0.0217
8	0.0166	0.0168
9	(not run)	0.6921
10	(not run)	0.2532

units. And finally, when an executable load module for the problem was prepared so that execution in the short calling program terminated without calling LPNLP to solve the problem, the loading and execution of this required 0.0076 units. If a given problem is of large dimension or is to be run with many different data sets (different constraint bounds perhaps) the normalized times for each execution, such as given in Table 2, are important. However, if a low dimensioned problem is to be solved but once, the time required to link the optimization code is often more significant.

The performance of LPNLP is dependent on scaling of variables and constraint functions. Except for problem 7, however, the scale factors given in Refs. 1 and 2 were used. For problem 7, the constraints as given[1] are

$$x_1^2 x_2 \leq 675.0 \tag{5}$$

and

$$10^{-7}(x_1 x_3)^3 \leq 0.419 \tag{6}$$

A common weighting factor is used for both of these constraints in LPNLP. In order to have these constraints weighted more evenly, they were scaled as follows:

$$0.1 x_1^2 x_2 \leq 67.5 \tag{7}$$

and

$$10^{-5}(x_1 x_3)^3 \leq 41.9 \tag{8}$$

with which results were obtained as given in Tables 1 and 2.

Conclusion

LPNLP is a robust optimization code; in addition to the low dimensioned problems of this work, it has been applied with success[7,8] to problems having as many as 20 variables and numerous constraints, and to a 100 variable generalization of problem 5. The development of augmented Lagrangian methods should do much to offset previously reported dissatisfaction[1,3] with gradient-based methods. Also, the Lagrange multipliers, which are obtained as an integral part of each solution, are a valuable adjunct to solution sensitivity analysis.[7] The documentation and FORTRAN program listings for LPNLP are readily available. Information on how to obtain card source can be obtained from the author.

References

[1]Eason, E.D. and Fenton, R.G., "A Comparison of Numerical Optimization Methods for Engineering Design," *Transactions of the ASME, Ser. B: Journal of Engineering for Industry*, Vol. 96, Feb. 1974, pp. 196-200.

[2]Eason, E.D. and Fenton, R.G., "Testing and Evaluation of Numerical Methods for Design Optimization," Rept. UTME-TP 7204, Sept. 1972, Dept. of Mechanical Engineering, Univ. of Toronto, Toronto, Canada, Sept. 1972.

[3]Pappas, M., "Performance of the 'Direct Search Design Algorithm' as a Numerical Optimization Method," *AIAA Journal*, Vol. 13, June 1975, pp. 827-829.

[4]Hestenes, M.R., "Multiplier and Gradient Methods," *Journal of Optimization Theory and Applications*, Vol. 4, Nov. 1969, pp. 303-320.

[5]Powell, M.J.D., "A Method for Nonlinear Constraints in Minimization Problems," in *Optimization*, R. Fletcher (editor), Academic Press, New York, 1969.

[6]Pierre, D.A., "Multiplier Algorithms for Nonlinear Programming," *Presented at the 8th International Symposium on Mathematical Programming*, Aug. 1973.

[7]Pierre, D.A. and Lowe, M.J., *Mathematical Programming Via Augmented Lagrangians: An Introduction with Computer Programs*, Addison-Wesley, Reading, Mass. 1975.

[8]Pierre, D.A. and Dudding, C.H., "Constrained Optimization with Gradient Approximations," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-7, Feb. 1977.

[9]Colville, A.R., "A Comparative Study on Nonlinear Programming Codes," IBM New York Scientific Center Report No. 320-2949, June 1968.

Static Pressure in Hypersonic Nozzle Boundary Layers

P. J. Finley*
Imperial College of Science and Technology, London, England

LARGE variations of static pressure may be observed in hypersonic boundary-layer flows (see Kemp and Owen,[1] Fig. 9, for a collection of observations), and are shown below to result from the combined effects of longitudinal curvature of the mean streamlines and the increasing importance, as the Mach number rises, of the Reynolds stress contribution to the total normal stress perpendicular to the wall.

The momentum equation normal to a streamline may be written

$$(\partial\sigma/\partial n) = (\partial/\partial n)(\bar{p} + \bar{\rho}\overline{v'^2}) = -\rho U^2/R_x \tag{1}$$

where σ is the total normal stress made up of contributions from the mean static pressure $\bar{p}$ and the mean normal Reynolds stress $\bar{\rho}\,\overline{v'^2}$, and R_x, the longitudinal radius of curvature of the streamline, is defined positive for concave walls. In hypersonic flows the specific momentum flux ρU^2 is potentially very large compared with the mean static pressure $\bar{p}$, and if therefore we integrate Eq. (1) across a boundary

Received Jan. 17, 1977; revision received March 8, 1977.
Index categories: Boundary Layers and Convective Heat Transfer—Turbulent; Supersonic and Hypersonic Flow.

*Lecturer, Department of Aeronautics.