The Dual Flow Between Linear Algebra and Optimization

Margaret H. Wright
Computer Science Department
Courant Institute of Mathematical Sciences
New York University

History of Numerical Linear Algebra Minisymposium - Part II
SIAM Conference on Applied Linear Algebra
Monterey, California
October 28, 2009

This is the eighth of eight talks whose collective purpose is

> . . . to look back at the history of the subject, to remember the founders of our field, to honor their legacy, and to review their accomplishments to ensure a vital future for our field.

Many thanks to Ilse and Sven for inviting me to speak!

My original goal for this talk: to discuss several of the numerous two-way connections between applied linear algebra and optimization.

This turned out to be much too ambitious for a 25-minute talk.

The compromise approach—a quick whisk through two topics:

1. rank-one updates in the simplex method for linear programming and
2. convexification in nonlinear optimization (what to do with indefinite matrices that "should be" positive definite).

Among the mathematically equivalent forms of linear programs, we consider standard form:

$$\underset{x \in \mathcal{R}^n}{\text{minimize}} \;\; c^T x \quad \text{subject to} \quad Ax = b \;\text{ and }\; x \geq 0,$$

where A is $m \times n$ and has rank m.

The point $x^* \in \mathcal{R}^n$ is optimal if

1. $Ax^* = b$ and $x^* \geq 0$;
2. $c = A^T y^* + z^*$, where $z^* \geq 0$ and $z_j^* x_j^* = 0$ for $j = 1, \ldots, n$.

The m-vector y^* is a Lagrange multiplier for the equality constraints $Ax = b$ and the n-vector z^* is a Lagrange multiplier for the bound constraints $x \geq 0$.

Basic and Nonbasic Variables

A *basic set* $\mathcal{B}$ is a set of m distinct indices such that the m columns of A designated by $\mathcal{B}$ form a *nonsingular* matrix. The submatrix defined by $\mathcal{B}$ is denoted by B, the columns of B are called the *basic columns* of A, and B itself is called a basis.

The *nonbasic set* $\mathcal{N}$ contains the $n - m$ indices of columns of A that are not in $\mathcal{B}$. The $m \times (n - m)$ matrix of nonbasic columns of A is denoted by N.

The subscripts $\mathcal{B}$ and $\mathcal{N}$ define subvectors and submatrices associated with basic and nonbasic indices, such as $z_{\mathcal{N}}$.

A key theoretical result: the optimal solution of an LP (essentially always) lies at a *vertex*, a feasible point where n linearly independent constraints are active.

For standard form, this means that, at every vertex, at least $n - m$ of the bound constraints must be active (in addition to the m equality constraints).

Inspired by this property, George Dantzig's great invention in 1947 was the *simplex method*, which solves linear programs by starting at a vertex and moving from vertex to vertex, reducing the objective function as it goes.

The mechanisms used by the simplex method in moving from vertex to vertex are (i) properties of a basic set, (ii) the need for all iterates to satisfy $Ax = b$, and (iii) the nonnegativity requirement for z at an optimal point.

1. Solve for the multipliers y from $B^T y = c_{\mathcal{B}}$, and calculate $z_{\mathcal{N}} = c_{\mathcal{N}} - N^T y$.

2. If $z_{\mathcal{N}} \geq 0$, x is optimal. Otherwise, pick a nonbasic variable that becomes basic by choosing s such that $[z_{\mathcal{N}}]_s < 0$.

3. Calculate the search direction for the basic variables from $Bp_{\mathcal{B}} = -a_{\nu_s}$, where ν_s is the sth index in $\mathcal{N}$ and a_{ν_s} is column ν_s of A.

4. Step along $p_{\mathcal{B}}$ (to remain feasible) to the next iterate, which is determined by a blocking basic variable with index β_t that will become nonbasic.

5. Update the basic and nonbasic index sets.

   $B \leftarrow B$ with column t replaced by column ν_s of A.

   $N \leftarrow N$ with column s replaced by column β_t of A.

The linear algebraic calculations associated with the simplex method are:

- At each iteration, two square linear systems need to be solved, involving B and B^T.
- If x is not optimal, a column of B is replaced by a column of N, producing the new basis for the next iteration.

And, as we all know, column replacement is a special rank-one modification.

Key names and dates relevant to linear algebra in the early years of the simplex method:

1947 Dantzig: the simplex method.

1947 Project SCOOP (Scientific Computation of Optimal Programs), a US Air Force research group; disbanded in 1955.

1952 Orden: the product form of the inverse (PFI). "...the simplex procedure may be considered to be a generalization of the elimination process".

1954 Orchard Hays: programmed the simplex method for an IBM CPC (Card-Programmed Calculator).

1957 Markowitz: the elimination form of the inverse, designed for sparse matrices.

For detailed and fascinating information about the early days of LP, I recommend

S. Gass (2002), “The first linear-programming shoppe”, *Operations Research*.

S. Gass and A. Assad (2005), *An annotated timeline of operations research*.

Consider the hardware available for pioneering simplex codes:

Computers read decks of punched cards, which were run on card-programmed calculators (CPCs), formed by connecting standard IBM office accounting equipment.

In 1950, there were 80 mechanical counters of the accounting machine (an IBM tabulator). There was additional storage for 16 10-digit numbers relayed from a Type 604 electronic calculator, which could perform about 2,000 additions or subtractions per second, and 86 multiplications or divisions per second.

The Gauss-Seidel CPC procedure inverted a 10×10 matrix in 30 minutes to 7 decimal places.

How did early LP codes work in this environment?

“In the revised simplex method both the inverse and inverse transpose of a ‘basic’ matrix are needed; more significant, however, is the fact that each iteration replaces one of the columns of the basis. In the product form of representation, this change can be conveniently effected by multiplying the previous matrix by an elementary matrix. Thus, only one additional column of information need be recorded with each iteration. . . . Using the IBM Card Programmed Calculator, a novel feature results: when the inverse matrix is needed at one stage and its transpose at another, this is achieved simply by turning over the deck of cards representing the inverse.”

From “The product form for the inverse in the simplex method”, Dantzig and Orchard-Hays, the RAND Corporation, 1953.

Orden (1952) first proposed using the so-called product form of the inverse in Gauss-Jordan inversion, and then

> I noticed that it was neither necessary nor desirable to fully invert the matrix—leaving it as the product form of the inverse cut the amount of computation in half. When I joined project SCOOP, there was the simplex algorithm waiting for the same device.

With Markowitz's elimination form of the inverse, the solution p of $Bp = b$ is expressed as $p = B^{-1}b = E_k \dots E_2 E_1 b$, where each E_j is the identity except for one sparse column. After replacing a column of B to produce $\bar{B}$, the solution of $\bar{B}\bar{p} = \bar{b}$ is given by

$$\bar{p} = \bar{B}^{-1}\bar{b} = E_{k+1}E_k \dots E_2 E_1 \bar{b}.$$

Only in the late 1960s (it appears), did linear algebra researchers enter the picture!

1969 Bartels and Golub: *updating the LU factors in stabilized product form*, meaning that the modified matrix $\bar{B}$ is represented as a product involving L, U, and stabilized Gauss transformations, where U is updated explicitly.

1972 Forrest and Tomlin: fast updates of LU in LP for sparse matrices. A restricted form of Bartels-Golub; much less concerned with stability. By far the most popular choice in sparse implementations of simplex.

1982 Reid: sparse and efficient implementation of Bartels-Golub.

1984 Fletcher and Matthews: updates of both L and U.

Sadly, there does not seem to be an “easy” (to present, or to implement) way to explain stable updating of LU because we need to allow interchanges.

In keeping updates to L in product form but explicitly updating U, stabilized elementary transformations are used. A new U is typically obtained by going from column spike to upper triangle via upper-Hessenberg form, or from column spike to upper triangle via row spike.

Standard practice is to recompute the LU factors from scratch after a specified number of updates, or sooner if there is unacceptably large growth in the elements of the updated upper triangle.

Lively research continues today on low-rank updates in large-scale active-set methods for nonlinear optimization, involving rectangular matrices.

And now for the second topic: strategies for “convexification” in Newton-based methods.

A reminder: When minimizing the twice-differentiable function $f(x)$, $x \in \mathcal{R}^n$, the pure Newton step p at the point x satisfies the linear system

$$H(x)p = -g(x)$$

where $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$.

If H is positive definite, p is the step to the (unique) minimizer of a local strictly convex quadratic model.

Famous for its local quadratic convergence.

In Courant's deservedly famous 1943 address to the American Mathematical Society, "Variational methods for the solution of problems of equilibrium and vibrations", he notes that

> . . . the n parameters c_i which we may now determine from the ordinary minimum problem of the calculus, . . . [which] leads to a system of linear equations for the parameters, a system which may be solved by established methods.

No mention of any special properties of the matrix.

Early papers related to Newton's method, in which side comments are made about the need for the Hessian (when minimizing) to be positive definite:

1955 Crockett and Chernoff, Gradient methods of maximization, *Pacific Journal of Mathematics*.

1962 Spang, A review of minimization techniques for nonlinear functions, *SIAM Review*.

Newton's method was clearly "in the air" by the mid- to late 1950s, e.g., in Hildebrand's (1956) textbook on numerical analysis.

But what happens when H is not positive definite???

How to define a good Newton-based direction when the local quadratic model is not convex?

In linear algebraic terms, what to do if a "Hessian", or Hessian-like matrix, is not positive definite, or, more generally, does not have the "correct" inertia?

When did optimizers thinking about linear algebra enter this picture?

1967 Greenstadt, On the relative efficiencies of gradient methods, *Mathematics of Computation*.

Compute the eigenvalues of H, flip the "negative enough" ones, and set the "too small" ones to a reasonable small value.

1974 Gill and Murray, Newton-type methods for unconstrained and linearly constrained optimization, *Mathematical Programming*.

Replace H by $H + E$, where E is diagonal, computed via a modified Cholesky factorization.

1979 Moré and Sorensen, On the use of directions of negative curvature in a modified Newton method, *Mathematical Programming*.

Based on the symmetric indefinite factorization.

1981 Gill, Murray, and MHW, *Practical Optimization*.

Modified Cholesky with symmetric interchanges.

1990 Schnabel and Eskow, A new modified Cholesky factorization, *SIAM Journal on Scientific and Statistical Computing*.

Based on Cholesky, exploiting Gerschgorin disks.

1984 The interior-point revolution in optimization!

Special kinds of matrices become of great interest.

1995 Forsgren, Gill, and Murray, Computing modified Newton directions using a partial Cholesky factorization, *SIAM Journal on Scientific Computing*.

Based on the symmetric indefinite factorization.

1998 Cheng and Higham, A modified Cholesky algorithm based on a symmetric indefinite factorization, *SIAM Journal on Matrix Analysis and Applications*.

1999 Schnabel and Eskow, A revised modified Cholesky factorization algorithm, *SIAM Journal on Optimization*.

2008 Fang and O'Leary, Modified Cholesky algorithms: a catalog with new approaches, *Mathematical Programming*.

Plus an unending stream of MANY papers in optimization about regularization techniques in primal-dual and sequential quadratic programming methods.

Let's focus only on the question of how to make a non-positive definite matrix positive definite.

This includes ways to determine (at reasonable cost) whether the matrix is in fact positive definite, or even "sufficiently" positive definite.

A big stumbling block: it's not clear what the ideal solution would be—if expense were no object, what would happen to an indefinite matrix that "should be" positive definite?

And what does “should be positive definite” mean?

Something to keep in mind: Newton’s method in which H is always positive definite does not always perform well.

If H is sufficiently positive definite, at least the pure Newton direction p is a descent direction. But this does not mean that it is a **good** direction.

On the bright side, because the problem we wish to solve is not well defined, there is great scope for new methods based on intuitive and possibly hand-waving rationales.

To say something that we all know: when H is “sufficiently” positive definite, Cholesky without interchanges is *guaranteed to be numerically stable*

$$H = LDL^T,$$

with L unit lower-triangular, D a positive diagonal, and an *a priori* bound on elements of L and D in terms of elements of H.

But otherwise, without positive-definiteness, there’s the famous 2×2 example:

$$H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

where LDL^T does not exist.

And, if H is indefinite, even if LDL^T exists, there is no *a priori* bound on the elements of L or D, as we know from

$$H = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix} = LDL^T,$$

for which

$$L = \begin{pmatrix} 1 & \\ 1/\epsilon & 1 \end{pmatrix} \quad \text{and} \quad D = \text{diag}(\epsilon, \quad \epsilon - 1/\epsilon).$$

What's the lesson?

Playing around with factorizations of indefinite matrices can be very tricky!

Doesn't everyone know this???

Apparently not. . . A very popular software package for nonlinearly constrained optimization replaces an indefinite Hessian H of the Lagrangian with $H + \delta I$, $\delta > 0$, where δ is chosen as follows:

1. Compute the LDL^T of H.
2. Scan the diagonal elements of D, looking for any of the wrong sign; let d^* denote the largest in magnitude of these.
3. Choose δ as a multiple of d^*.

This is almost identical to a similarly bad suggestion made by Fiacco and McCormick (1968).

The horrible effects can be seen with the 2-variable example of minimizing $x_1 x_2$ subject to $0 \le x_1^3 \le 2$ and $-1 \le x_2^3 \le 3$.

The solution is

$$x^* = \begin{pmatrix} \sqrt[3]{2} \\ -1 \end{pmatrix}, \quad \text{with} \quad f^* = -1.2599.$$

At $x_0 = (10^{-7},\ -10^{-7})$, with $\lambda_0 = (1, 1, 1, 1)$, the Hessian of the Lagrangian is

$$\begin{pmatrix} -1.2 \times 10^{-4} & 1.0 \\ 1.0 & 6.0 \times 10^{-5} \end{pmatrix},$$

which is very similar to the 2×2 example with large elements of L and D.

Verbatim extracts from a run with the latest version of this software on NEOS (Network-Enabled Optimization System), made on October 20, 2009:

```
nonconvex subproblem:  diag[ 5] = 1.50e+00
pert1 = 1.650000e+00

nonconvex subproblem:  diag[ 5] = 4.00e+01
pert1 = 4.397297e+01

nonconvex subproblem:  diag[ 5] = 8.00e+02
pert1 = 8.800166e+02

nonconvex subproblem:  diag[ 5] = 1.60e+04
pert1 = 1.760056e+04

nonconvex subproblem:  diag[ 5] = 3.20e+05
pert1 = 3.520167e+05
```

```
nonconvex subproblem:  diag[ 5] = 6.40e+06
pert1 = 7.040514e+06

nonconvex subproblem:  diag[ 5] = 1.28e+08
pert1 = 1.408173e+08

nonconvex subproblem:  diag[ 5] = 2.56e+09
pert1 = 2.816658e+09

nonconvex subproblem:  diag[ 5] = 1.00e+10
pert1 = 1.100543e+10

nonconvex subproblem:  diag[ 5] = 1.00e+10
pert1 = 1.100419e+10
```

[23 more iterations with perturbations of size 10^{10} added to the relevant matrix; then the method gradually rallies.]

```
nonconvex subproblem:  diag[ 5] = 8.82e+09
pert1 = 9.701931e+09

nonconvex subproblem:  diag[ 5] = 8.62e+08
pert1 = 9.483910e+08

nonconvex subproblem:  diag[ 5] = 2.38e+07
pert1 = 2.623226e+07

nonconvex subproblem:  diag[ 5] = 6.08e+05
pert1 = 6.693377e+05

nonconvex subproblem:  diag[ 5] = 1.55e+04
pert1 = 1.702972e+04

nonconvex subproblem:  diag[ 5] = 3.95e+02
pert1 = 4.349272e+02

nonconvex subproblem:  diag[ 5] = 1.01e+01
pert1 = 1.112929e+01
```

[42 more iterations]

```
OPTIMAL SOLUTION FOUND
...optimal solution (84 iterations, 84 evaluations)
primal objective -1.259921085
  dual objective -1.259920712
x [*] :=
1   1.25992
2  -1
;
```

Strategies for treating non-convexity are complicated, with no single strategy guaranteed to succeed every time.

Even for IPOPT (one of today's best interior-point codes for nonlinearly constrained optimization), difficulties related to nonconvexity in this 2-variable problem arise, and are revealed via the linear algebra.

```
*****************************************************
*** Solving the Primal Dual System for Iteration 0:
*****************************************************

In Ma27TSolverInterface::Factorization: negevals_ = 5,
but numberOfNegEVals = 4

Factorization failed with retval = 2

In Ma27TSolverInterface::Factorization: negevals_ = 5,
but numberOfNegEVals = 4

Factorization failed with retval = 2

In Ma27TSolverInterface::Factorization: negevals_ = 5,
but numberOfNegEVals = 4

Factorization failed with retval = 2
```

```
In Ma27TSolverInterface::Factorization: negevals_ = 5,
but numberOfNegEVals = 4

Factorization failed with retval = 2

Factorization successful.

Number of trial factorizations performed:  5
```

Similar difficulties with non-convexity arise for the first 24 iterations.

```
**************************************************
*** Solving the Primal Dual System for Iteration 24:
**************************************************

In Ma27TSolverInterface::Factorization: negevals_ = 5,
but numberOfNegEVals = 4

Factorization failed with retval = 2

Factorization successful.

Number of trial factorizations performed:  2
```

The matrix finally becomes “nice” at iteration 25, and remains so thereafter.

```
****************************************************
*** Solving the Primal Dual System for Iteration 25:
****************************************************
Factorization successful.
Number of trial factorizations performed:  1
```

Success at iteration 37.

```
Number of Iterations....: 37
Number of objective function evaluations             = 42
Number of objective gradient evaluations             = 35
Number of inequality constraint Jacobian evaluations = 39
EXIT: Optimal Solution Found.

x [*] :=
1   1.25992
2  -1
```

Intense efforts continue to develop improved convexification/regularization techniques, driven by plausible motivation, with the goal of generating linear systems and, in some instances, other linear algebraic subproblems, that produce efficient and reliable optimization methods.

In the spirit of this session, we should salute the pioneers of linear algebra and optimization for their many accomplishments, which produced today's lively research areas and their connections.

Linear Algebra and Optimization:

Together Forever!