Stanford University, ICME

CME 338 Large-Scale Numerical Optimization

Instructor: Michael Saunders Spring 2019

Notes 10: **MINOS Part 1: the Reduced-Gradient Method**

1 Origins

The first version of MINOS (Murtagh and Saunders [25]) was designed to solve *linearly constrained* optimization problems of the form

LC1	$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \phi(x)$
	$\text{subject to} \quad \ell \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u,$

where $\phi(x)$ is a smooth function (ideally involving only some of the variables), and A is a sparse $m \times n$ matrix as in a typical LO problem. The gradients of $\phi(x)$ were assumed to be available, but no use could be made of second derivatives.

Earlier algorithms for dense LC problems had been proposed by several authors, including the gradient-projection method of Rosen [27], the reduced-gradient method of Wolfe [30, 31], the variable-reduction method of McCormick [22], and the active-set methods of Gill and Murray [12, 13]. The focus was on constraints $Ax \ge b$, $m > n$ with projections onto the set of active constraints requiring factorizations of matrices with changing *row dimensions*. The only large-scale implementation was that of Buckley [5].

The particular reduced-gradient/variable-reduction algorithm developed in MINOS was a natural way to combine the established *sparse-matrix technology of implementations of the simplex method* with the fairly recent quasi-Newton/variable-metric approach to *dense unconstrained optimization* [7]. In contrast to Buckley's $Ax \ge b$ focus, MINOS follows typical large-scale simplex implementations in working with the equality form, where A and x here include a full set of slack variables:

LC	$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \phi(x)$
	$\text{subject to} \quad Ax = b, \quad \ell \le x \le u.$

We use the term *reduced-gradient* (RG) in the context of LC optimization, even though *variable-reduction* conveys the correct idea. Similarly, we use *quasi-Newton* (QN) rather than *variable-metric* for unconstrained optimization. This avoids using "variable" as a noun in one context and an adjective in the other, and quasi-Newton conveys the correct idea of approximating second derivatives.

Although explicit *sparse QN methods* have been developed for unconstrained optimization (Toint [28]), they are not likely to be useful for the reduced-gradient method because the underlying *exact reduced Hessian* is usually not sparse. For problems with very large reduced Hessians, we may consider Conjugate-Gradient methods or *limited-memory QN methods* [6, 33], but factorization of the full KKT system is likely to be more efficient, as in the QP solver SQIC [15] or interior solvers like IPOPT and KNITRO [16, 17].

2 Concave objectives

If the objective function $\phi(x)$ happens to be *concave*, an active-set algorithm could proceed exactly like the simplex method. Whenever a nonbasic variable is chosen to be moved from its current value, it would want to continue moving as far as possible until some variable reached a bound. Thus, all solutions of problem LC lie at a vertex of the feasible region, the same as for LO problems. However, there are likely to be many *local minima*, most of which will be of little interest.

The classic example is continually rediscovered by modelers trying to impose integer constraints. If the first N variables are supposed to be either 0 or 1, it is tempting to consider the objective function

$$\phi(x) = \sum_{j=1}^{N} x_j(1 - x_j)$$

with bounds $0 \le x_j \le 1$. However, MINOS and other nonlinear solvers are most likely to find a *local* optimum with non-integer values for many of the x_j. A more sophisticated solver is needed such as BARON [3], which proceeds by solving a sequence of continuous subproblems that *can* be handled by solvers such as MINOS or SNOPT.

As an alternative to BARON, the *global smoothing* approach of Murray and Ng [23, 26] has achieved significant success. It proceeds by solving a sequence of subproblems of the form

LC(γ, μ)	$\underset{x}{\text{minimize}}$	$\phi(x) + \gamma \sum x_j(1 - x_j) - \mu \sum \ln(x_j(1 - x_j))$
	subject to	$Ax = b, \quad 0 \le x \le 1,$

in which γ, $\mu > 0$ and μ is initially *large* in order to make the combined objective convex. As μ is reduced, the concave γ term becomes increasingly effective. Thus, *concave objective functions* and *local LC solvers* have a promising new use. (Note that a specialized LC solver is probably needed. In order to deal effectively with saddle points it would ideally use second derivatives.)

Further success has been achieved at SOL with a local LC solver on problems with many discrete and continuous variables, using local improvement to solutions from a series of LC subproblems (Murray and Shanbhag [24]).

3 Superbasic variables

With nonlinear objective functions, we generally do not expect an optimal point to be a basic solution. For example, the problem

$$\min \ \phi(x) = x_1^2 + x_2^2 \quad \text{subject to} \quad x_1 + x_2 = 2, \quad 0 \le x \le 3,$$

has a unique optimum at $(x_1, x_2) = (1, 1)$ with $\phi(x) = 2$, whereas both possible basic solutions give $\phi(x) = 4$. If we started a simplex-type method at the basic solution $(2, 0)$ and began to increase the (only) nonbasic variable x_2, the objective function would start to decrease as desired. However, it would reach a minimum *before* x_1 reached 0 or x_2 reached its upper bound of 3. One of the innovations in MINOS was to extend the concept of basic and nonbasic variables by introducing a set of *superbasic variables* to include variables like x_2 that move away from their bound but end up in "mid-air" without causing some other variable to reach a bound. The constraints $Ax = b$ are partitioned as follows, with B nonsingular as for simplex methods:

$$Ax = \begin{array}{|c|c|c|} \hline B & S & N \\ \hline \end{array} \begin{pmatrix} x_B \\ x_S \\ x_N \end{pmatrix} = b.$$

$$\begin{matrix} m & s & n - m - s \end{matrix}$$

The active set may be written as $\begin{pmatrix} B & S & N \\ & & I \end{pmatrix} \begin{pmatrix} x_B \\ x_S \\ x_N \end{pmatrix} = \begin{pmatrix} b \\ x_N \end{pmatrix}$, with $\ell_N \le x_N \le u_N$.

3.1 The size of S

The number of superbasic variables is a measure of the *nonlinearity* of the problem in several ways, and it could be called the *number of degrees of freedom*:

- If the objective function is linear (so that problem LC is a linear program), the reduced-gradient method in MINOS unknowingly mimics the simplex method. The value of s oscillates between 0 and 1.
- On nonlinear programs with only n_{NL} variables appearing nonlinearly in the objective, s need not be larger than $n_{NL} + 1$.
- Even if all variables appear nonlinearly, the objective may be "nearly linear" and s could remain small. For example, $\phi(x) = c^T x + \psi(x)$ might have c significantly larger than $\nabla\psi(x)$ at all feasible points x, or $\nabla\psi(x)$ might be nearly constant.

There is no vital difference between basic and superbasic variables. All we need is a non-singular B, ideally not too ill-conditioned. MINOS performs two kinds of "basis repair" at times in order to preserve this situation (see [14, section 5]):

BR factorization checks the rank of B using LUSOL's TRP (threshold rook pivoting) option and may replace some of its columns by unit vectors belonging to slacks in S or N.

BS factorization may shuffle the columns of $(B\ S)$, using LU factors of $(B\ S)^T$ (with L well-conditioned) to choose a better-conditioned B.

3.2 Optimality conditions

A triple (x, y, z) satisfies the *first-order KKT conditions* for problem LC if

$$\begin{aligned} Ax &= b, \\ \ell \le x &\le u, \\ z &= g(x) - A^T y, \\ \min(x - \ell,\quad z) &= 0, \\ \min(u - x, -z) &= 0, \end{aligned}$$

where $g(x) = \nabla\phi(x)$ is the gradient of the objective, and z is a vector of *reduced gradients*. We assume that $Ax = b$, $B^T y = g_B$ and $z = g - A^T y$ can be satisfied with high accuracy throughout (so that $z_B = 0$ by construction). A point (x, y, z) is regarded as feasible and optimal to within tolerances δ_P, δ_D (typically 10^{-6}) if

$$\begin{aligned} \|\min(x - \ell, u - x)\|_\infty &\ge -\delta_P, \\ \|\min(x - \ell\,,\quad z)\|_\infty &\le \delta_D, \\ \|\min(u - x, -z)\|_\infty &\le \delta_D. \end{aligned}$$

3.3 Suboptimization

As in the simplex method, the nonbasic variables are temporarily frozen at their current value. The variables in B and S are then optimized (to some extent) before another nonbasic variable is chosen to be part of S. We may think of this as suboptimization on the problem

BSprob	$\underset{x_B,\ x_S}{\text{minimize}}$	$\phi(x)$
	subject to	$\begin{pmatrix} B & S \end{pmatrix} \begin{pmatrix} x_B \\ x_S \end{pmatrix} = b - N x_N,$
		$\ell \le x \le u, \qquad x_N = x_N.$

Recall that Phase 1 simplex performs just *one* iteration on the current modified problem before defining the next modified problem. Similarly here—we do just one iteration before checking if (B, S, N) should be redefined:

- If a superbasic variable reaches a bound, it is moved from S to N.
- If a basic variable reaches a bound, a judicious column of S is chosen to enter B, and the basic variable is moved into N.
- If BSprob is sufficiently optimized, a nonbasic variable x_s is chosen to move from N into S, as in a simplex pricing operation.

The *active-set strategy* in MINOS is designed to avoid zig-zagging (bouncing on and off the same set of constraints) while limiting the effort devoted to any particular BSprob. Note that BSprob is optimal if the largest superbasic reduced gradient satisfies $\|z_S\|_\infty \leq \delta_D$. A dynamic tolerance δ_S is therefore defined and used as follows:

- Whenever a nonbasic variable x_s is moved from N into S to define a new BSprob, the dynamic tolerance is defined to be $\delta_S = 0.2|z_s|$, say (where 0.2 is the default `Subspace tolerance`—it may be specified at run-time).
- If $|z_s|$ is not sufficiently large ($|z_s| \leq 1.1\|z_S\|_\infty$, say) the chosen nonbasic variable is *not* moved into S. The dynamic tolerance is reduced ($\delta_S \leftarrow 0.9\|z_S\|_\infty$) and BSprob remains the same as before.
- The current BSprob is optimized until $\|z_S\|_\infty \leq \max\{\delta_S, \delta_D\}$. Only then is z_N evaluated to suggest a new variable for S.

3.4 Unconstrained optimization

A single iteration on problem BSprob requires a search direction (p_B, p_S, p_N) satisfying

$$Bp_B + Sp_S = 0, \qquad p_N = 0.$$

This may be thought of as finding a search direction for an *unconstrained* problem involving the superbasic variables x_S. It is equivalent to defining

$$p = Zp_S, \quad \text{where} \quad Z = \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}.$$

For this "reduced-gradient operator" Z, we have

$$\phi(x + Zp_S) = \phi(x) + g^T Zp_S + \tfrac{1}{2} p_S^T Z^T H Z p_S + \cdots,$$

where g and H are the current values of $\nabla\phi(x)$ and $\nabla^2\phi(x)$. Thus, the objective function behaves like an unconstrained function of the superbasic variables with gradient $Z^T g(x)$ and Hessian $Z^T H(x) Z$. In MINOS we use a nonsingular upper-triangular matrix R to maintain a quasi-Newton approximation

$$R^T R \approx Z^T H(x) Z.$$

For each subproblem BSprob, the primary steps to generate a search direction are as follows:

- Solve $R^T R p_S = -Z^T g$ (where $B^T y = g_B$ and $Z^T g = g_S - S^T y$).
- Define $p = Zp_S$ (i.e., solve $Bp_B = -Sp_S$).
- Find the maximum feasible steplength $\alpha_{\max}$ such that $\ell \leq x + \alpha_{\max} p \leq u$.
- Perform a linesearch to find a step α that approximately minimizes the objective $\phi(x + \alpha p)$ along the direction p within the interval $0 < \alpha \leq \alpha_{\max}$.
- Perform a quasi-Newton update on R to account for the "unconstrained" optimization step, using α, p_S, and the change in reduced gradient $Z^T g$.
- If the linesearch encountered a bound ($\alpha = \alpha_{\max}$), redefine BSprob and perform one or two "static" updates on R.

Note that operations with Z and Z^T involve solves with B and B^T as in the simplex method.

4 Utility

MINOS has been an effective large-scale solver (alongside CONOPT [8, 2]) since the earliest days of GAMS [4, 11] and AMPL [9, 10, 1], especially for problems with linear constraints and cheap functions and gradients, and for cases where the number of superbasics variables remains below 2000 (say). Warm starts are straightforward.

Many other solvers are now accessible from GAMS and AMPL (and TOMLAB [29]), including nonlinear interior methods such as IPOPT [16] and KNITRO [17], which can make use of second derivatives but usually can't be warm-started (although that folklore changed recently [18, 19, 20]).

MINOS 5.6 exists as two separate Fortran 77 programs (double and quad precision) for both linear and nonlinear optimization. In double MINOS, floating-point variables are declared `real(8)`. Quad MINOS is possible because certain compilers such as GNU gfortran allow those variables to be declared `real(16)` (increasing their precision from about 15 to 34 digits). For quad precision, the compiler generates calls to a software library and is therefore considerably slower, but the warm-start features of MINOS allow the double version to provide a reasonable starting point for quad MINOS.

Quad MINOS is now available in GAMS for linear optimization problems; see DQQ in the GAMS library [21]. (A major effort would be needed to make a quad version of the nonlinear functions in GAMS.) The use of quad MINOS for nonlinear models of metabolism and macromolecular expression (ME) is described in [32].

For problems where gradients can be coded, it is normal for quad MINOS to obtain solutions with 20 to 30 correct digits. If only function values are available, the finite-difference approximations to gradients are good to about 15 digits, and it is normal for quad MINOS to obtain solutions of that accuracy (equivalent to full accuracy in double precision).

References

[1] AMPL modeling system. `http://www.ampl.com`.

[2] ARKI Consulting & Development A/S. `http://www.conopt.com`.

[3] BARON global optimization system. `http://archimedes.scs.uiuc.edu/baron/baron.html`.

[4] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, Redwood City, California, 1988.

[5] A. Buckley. An alternative implementation of Goldfarb's minimization algorithm. *Math. Program.*, 8:207–231, 1975.

[6] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited-memory methods. *Math. Program.*, 63:129–156, 1994.

[7] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983. Reprinted as Classics in Applied Mathematics 16, SIAM, Philadelphia, 1996.

[8] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Math. Program.*, 31:153–191, 1985.

[9] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press, South San Francisco, 1993.

[10] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press / Brooks/Cole Publishing Company, second edition, 2002.

[11] GAMS modeling system. `http://www.gams.com`.

[12] P. E. Gill and W. Murray. Newton-type methods for linearly constrained optimization. In P. E. Gill and W. Murray, editors, *Numerical Methods for Constrained Optimization*, pages 29–66. Academic Press, London, 1974.

[13] P. E. Gill and W. Murray. Quasi-Newton methods for linearly constrained optimization. In P. E. Gill and W. Murray, editors, *Numerical Methods for Constrained Optimization*, pages 67–92. Academic Press, London, 1974.

[14] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. SIGEST article.

[15] P. E. Gill and E. Wong. Methods for convex and general quadratic programming. *Math. Prog. Comp.*, 7:71–112, 2015.

[16] IPOPT open source NLP solver. `https://projects.coin-or.org/Ipopt`.

[17] KNITRO optimization software. `https://www.artelys.com/tools/knitro`.

[18] D. Ma, K. L. Judd, D. Orban, and M. A. Saunders. Algorithm NCL for constrained optimization. `http://stanford.edu/group/SOL/talks.html`, 2018.

[19] D. Ma, K. L. Judd, D. Orban, and M. A. Saunders. Stabilized optimization via an NCL algorithm. `http://stanford.edu/group/SOL/multiscale/models/NCL/`, 2018.

[20] D. Ma, K. L. Judd, D. Orban, and M. A. Saunders. Stabilized optimization via an NCL algorithm. In M. Al-Baali, Lucio Grandinetti, and Anton Purnama, editors, *Numerical Analysis and Optimization, NAO-IV, Muscat, Oman, January 2017*, volume 235 of *Springer Proceedings in Mathematics and Statistics*, pages 173–191. Springer International Publishing Switzerland, 2018.

[21] D. Ma, M. A. Saunders, and S. Dirkse. dqq.gms: Warm-starting quad-precision MINOS in GAMS, 2017. `https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_dqq.html`.

[22] G. P. McCormick. The variable-reduction method for nonlinear programming. *Management Science*, 17(3):146–160, 1970.

[23] W. Murray and K.-M. Ng. Algorithms for global and discrete problems based on methods for local optimization. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization, Volume 2*, pages 87–113. Kluwer, Dordrecht, 2002.

[24] W. Murray and V. V. Shanbhag. A local relaxation approach for the siting of electrical substations. *Comput. Optim. Appl.*, 33:7–49, 2006. COAP 2006 Best Paper Award.

[25] B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Math. Program.*, 14:41–72, 1978.

[26] K.-M. Ng. *A Homotopy Algorithm for Nonlinear Discrete Problems*. PhD thesis, Dept of Management Science and Engineering, Stanford University, Jun 2002.

[27] J. B. Rosen. The gradient projection method for nonlinear programming. Part I: linear constraints. *SIAM J. Appl. Math.*, 8:181–217, 1960.

[28] Ph. L. Toint. Sparsity exploiting quasi-Newton methods for unconstrained optimization. In L. C. W. Dixon, E. Spedicato, and G. P. Szego, editors, *Nonlinear Optimization: Theory and Algorithms*, pages 65–90. Birkhäuser, Boston, 1980.

[29] TOMLAB optimization environment for MATLAB. `http://tomopt.com`.

[30] P. Wolfe. The reduced gradient method. Unpublished manuscript, RAND Corporation, 1962.

[31] P. Wolfe. Methods of nonlinear programming. In J. Abadie, editor, *Nonlinear Programming*, pages 97–131. North-Holland, Amsterdam, 1967.

[32] L. Yang, D. Ma, A. Ebrahim, C. J. Lloyd, M. A. Saunders, and B. O. Palsson. solveME: fast and reliable solution of nonlinear ME models. *BMC Bioinformatics*, 17:391:(10 pages), 2016.

[33] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 78: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Trans. Math. Software*, 23(4):550–560, 1997.