SOLVING LARGE-SCALE OPTIMIZATION PROBLEMS WITH GAMS AND MINOS

Michael A. SAUNDERS
Department of Operations Research
Stanford University
Stanford, California 94305-4022

OPERATIONAL RESEARCH SOCIETY
OF NEW ZEALAND
24TH ANNUAL CONFERENCE
University of Auckland
August 18–19, 1988

Abstract

Operations researchers have long used modeling languages to construct large linear programming models. The modeling system GAMS and the optimization system MINOS now allow the development of large *nonlinear* models with relative ease.

We discuss some applications of GAMS and MINOS. At the same time we promote the philosophy of putting *all* variables and equations into the optimization model, rather than eliminating variables.

TEMPORARY VARIABLES

"World War 2" Fortran

```
      DO 20 J=1,N
      CS=A(J,J)/SQRT(A(J,J)*A(J,J)+Y(J)*Y(J))
      SN=Y(J)/SQRT(A(J,J)*A(J,J)+Y(J)*Y(J))
      JM1=J-1
      IF (JM1.LT.0) GOTO 20
      DO 10 I=1,JM1
      A(I,J)=CS*A(I,J)+SN*Y(J)
   10 Y(J)=SN*A(I,J)-CS*Y(J)
   20 CONTINUE
```

Spot the deliberate error!

Fortran 77

```
      DO 20 J = 1, N
         X    = SQRT( A(J,J)**2 + Y(J)**2 )
         CS   = A(J,J) / X
         SN   = Y(J)   / X

         DO 10  I  = 1, J - 1
            AIJ    = A(I,J)
            A(I,J) = CS*AIJ + SN*Y(J)
            Y(J)   = SN*AIJ - CS*Y(J)
   10    CONTINUE
   20 CONTINUE
```

TWO TYPES OF VARIABLE

Linear least squares

Given matrix A and vector b, find y such that $Ay \approx b$:

$$\underset{y}{\text{minimize}} \quad (b - Ay)^T(b - Ay).$$

Alternatively:

$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & \phi(x, y) = x^T x \\ \text{subject to} & x + Ay = b, \end{array}$$

where x is the "residual vector".

Generalized least squares

$$\underset{y}{\text{minimize}} \quad (b - Ay)^T W^{-1}(b - Ay).$$

Assume $W = LL^T$ is positive definite (or semi-definite):

$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & \phi(x, y) = x^T x \\ \text{subject to} & Lx + Ay = b. \end{array}$$

This is a (special) QP.

- The data A and b appear only once.
- The partial derivatives $\frac{\partial \phi}{\partial x}$, $\frac{\partial \phi}{\partial y}$ are trivial.
- Does not matter if W is singular.
- Often L is known anyway (rather than W).

x, y sometimes called "state" variables and "control" variables.

BLACK BOXES

For given control variables y, there is often "black box" software available to compute x satisfying the nonlinear equations

$$f(x, y) = b.$$

- Electrical power transmission ("Load Flow")
- Oil refinery models
- Steady-state flowsheeting
- Dynamic process simulation
- Design of pipe with "leaks" (Water Studies, Stanford)
- Aquifer reclamation (maybe)

Obvious wish: to optimize with respect to y.

PROCESS OPTIMIZATION

$$\text{minimize}\, \phi(x,y) \quad \text{subject to} \quad f(x,y) = b.$$

The natural inclination is to keep using the *immensely complicated and expensive* black boxes, and simply build an optimization system around them:

0. Set $k = 0$. Choose y^0.
1. Solve $f(x, y^k) = b$ to obtain $x = x^k$.
2. Evaluate objective function $\phi^k = \phi(x^k, y^k)$.
3. Estimate derivatives of objective function wrto y;
   i.e., for each y_j,
   - $\bar{y} \leftarrow y^k + \delta e_j$.
   - Solve $f(\bar{x}, \bar{y}) = b$.
   - $g_j \leftarrow (\phi(\bar{x}, \bar{y}) - \phi^k)/\delta$.
4. Use gradient vector g to obtain a search direction p.
5. $y^{k+1} \leftarrow y^k + \alpha p$ for some steplength α.
6. $k \leftarrow k + 1$. Repeat from Step 1.

This approach is far from ideal.

DISADVANTAGES OF "BLACK BOX" OPTIMIZATION

- The nonlinear equations $f(x, y) = b$ have to be solved many times.
- Gradients are only estimates (not analytic).
- Computation of p and α is therefore less reliable.
- Are "World War 2" black boxes worth saving?
- Some of them already have to "juggle" things to be able to solve $f(x, y) = b$. Might as well let the optimizer do the juggling.

Example in electrical power industry:

- New York Power Authority.
- Load flows are terminated when the iterates start diverge(!).
- "Optimal" power flows can have errors of 300,000 watts (error in satisfying Kirchoff's nonlinear equations).
- Typical error for the General Electic OPF code: 1 watt.

PUT EVERYTHING INTO THE OPTIMIZATION MODEL

A large-scale optimization system (e.g., MINOS) should be able to treat the "black box" constraints directly:

$$\begin{array}{ll} \text{minimize} & \phi(x,y) \\ \text{subject to} & f(x,y) = b, \\ \textit{and} & l \le \begin{pmatrix} x \\ y \end{pmatrix} \le u. \end{array}$$

As in LP, bounds on the variables can be handled easily. (These are what some of the black boxes are busy juggling in an *ad hoc* fashion.)

- The functions in $f(x,y)$ are typically simple.[1]
- A modelling language (e.g., GAMS) can obtain the partial derivatives of $f(x,y)$ analytically.
- $\partial f(x,y)/\partial(x,y) = J(x,y)$, the *Jacobian matrix*.
- Certain oil companies now use GAMS/MINOS (and SLP).
- Refinery models are typically 70–80% nonlinear.
- A three-refinery model has about 900 equations, 1000 variables, 4000 entries in $J(x,y)$.
- Cold-start solve on an IBM 3090 takes 10–20 minutes.

[1] Well ... In a preliminary GAMS refinery model, one equation (i.e., one GAMS statement) filled 20 pages!

GAMS, GAMS/MINOS, GAMS/ZOOM

- General Algebraic Modeling System.
- An equation-oriented modeling language
  for LP, NLP, MIP.
- *GAMS: A User's Guide*,
  Anthony Brooke, David Kendrick and Alex Meeraus,
  with tutorial by Richard Rosenthal,
  The Scientific Press, Redwood City, California, 1988.
- IBM PC, PS/2 and compatibles (DOS).
- IBM mainframes (MVS and CMS),
  DEC VAX systems (VMS and Ultrix),
  CDC (if you must!).
- GAMS is in Pascal, MINOS and ZOOM are in Fortran 77.

STRUCTURED CONTROLLER DESIGN
A GAMS Example

Lyapunov's Theorem (for linear systems)

Solutions of $dz/dt = Az$ are bounded for $t \geq 0$
$\Leftrightarrow$ There exists $X > 0$ such that $A^T X + XA \leq 0$.

Illustration

- A large chemical process.
- A controller might form each control signal (e.g., a pump voltage) based on local information (e.g., temperatures and flow rates near the pump).
- "Structured" controller means there are constraints on X or A.
- Example 1: X should be block-diagonal.
- Example 2: A should be a linear combination of some given matrices:
$$A = A_0 + y_1 A_1 + \cdots + y_k A_k.$$
- Typical objective:
$$\text{minimize} \;\; \text{Trace} \int_0^\infty e^{A(y)^T t} e^{A(y)t} \, dt.$$

Optimization Problem

Find y and a positive-definite matrix X that is Lyapunov stable for a matrix of the form $A = A_0 + y_1 A_1 + \cdots + y_k A_k$ (where $A_0, \ldots, A_k$ are given). Minimize Trace(X).

$$\begin{array}{rcl} \underset{X, y, A, U}{\text{minimize}} & & \Sigma_i X_{ii} \\ \text{subject to } A & = & A_0 + y_1 A_1 + \cdots + y_k A_k \\ X & = & U^T U \\ A^T X + X^T A & = & -I \end{array}$$

- U is upper triangular (Cholesky).
- To make U nonsingular, include bounds on its diagonals: $U_{ii} \geq \delta$.
- A and X are $n \times n$.
  Problem is significant even for $n = 5$, $k = 3$.

ALGORITHMS FOR NONLINEAR CONSTRAINTS

$$\boxed{\begin{array}{ll} \text{minimize} & \phi(x) \\ \text{subject to} & f(x) = b, \quad l \le x \le u. \end{array}}$$

Solve a sequence of linearly constrained subproblems

Assume x_k and λ_k are the kth estimates of x and λ.

Approximate Lagrangian:	$F_k(x) \approx \phi(x) - \lambda_k^T f(x)$.
Linearize constraints:	$f_k(x) \approx f(x_k) + J_k(x - x_k)$.
Solve subproblem:	$\boxed{\begin{array}{l} \min F_k(x) \ \text{ subject to} \\ f_k(x) = b, \ \ l \le x \le u. \end{array}}$
Form search direction:	$p \leftarrow x - x_k$,
	$q \leftarrow \lambda - \lambda_k$.
Step:	$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} \leftarrow \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} + \alpha \begin{pmatrix} p \\ q \end{pmatrix}$.

SLC and SQP algorithms

SLC: Subproblem objective is the Lagrangian.
SQP: Objective is a quadratic approx'n to the Lagrangian.

OPTIMAL POWER FLOW
EVOLUTION OF OPF ALGORITHMS
Rob Burchett at General Electric, Schenectady, NY
SOL Algorithms Group at Stanford

1980 MINOS 4.0 used directly (SLC method).
Subproblems solved with the reduced-gradient method.
Combined LP simplex technology with quasi-Newton method.
(Needs first derivatives; approximates *reduced* Hessian.)

Jul 81 OPF problem solved at GE with 1200 nonlinear constraints.
500 iterations, 1000 evaluations of functions and gradients,
11 hours on VAX 11/780, 0.1 billion page faults (??!).
Never before feasible, let alone optimal.

Aug 81 Replaced P^4 basis factorization with Reid's LA05 package.
(P^4 assumed bases were nearly triangular, but most of the Jacobian is symmetric.)
1200-row problem: 1 hour on VAX 11/780.

Oct 81 EPRI awarded contract to ESCA (not GE).

Feb 82 2200-row problem: 90 mins on VAX.

Mar 82 Philadelphia Electric Company test OPF. Quote:

> "After using the only other two existing OPF programs available to the industry (BPA and PG&E) we have found the GE program to be far superior."

Nov 82 PG&E test OPF in San Francisco.
Nonlinear constraints satisfied to machine precision.
Many different starting point gave same solution!

Mar 83 OPF problem solved with NPSOL (dense SQP method).

340-row problem:
6 QP subproblems, 300 QP iterations,
8 evaluations of functions + 1st and 2nd derivatives.

1983 MINOS 5.0 adapted to solve QP subproblems with exact 1st and 2nd derivatives (OPF version 2.5).
Still used quasi-Newton to approximate the reduced Hessian.

2200-row problem:
7 QP subproblems, 900 iterations, 4 hours on VAX.

1984 OPF 3.0 solves QPs using full Kuhn-Tucker system:

$$\begin{pmatrix} 0 & J_k \\ J_k^T & H_k \end{pmatrix} \begin{pmatrix} \lambda \\ -p \end{pmatrix} = \begin{pmatrix} 0 \\ g \end{pmatrix}.$$

Used Harwell code MA27 on symmetric indefinite system.
Factors repacked to suit MINOS factorization routines to allow sparse LU updates each iteration.

Up to 3 times faster than OPF 2.5.
Sporadic trouble with indefiniteness, singularity.

1984 OPF 3.1: "Artificial column" method for Phase 1.
(Improves objective during Phase 1.)

1984 OPF 3.2: Schur-complement updates to KT factorization.
Up to 5 times faster than OPF 2.5.

1985 Good ordering for MA27 obtained using 1/4 of KT matrix.

1985 SECURITY CONSTRAINED DISPATCH (SCD).
Allow for K contingencies (transmission breakdowns).
Problem is K times as large ($K = 1$–10).

Benders decomposition: QP master problem as in OPF,
LP subproblems generate cuts to be added to QP.

Feb 85 Wrote to NZE Wellington asking if we could obtain data for NZ power network. Requested IEEE standard format if possible.

Apr 85 NZE wrote to GE asking for definition of IEEE format.

May 85 GE sent format to NZE (now Electricity Division, Ministry of Energy), asked for data on tape. Offered to do the conversion to IEEE if necessary.

Jan 86 Met with Power Transmission people at NZE.
Happy there with Load Flows.
(No optimization. Can't measure load voltages better than 1%.)
Asked for NZ data.

1986 MicroVAX II acquired for SCD project.
Can handle problems with 5000 rows.
(New Zealand system would be about 500 rows.)

Feb 87 Met with Power Transmission people at NZE (now Electricorp).
Still happy with Load Flows. They estimate 3% saving in transmission losses = $NZ 2.4 million annually.
Asked for NZ data.

Apr 87 Moved office at Stanford, threw away lots of listings.
Acquired 4 VAXstation IIs at SOL (courtesy of US Army).
Experience with Rob's machine and VMS a large influence.

Apr 87 Wrote to NZE apologizing for short notice last visit.
Asked for NZ data.

Apr 87 NZE sent GE printouts(!) of typical NI and SI power flows, in IBM PSP format.

Jul 87 *Shock*: Rob resigned from GE to set up his own company.
Began work on totally new code. Recriminations rife.

Dec 87 Trip to GE (Schenectady NY) to try to resurrect SCD.
Hopeless without Rob. (One line of code changed.)

May 88 Prospective first sale for Rob's new system—to NY company who market a large Energy Management System containing a Load Flow code but no OPF. Current customers 150, including NZE!

Jun 88 More successful effort to resurrect SCD.
(Recovery when MA27 says KT system is singular.)

Jun 88 Temporary truce reached between Rob and GE.

1999 Optimal Power Flow in NZ?

MAIN THEMES

Remember your temporary variables (x).
More equations but simpler functions, sparser Jacobians:

$$\begin{aligned} Bx + Cy &= b \quad (B \text{ nonsingular}) \\ Dx + Ey &= c \\ f(x, y) &= d \end{aligned}$$

General sparse-matrix methods can handle several thousand variables and constraints, and the nonlinear constraints do not have to be satisfied throughout—only in the limit (at an optimal solution).
One day, GAMS might be able to generate both $\partial f/\partial(x, y)$ and $\partial^2 f/\partial x^2$, $\partial^2 f/\partial x \partial y$, etc.

- *Think large-scale* (don't eliminate variables).
- *Think nonlinear* (let GAMS do the derivatives).