Symbiosis between Linear Algebra and Optimization*

Dianne P. O'Leary†

May 27, 1999

Abstract

The efficiency and effectiveness of most optimization algorithms hinges on the numerical linear algebra algorithms that they utilize. Effective linear algebra is crucial to their success, and because of this, optimization applications have motivated fundamental advances in numerical linear algebra. This essay will highlight contributions of numerical linear algebra to optimization, as well as some optimization problems encountered within linear algebra that contribute to a symbiotic relationship.

1 Introduction

The work in any continuous optimization algorithm neatly partitions into two pieces: the work in acquiring information through evaluation of the function and perhaps its derivatives, and the overhead involved in generating points approximating an optimal point. More often than not, this second part of the work is dominated by linear algebra, usually in the form of solution of a linear system or least squares problem and updating of matrix information.

Thus, members of the optimization community have been *consumers* of linear algebra research, and their needs have set some research directions for the computational linear algebra community. Recently, this relationship has entered a new phase in which optimization problems arising in linear algebra are also setting research agendas for the optimization community.

This essay will highlight the advances in numerical linear algebra that contributed to this symbiotic relationship. First, in Section 2 we review the modeling problems that give rise to linear algebra problems. Least squares modeling

*This work was completed at the Departement Informatik, ETH Zürich, Switzerland. This work was also supported in part by the U.S. National Science Foundation under Grant CCR-97-32022.

†Dept. of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 (oleary@cs.umd.edu)

is the subject of Section 3. We turn our attention to the linear algebra of unconstrained optimization problems in Section 4, and then review the simplex method for linear programming in Section 5. Section 6 discusses linear algebra problems arising in interior point methods. Nonlinear problems are briefly considered in Section 7. Section 8 concerns linear algebra problems giving rise to optimization, and Section 9 discusses computational issues in optimization. We summarize our survey in Section 10.

2 Linear and Quadratic Models

The modeling of complex phenomena in science and economics by linear and quadratic models is ubiquitous. It is motivated by the Taylor series expansion of a thrice continuously differentiable function $f : \mathcal{R}^n \to R$ as

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T f''(x_0)(x - x_0) + O(\|x - x_0\|^3),$$

as well as by the relative ease of handling these models rather than fully-nonlinear ones. Often the full nonlinearity of f is neglected in the modeling process, either because the simpler models yield sufficient accuracy or because the modeling process yields insufficient information about the higher order coefficients.

We often determine the coefficients in a linear model by obtaining the "best possible" fit to experimental data. The coefficients can be highly dependent on our way of measuring "best." In general, given a model $M(t, z)$ of some function $y(t)$, and data (t_i, y_i), $i = 1, \ldots, m$, we try to determine the model coefficients $z \in \mathcal{Z} \subseteq \mathcal{R}^p$ to minimize the norm of the residual vector, whose entries are

$$r_i = y_i - M(t_i, z), \quad i = 1, \ldots, m.$$

Common choices of the norm are the 1-norm or infinity-norm, leading to linear programming problems (See §5) or the 2-norm, leading to a linear least squares problem (See §3). Narula [68] discusses solution of these various regression problems. If the set $\mathcal{Z}$ is something other than $\mathcal{R}^p$, then there are constraints on the minimization problem.

Thus, modeling of physical phenomena leads to optimization problems, but, conversely, algorithms for optimization often lead to linear and quadratic modeling. For instance, an objective function $f(x)$ might be locally modeled as a quadratic function in algorithms such as sequential quadratic programming. As another example, we often temporarily replace a constraint by a local linear model in order to make a subproblem easier to solve (See §7).

Perhaps the oldest use of quadratic models to solve nonlinear problems is the iteration of Newton for minimizing a function or finding a zero of a system of nonlinear equations. At each step in the iteration, we construct a quadratic model of the function (or a linear model of the system of equations) and use that model to generate a step in the direction of a better solution. A wonderful survey of Newton's method is given in [96], and we consider this method in §4.

3 Least Squares

Consider the modeling problem

$$\min_z \|Mz - y\|_2 \tag{1}$$

where $M \in \mathcal{R}^{m\times n}$, $z \in \mathcal{R}^n$, and $y \in \mathcal{R}^m$. This *linear least squares problem* was shown by Gauss [38, 37, 39] to produce the z that yields the best linear unbiased estimator of any function $c^T z_{true}$ whenever the errors in y have mean zero and variance $\sigma^2 I$.

The oldest algorithms for solving the linear least squares problem can be viewed as applying direct or iterative methods to solve the *normal equations*

$$M^T M z = M^T y\,,$$

obtained by setting the derivative of (1) to zero.

Within the past 50 years, advances in the solution of least squares problems have been of three types: analysis of sensitivity and stability, development of computational tools, and consideration of problem variants.

The lucid textbook by Björck [8] is the definitive reference on the entire subject of numerical solution of least squares problems, and we recommend it for exposition and further references. Higham [48] is an alternate source for the history of sensitivity analysis for these problems.

3.1 Sensitivity and Stability of Least Squares Problems

Important contributions to the study of sensitivity of least squares problems have been made in recent years.

Wedin [94, Thm. 5.1], studied the normwise perturbation of z and the residual $r = y - Mz$ when M is perturbed, showing that if the relative perturbations in M and y are less than ϵ, and if the condition number $\kappa_2(M)$ (the ratio of its largest to its smallest singular value) satisfies $\kappa_2(M)\epsilon < 1$, then

$$\begin{aligned} \frac{\|z-\hat{z}\|}{\|z\|} &\le \frac{\kappa_2(M)\epsilon}{1-\kappa_2(M)\epsilon}\left(2 + (\kappa_2(M)+1)\frac{\|r\|_2}{\|M\|_2\|z\|_2}\right), \\ \frac{\|r-\hat{r}\|}{\|y\|} &\le (1+2\kappa_2(M))\epsilon \end{aligned}$$

This result says that if the residual is small, then perturbations are proportional to $\kappa_2(M)$, but if the residual is large, then perturbations proportional to $\kappa_2(M)^2$ might be seen, and that is indeed the case.

Further analysis can be found in [8, Chapter 1], including component-wise bounds on the error [4].

3.2 Computational Tools for Least Squares Problems

The main computational algorithm for least squares solves the problem by using the QR factorization of the matrix M into the product of a matrix $Q \in \mathcal{R}^{m \times n}$ with orthogonal columns, and an upper triangular matrix $R \in \mathcal{R}^{n \times n}$. Use of this tool was first proposed by Golub [41], but great attention has been given to the relative advantages of factorization using Householder reflections, Givens rotations, or modified Gram-Schmidt [8, Sec. 2.4]. The first two alternatives were known to have similar desirable error properties, and modified Gram-Schmidt was finally shown stable in a paper of Björck and Paige [10] by exploiting the fact, known to many early practitioners such as Sheffield, that modified Gram-Schmidt is numerically equivalent to Householder QR applied to the matrix

$$\begin{bmatrix} 0 \\ M \end{bmatrix}.$$

If the problem is difficult in the sense that M is ill-conditioned, then more refined tools are needed. The QR factorization with column pivoting [41] can be used to try to identify the most linearly independent columns first and perhaps construct a model of reduced size; see [18] for a survey of such rank-revealing QR factorization methods. This is not foolproof, however, and the singular value decomposition [42, §2.5] is a more reliable (and more expensive) factorization algorithm for identifying dependencies; see Stewart [85] for historical remarks on the SVD.

The LU factorization of M can also be used to solve least squares problems [77], but its use is not common except when the matrix M is sparse, with many zero elements. In that case, the QR factors may be quite dense, due to creation of nonzeros in the course of the factorization. To minimize this fill-in, it is important to use the best algorithms for reordering the rows and columns of the matrix [29] before factorization.

The normal equations can be solved by Cholesky factorization into the product of a lower triangular matrix times its transpose, but if the problem is large and sparse, then reordering strategies should again be used to minimize fill [40].

An alternate to factorization for large sparse problems is the use of iterative methods. The preconditioned conjugate gradient algorithm [42] can be used to solve (8), and row-action methods [17] and other specialized methods such as CGLS and LSQR avoid forming the normal equations [8, Chapter 7].

3.3 Variants of Least Squares Problems

Often the matrix M has special structure that can be exploited in order to make solution of the least squares problem more efficient. One example is the matrix that arises from fitting polynomials using the power basis and equally spaced data points. The resulting matrix for the normal equations, a Vandermonde matrix, has beautiful structure but is quite ill-conditioned [11, 9, 27, 47]. A second example is the band matrix structure that results from fitting functions whose support is local [80, 23]. Wavelet [20] and Fourier bases often give matrices with small displacement rank [51] again leading to efficient solution algorithms [86, 24, 67, 44, 76].

Some models give rise to nonlinear least squares problems

$$\min_z \|r(z)\| ,$$

where $r : \mathcal{R}^n \to \mathcal{R}^m$. These are usually solved by Newton variants discussed in Section 4.

Constrained least squares problems also arise frequently in practice. For instance if the parameters z are constrained to be nonnegative, then the resulting least squares problem is a special case of quadratic programming

$$\min_z \frac{1}{2} z^T M z + z^T w , \tag{2}$$

$$Cz \geq d ,$$

and efficient algorithms for solving such non-negativity constrained least squares problems were first proposed by Lawson and Hanson [59]. Alternatively, if the vector z is constrained in 2-norm, then this results in a quadratic objective function with a single quadratic constraint. This is the situation, for example, in trust region methods for optimization (See §4).

Often a sequence of least squares problems needs to be solved, each representing an update of the previous one due to the addition of new data or the downgrading of the importance of old data. Such problems arise, for example, in signal processing when we try to estimate the position of an unknown number of signal sources (e.g., finding the position of each aircraft within a given zone) given data from a set of receivers. Updating and downdating can be done quite stabily if the full QR factorization is saved; in this case, Q is $m \times m$. If this is too expensive, then a variety of algorithms have been proposed that have decent numerical properties [8, Chap. 3].

The weighted least squares problem

$$\min_z \|Mz - y\|_W ,$$

where $\|x\|_W^2 = x^T W x$, is also useful in practice. Here W is an estimate of the inverse covariance matrix for the zero-mean errors in measurement of y. The normal equations become

$$M^T W M z = M^T W y ,$$

and if we introduce the residuals $s = W(y - Mz)$, then we can transform the normal equations into an augmented system

$$\begin{bmatrix} W^{-1} & M \\ M^T & 0 \end{bmatrix} \begin{bmatrix} s \\ z \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}. \tag{3}$$

We will see this system again in Section 6.

If there are outliers in the data, then the least squares criterion is not very useful unless the weights are adjusted so that the outliers do not affect the fit very much. This is the goal in *iteratively reweighted least squares*, or *robust regression* [50], in which the fixed weight matrix W is replaced by some function of the size of a component of the residual

$$\min_z \sum_{i=1}^{m} w(y_i - (Mz)_i)\,.$$

If $w(u) = u^2$, then we recover the least squares problem. Functions that diminish the effects of outliers include Huber's choice [49]

$$w(u) = \begin{cases} u^2/2, & |u| \leq \beta, \\ \beta|u| - \beta^2/2, & |z| > \beta, \end{cases}$$

where β is a problem dependent parameter. Minimizing Huber's function leads to a quadratic programming problem. Computational issues arising in iteratively reweighted least squares problems are discussed, for example, in [73].

4 Unconstrained Optimization

Given a point x_0 and a quadratic model of a function

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^T f''(x_0)(x - x_0)\,,$$

it is natural to approximate the minimizer of f by the minimizer of this model. If $f''(x_0)$ is positive definite, this minimizer is given by

$$x = x_0 - f''(x_0)^{-1} f'(x_0)\,.$$

This equation motivates Newton's method. Given x_0, we define a sequence of iterates

$$x_{k+1} = x_k + p_k$$

where the direction p_k is the solution to the linear system

$$\bar{B}_k p_k = -f'(x_k)\,. \tag{4}$$

and $\bar{B}_k = f''(x_k)$. If f is quadratic, then x_1 is a stationary point of f, a global minimizer if f'' is positive definite. If f is not quadratic, the procedure is convergent at a quadratic rate of convergence to a local minimizer of f under conditions such as those of the Newton-Kantorovich theorem [74, §12.6]

Developments of Newton's method during the last forty years have focussed on improving this method by making it more reliable and by adapting it for use on very large problems.

4.1 Making Newton's Method More Reliable: Line Searches and Trust Regions

Two methods have been used to make Newton's method (or its variants) globally convergent to a local minimizer: line searches and trust regions.

In the line search method, the Newton-like direction p_k is scaled so that

$$x_{k+1} = x_k + \alpha_k p_k \,,$$

where α_k is a parameter chosen to ensure that the objective function f decreases sufficiently in proportion to the size of the step. See [74, §14.4.3] for conditions on α_k that guarantee global convergence (e.g., Wolfe conditions, Goldstein-Armijo conditions).

The trust region method constrains the length of the step so that we do not exit some region in which we "trust" the accuracy of the quadratic model. Thus we solve the problem

$$\min_p M(x_k + p)\,,$$

$$\|p\| \le \tau\,,$$

where M is the quadratic model and τ is the radius of the trust region. If the constraint is active, then the solution to this problem is

$$(\bar{B}_k + \lambda I)p = -f'(x_k)$$

for some nonnegative parameter λ chosen to make $\|p\| = \tau$. This problem can be solved by eigendecomposition of $\bar{B}_k$, but this is generally too expensive. Often an iterative approach is used; we generate a sequence of approximations to p, stopping and backtracking when the norm of p exceeds τ; see, for example, [71, p.320]. This does not give a step in the Newton direction unless the radius of the trust region exceeds the norm of tne Newton direction p_k defined in (4).

4.2 Making Newton's Method More Reliable for Nonconvex Functions

If the matrix $\bar{B}_k$ used in the Newton equation is not positive definite, then Newton's method may fail to have a downhill search direction. To remedy this, algorithms based on line search usually diagnose indefiniteness as (4) is solved and cure it by adding a small correction matrix. These techniques are easily incorporated into a Cholesky factorization of the matrix [71, p.320].

Another approach to making Newton's method more reliable is to take very small steps – in fact, to follow the path

$$\frac{dx}{dt} = -f''(x)^{-1} f'(x)$$

starting with $x(0) = x_0$. This is the idea behind methods such as homotopy methods [54], which also introduce a parameterized function in order to locate multiple local minimizers. The linear algebra is heavily drawn from that used in ordinary differential equation solvers [93].

4.3 Adapting Newton's Method for Large Problems.

Computing, storing, and factoring the Hessian matrix may be impractical if the size is large. *Quasi-Newton* methods mimic Newton's method by generating less expensive approximations B_k to the matrix $\bar{B}_k$. These approximations are generated by updating the approximation for $\bar{B}_{k-1}$, and some have come to be interpreted as matrix approximation problems [28]. The most popular quasi-Newton variant is that proposed by Broyden, Fletcher, Goldfarb, and Shanno (BFGS), which is defined by the update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

where y_k is the change in gradient and s_k is the change in x.

An alternative is to use $\bar{B}_k$ but avoid factoring it. This can be achieved by computing an approximate Newton search direction p_k as the solution to (4) using an iterative method (e.g., conjugate gradients or some other Krylov subspace method) that uses $\bar{B}_k$ only for matrix-vector products. If a conjugate gradient algorithm is used, then there are very effective ways to handle indefiniteness and to determine the step for the trust region. The iteration can be terminated if the increment d to p is a direction of negative curvature (i.e., $d^T \bar{B}_k d < 0$) or if the algorithm steps out of the trust region [71, p.320] Preconditioning can be used to improve convergence of the iterative methods [42, §10.3], but how this biases the generation of directions is an open question.

If $\bar{B}_k$ is too expensive to compute or store, then the necessary products in the iterative method can be approximated by difference quotients

$$\bar{B}_k p = f''(x_k) p \approx \frac{f'(x_k + hp) - f'(x_k)}{h}$$

for a suitably small parameter h. This produces an algorithm that has come to be known as the truncated Newton method [26, 72, 69].

In very large problems, the updates to the quasi-Newton matrix may prove too numerous to store. In that case we might discard updates as they age, or skip some intermediate updates. These limited memory methods were proposed by Nocedal [70], and the properties of various discarding strategies are studied in [57].

4.4 Alternatives to Newton's Method for Large Problems

There is a vast set of low-storage alternatives to Newton-like methods. They sacrifice the superlinear convergence rate that can be achieved under careful implementation of the Newton-like methods [71] in order to avoid storing a matrix approximation. Many of these methods are derived from methods for solving linear systems $Ax^* = b$ involving a symmetric positive definite matrix A.

The conjugate gradient method [46] takes a sequence of A-conjugate descent steps for the function $(x-x^*)^T A(x-x^*)$ beginning with the steepest descent direction. Many authors proposed nonlinear extensions of this method, beginning with Fletcher-Reeves [34]. The algorithms are all equivalent to quasi-Newton algorithms on quadratic functions [33, Chap. 3], but the most robust algorithm for general functions is that of Polak and Ribière [78], restarting with the steepest descent direction in case trouble is diagnosed.

Another set of methods is related to fixed-point methods for solving linear systems. These linear methods are of the form

$$x_{k+1} = Ex_k + c\,,$$

where x^* is a fixed point of the iteration and the matrix E is chosen so that its spectral radius is small, yielding linear convergence of the sequence to x^*. Often these methods are derived by some variant of solving the i-th equation for the i-th variable, and then using estimates for the other variables to form a new value for the i-th component of x. Examples of such methods are Jacobi, Gauss-Seidel, and SOR. See Varga [90] for a good discussion of such iterations in the linear case, and Ortega and Rheinboldt [74] for the general case.

5 Simplex Method for Linear Programming

In the 1940s and 1950s, several events led to an explosion of interest in computational methods. The first was the computational need generated by the participants in World War II. Data fitting (least squares) and logistics support (optimization) created enormous demands for solution to ever-larger models. At the same time, electronic computing machines for the first time made it possible to solve problems larger than those that could be handled by a roomful of human calculators.

George Dantzig was among those who realized the need for automatic algorithms for solving optimization problems, and, working for the U.S. war effort at Rand Corporation, he devised a *tableau* to organize the data in a linear programming problem

$$\min_x c^T x \tag{5}$$

$$Ax = b$$
$$x \geq 0\,,$$

where $A \in \mathcal{R}^{m\times n}$, with $m < n$. The tableau of numbers could be stored and modified systematically to produce an optimal solution to the problem, as well as information on the sensitivity of the solution to the data in A, b, and c [25]. Not only was this scheme well adapted to single or multiple human calculators, but it could also be implemented for large problems on electronic computers, enabling logistics planning that was unthinkable a few years earlier. The solution of linear programs, which earlier could in general be done only approximately by heuristic methods, could now be automated.

Dantzig's *simplex algorithm* was based on generating a path through the feasible set $\mathcal{S} = \{x \geq 0 : Ax = b\}$ through a set of *vertices* (i.e., points x for which at most m components are nonzero) that are adjacent (i.e., have all but one zero component in common). Along this path, the objective function $c^T x$ usually decreases, but in any case does not increase. Once an *anti-cycling* safeguard is added that prevents any vertex from being visited too many times, the algorithm can be proven to converge, because there are only a finite number of vertices and it can be shown that one of them is an optimal solution.

For a given vertex x, we let $\mathcal{I}$ denote the set of indices i for which x_i is nonzero. Then, if $A_\mathcal{I}$ denotes the set of columns of A corresponding to indices in $\mathcal{I}$, we see that the nonzero components $x_\mathcal{I}$ are defined by

$$A_\mathcal{I} x_\mathcal{I} = b\,. \tag{6}$$

In order to step from this vertex to an adjacent one, we replace one index in $\mathcal{I}$ by an index not in $\mathcal{I}$, and the index is determined by solving a linear system involving the matrix $A_\mathcal{I}^T$. In order to compute the x corresponding to this step, we must modify our coefficient matrix by replacing one column with a new one. Dantzig proposed accumulating the matrix inverse and updating it using elementary row operations. Equivalently, his algorithm can be viewed as Gauss-Jordan elimination without pivoting [5]. This algorithm is numerically unstable, and simplex practitioners use periodic *reinversions* to recalculate the matrix inverse directly and eliminate the accumulated inaccuracies. This is still not unconditionally stable, but for many applications it works well. For dense matrices, the initial factorization cost is $O(m^3)$ and the cost of each update is $O(m^2)$. Typically the simplex algorithm takes a reasonably small number of iterations – a small multiple of m [25, p.160] – but in the worst case the algorithm can visit every vertex [56], so the bound on the cost is exponential in the problem size.

In the last half of the century, the dimensions of linear programming problems have become much larger than first envisioned. At the same time, the matrices of interest have tended to become more *sparse*, with many zero elements. Consequently, even though the original matrix has large dimension, it usually can be stored in a small amount of memory. But the original simplex algorithm explicitly stores the matrix inverse, which is usually completely dense. Thus various modifications were made to the linear algebra of the algorithm to make it less of a storage hog. In the *revised simplex algorithm* with product form of the inverse, the inverse matrix is stored as a product of updates: a matrix $A_{\mathcal{I}}$ is represented as

$$A_{\mathcal{I}}^{-1} = R_{k-1} \dots R_1,$$

where each update matrix R_i differs from the identity by one column. This form is easily updated by accumulating additional matrices R, but when the storage for the updates becomes prohibitive, or when inaccuracies accumulate, reinversion is performed.

The computational linear algebra community became interested in the simplex method in the late 1960s. Bartels and Golub [6, 5] showed how the updating algorithm could be made stable through the use of partial pivoting, the interchange of rows of the matrix in order to bring the largest magnitude element in the current column to the main diagonal at every stage of the factorization. By computing in this way, it is possible to bound the error in the computed solution in two important ways: the computed solution solves a nearby problem, and the computed solution is close to the true solution [48, Chapter 9]. Neither of these properties is guaranteed to hold for earlier variants of the simplex algorithm. Still, the use of this stabilized algorithm met with resistance. Pivoting makes the implementation of updating much more costly, and for sparse matrices, it makes the data handling more difficult and the storage space generally higher.

The QR algorithm is an alternate matrix factorization that does not require pivoting for stability, but its fill-in often makes it prohibitively expensive for sparse matrices, so it was never widely used.

Much research in matrix reordering was spurred in part by the simplex algorithm. See [29] for more information on reordering.

Although iterative methods could be used to solve the linear systems in the simplex method, they have been proposed only for some special applications.

6 Interior Point Methods for Linear Programming

The proof by Khachian [55] that linear programming problems can be solved in polynomial time began a new era in the solution of optimization problems.

Khachian's algorithm was not practical for computation, but suddenly a great deal of attention was focused on the *interior point method* (IPM), algorithms in which the path of the iterates stays in the relative interior of the the feasible set rather than marching around the boundary from vertex to vertex. Karmarkar [52] was the first to propose a relatively practical interior point algorithm that had polynomial complexity, and that announcement spurred a flurry of work on new methods, as well as further work on older proposals such as the SUMT technique of Fiacco and McCormick [31].

The structure of IPMs is quite different from that of the simplex algorithm, but one similarity remains: the main computational work in the algorithm is the solution of a linear system of equations. Unlike the simplex algorithm, however, this linear system arises from a linear least squares problem, and this extra structure can be quite useful. Further, although the sparsity structure of the matrix in the simplex algorithm changes from iteration to iteration, the structure of the matrix in the IPM is constant, and only the weights in a diagonal matrix are changing. This fact makes data management much easier.

Consider our linear programming problem (5). Gonzaga [43] and Wright [95] surveyed interior point methods, and many computational issues are addressed by Lustig, Marsten, and Shanno [64] and Andersen, Gondzio, Mészáros, and Xu [1]. The basic idea is to replace the linear program by a nonlinear problem formed by using Lagrange multipliers y to handle the linear equality constraints, and using barrier functions to avoid violating the nonnegativity constraints. One popular barrier function is the logarithmic barrier, $\ln x_j$, which goes to $-\infty$ as $x_j \to 0^+$. The resulting Lagrange-barrier function is

$$L(x, y, \mu) = c^T x - y^T (Ax - b) - \mu \sum_{j=1}^{n} \ln x_j \,.$$

The solution to our linear programming problem (5) is the limit of the saddle-points of L as $\mu \to 0$. If we set the derivative of L equal to zero, we obtain necessary (first-order) conditions for a solution (x, y, μ) to be optimal:

$$\begin{aligned} Ax &= b\,, \\ c - A^T y - z &= 0\,, \\ XZe &= \mu e\,. \end{aligned}$$

Here, e denotes a vector of all ones, and upper case letters X and Z denote diagonal matrices created from the entries of the vectors x and z respectively. In some sense this is a relaxation of the linear program, since these are optimality conditions for the linear program if we take $z = 0$ and $\mu = 0$. The idea is to solve a sequence of problems; initially, μ is taken large in order to easily obtain a solution, and then μ is reduced.

The introduction of the variables z makes the first two equations linear, and the Lagrange multipliers y can also be interpreted as the solution to the linear programming problem that is *dual* to (5). The most successful IPMs have been those that preserve primal feasibility by keeping $Ax = b$ while at the same time maintaining dual feasibility by keeping $c - A^T y \geq 0$.

We now have a system of nonlinear equations to solve, and we can apply Newton's method. We compute the Newton direction by solving the KKT (Karush-Kuhn-Tucker) system

$$\begin{pmatrix} -X^{-1}Z & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_d + Ze - \mu X^{-1} e \\ r_p \end{pmatrix}, \tag{7}$$

or by solving the equations formed by eliminating Δx from this system. Defining $r_p = b - Ax$, $r_d = c - A^T y - z$, and $D^2 = Z^{-1}X$, we obtain the normal equations

$$(AD^2A^T)\Delta y = AD^2(r_d + Ze - \mu x^{-1} e) + r_p. \tag{8}$$

Once Δy is determined, Δx may be easily computed from

$$-(X^{-1}Z)\Delta x + A^T \Delta y = r_d + Ze - \mu X^{-1} e.$$

Solving either equation (7) or (8), then, is the central computational problem in applying IPMs to linear programming problems. The remaining considerations are what sequence of μ values to use, how accurately to solve intermediate problems, and when to terminate the algorithm or switch to direct solution once the optimal vertex is identified. For more information on these aspects, see, for example, [64, 1]. Here we concentrate on the issues involved in solving (7) or (8).

The normal equations (8) involve a symmetric positive semi-definite matrix (positive definite if A is full rank), and the Cholesky factorization is an efficient tool for solving such systems. If the matrix is sparse, though, then the Cholesky factor has the same sparsity structure as the triangular factor in the QR factorization of A, and this can be quite dense. We observe that

$$A^T DA = \sum_{j=1}^{n} a_j d_{jj} a_j^T ,$$

where a_j is a column of A. If a small number of columns are causing excessive fill-in, then these columns can be omitted from the factorization, and the Sherman-Morrison-Woodbury formula [42, Sec 2.1.3] can be used to correct for their absence [3].

Table 6 compares the features of the matrix problems in the simplex algorithm and in IPMs.

If the matrix is ill-conditioned (which often happens at the end-stage, when μ is small and some solution components go to zero) or rank-deficient (perhaps due to omission of columns in the factorization), it may be desirable to add a diagonal correction to $A^T DA$ so that a factorization of a better conditioned

Table 1: Comparison of Matrix Problems in the Simplex Algorithm and in IPMs

Simplex Method	Interior Point Methods
nonsymmetric	symmetric positive definite (normal equations) or indefinite (augmented system)
changing sparsity pattern	fixed sparsity pattern
usually well-conditioned	can become increasingly ill-conditioned as μ becomes smaller
matrix changes by rank-2 update	matrix changes completely, but only because D is changing

matrix is computed. This technique of Stewart [84] has been used by Andersen [3] and others.

The matrix of the KKT system (7) is always symmetric indefinite. We also saw this matrix in optimality conditions (3) for weighted least squares problems. Cholesky factorization is unstable for indefinite matrices, so other alternatives must be applied. The Bunch-Kaufman-Parlett factorization [14, 13] into the product LSL^T, where L is lower triangular and S is block diagonal with 1×1 or 2×2 blocks, is a convenient tool for such problems.

If A is large and sparse, then the factorizations for (7) or (8) usually include sparsity considerations in the choice of pivot order.

Iterative methods for solving the linear systems in IPMs have received a great deal of attention but rather limited success. The key problem is the choice of preconditioner. Chin and Vannelli [19] solved the KKT system (7) using an incomplete factorization as a preconditioner, while Freund and Jarre [36] proposed SSOR preconditioners. Most of the preconditioning work has been on the normal equation formulation (8). Karmarkar and Ramakrishnan [53] used the factorization of the matrix for one value of μ to precondition the problem when μ is changed. Mehrotra [65] used an incomplete Cholesky factorization as a preconditioner, recomputing the factorization for each new μ value. Carpenter and Shanno [16] used diagonal preconditioning, and Portugal, Resende, Veiga, and Júdice [79] also used spanning-tree preconditioners.

The best solution algorithm will surely be a hybrid approach that sometimes chooses direct solvers and sometimes iterative ones. Wang and O'Leary [92, 91] proposed an adaptive algorithm for determining whether to use a direct or iterative solver, whether to reinitialize or update the preconditioner, and how many updates to apply, but further work remains.

The ill-conditioning of the matrices has stimulated a lot of work in trying to understand why the algorithms work as well as they do. Saunders [81] sets forth a set of reliable solution strategies, and a stability analysis is presented in

[35].

7 Nonlinear Programming

Optimization problems with nonlinearities in their objective function or their constraints can be more difficult to solve than linear programming. We survey selected nonlinear programming problems and strategies that make use of linear algebra.

Linear algebra plays a key role in the solution of quadratic programming problems (2) and of linear complementarity problems (LCP)

$$\begin{aligned} Ax - b &= z\,, \\ x^T z &= 0\,, \\ x \geq 0\,, \qquad & z \geq 0\,. \end{aligned}$$

Approaches include variations of the simplex algorithm, extensions of linear iterations such as Jacobi and Gauss-Seidel, descent methods such as conjugate gradient, and interior point algorithms. See [66] for a comprehensive discussion. Questions of existence and uniqueness of solutions to LCP spurred work in matrix theory on matrix cones [22].

In the past, two popular methods were used to handle constraints in nonlinear programming problems [62, Chap. 11]. In the first, certain constraints were held *active* for a portion of the iteration, and the iterates were not permitted to depart from them. Any descent step was augmented by a step back to the active constraints. In the second, descent directions were projected onto equality constraints before the step was taken; thus, steps were computed relative to a *reduced gradient* that corresponded to the gradient on the constraint surface. The computations were performed by projection matrices. Both of these strategies are currently in eclipse, due to the advent of sequential quadratic programming (SQP) and interior point methods.

In SQP, we solve a sequence of quadratic programming problems (2) arising from quadratic models of the original constrained problem, using IPM or simplex-based methods for the subproblems. Again we need modifications to maintain positive definiteness. Boggs and Tolle [12] give an excellent survey of these methods.

Interior point methods are also applied to nonlinear optimization problems directly [87]. The matrix in the augmented system (7) becomes somewhat more complicated than in the linear case; the lower right block can become nonzero, the upper left block may be full instead of diagonal, and in many formulations, the matrix is increasingly ill-conditioned [95, 83]. The structure of this ill-conditioning is now somewhat understood, though, and, with care, the linear systems can be solved successfully [81].

Even these newer approaches, SQP and IPM, are not completely satisfactory, especially when the constraints are ill-behaved [21].

A rather different approach to some classes of optimization problems is the use of neural networks [30]. These networks fit a surface to a function of many variables. There are various viewpoints for interpreting such methods, but one way is that the training of the network corresponds to optimizing parameters in a function that discriminates among different classes of points. The functional form is predetermined, and the optimization problem is generally nonconvex, with many local solutions. To overcome this difficulty, Vapnik and colleagues proposed *support vector machines*, a more limited set of functional forms that are easier to analyze; see, for example, [89]. Many useful choices lead to convex optimization problems – in fact, to very large dense least squares or quadratic programming problems (2). Burges [15] provides a good introduction to the concepts and computational issues, while [88] is a more detailed study.

8 Matrix and Eigenvalue Optimization Problems

And now we come full circle. We have seen how computational linear algebra has enabled efficiency advances in computational optimization. We have seen that the optimization community has raised interesting questions about matrix theory and about stability analysis of linear algorithms. Now we discuss a set of optimization problems that arise from linear algebra and have motivated the development of important optimization algorithms and advances in the understanding of duality theory for optimization problems.

These problems involve eigenvalue optimization [60]. An important subclass is the class of *semidefinite programs*. Superficially, they resemble linear programming problems (5), since they can be written

$$\min_X C \bullet X \tag{9}$$

$$\begin{aligned} AX &= B\,, \\ X &\geq 0\,, \end{aligned}$$

but here C and X are symmetric $n \times n$ matrices, $C \bullet X = \mathrm{trace}(CX)$, and $X \geq 0$ means that X is positive semidefinite. This problem is convex but nonlinear. The duality structure for semidefinite programming, the existence and construction of another problem that has the same optimality conditions as (9), is not as satisfying as that for linear programming. Despite the differences between the two classes of problems, linear programming gives much insight here, both for the theory and for the algorithms, and interior point methods that are direct generalizations of those for linear programming are the methods of choice.

Thus, semidefinite programming problems are eigenvalue optimization problems, and these problems have important linear algebra applications in control, in minimizing the condition number of a matrix by diagonal scaling, and in solving Lyapunov inequalities. Further information can be obtained from a review article of Lewis and Overton [60], a review article of Lobo, Vandenberghe, Boyd, and Lebret describing a subclass known as second-order cone programming [61], and a collection of papers [75].

9 Computational Trends

Optimization algorithms can consume a great deal of computational resources, and they have always been run on state-of-the-art computer architectures. More and more, these algorithms are packaged and portable. There is reliable software for least squares problems on parallel computers [82], and significant work has been done with neural networks [30, §4.1] and systolic arrays [58]. But there is limited experience with parallelization of constrained optimization codes. A notable effort is the parallel version of the CPLEX code by Lustig and Rothberg [63].

A second computational trend is the development of software that performs more of the drudgery for the user. Problem generators have been widely available for many years, but new tools are also being developed. Programs for automatic differentiation, for example, have contributed to the practical application of optimization techniques to a much larger set of problems. An automatic differentiation program uses the computational definition of a function in some programming language to generate a program for evaluating the derivative of the function. There are two basic strategies, both involving repeated applications of the chain rule. The forward mode consumes a great deal of intermediate storage, while the backward mode generally takes more time. Practical implementations generally use a combination of the two strategies, guided by linear algebra tools such as sparsity structure analysis and the construction of structurally orthogonal basis vectors [7].

10 Conclusions

Major developments in the basic linear algebra of optimization algorithms in the 20th century include:

- Invention of the simplex algorithm, based on Gauss-Jordan elimination and updating.
- Learning to implement the simplex algorithm in a stable way while preserving sparsity.
- Development and understanding of Newton alternatives: truncated Newton for use when derivatives are not available, quasi-Newton for use when

second derivatives are not available, limited-memory and conjugate gradient methods for large problems.

- Development of least squares algorithms for solving dense and sparse problems in a provably stable way.
- Development of algorithms for a wider range of constrained optimization problems, including those involving eigenvalue placement.
- Making automatic differentiation practical.
- Understanding the sensitivity of linear [48] and constrained problems [25, Sec. 12.4], [32] to perturbations in the data.

In addition, the development of efficient "off-the-shelf" packages of reliable software for dense linear algebra (LAPACK) [2] and sparse linear algebra (e.g., Harwell codes [45]) makes the development of efficient and reliable optimization software much easier, and most optimization packages do make use of this linear algebra basis.

Research areas that will remain active in the next century include:

- Hybrid algorithms for solving the linear systems from IPMs and other sources, involving automatic preconditioning.
- More effective algorithms for global optimization.
- More effective algorithms for nonlinear constraints.
- Sensitivity analysis.

Much progress in linear algebra in the 20th century has been motivated, at least in part, by optimization problems. This progress includes matrix up- and down-dating, sparse direct and iterative methods for linear systems, and solution of least squares problems. Conversely, progress in optimization enables many previously intractable linear algebra problems to be solved, especially those related to eigenvalue placement. During the next century, this symbiosis will undoubtably continue. Progress in optimization will inevitably be linked with progress in linear algebra.

11 Acknowledgements

I am grateful for the hospitality provided by Professor Walter Gander and the Departement Informatik, ETH Zürich, Switzerland, which enabled this work to be completed.

References

[1] Erling D. Andersen, Jacek Gondzio, Csaba Mészáros, and Xiaojie Xu. Implementation of interior point methods for large scale linear programs. In Tamás Terlaky, editor, *Interior Point Methods of Mathematical Programming*, pages 189–252. Kluwer Academic Publishers, Boston, 1996.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 1992.

[3] Knud D. Anderson. A modified Schur-complement method for handling dense columns in interior point methods for linear programming. *ACM Trans. Math. Software*, 22:348–356, 1996.

[4] M. Arioli, J. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10:165–190, 1989.

[5] Richard H. Bartels. A stablization of the simplex method. *Numer. Math.*, 16:414–434, 1971.

[6] Richard H. Bartels and Gene H. Golub. The Simplex method of linear programming using LU decomposition. *Comm. ACM*, 12:266–268, 1969.

[7] Christian Bischof, Ali Bouaricha, Peyvand Khademi, and Jorge Moré. Computing gradients in large-scale optimization using automatic differentiation. *INFORMS J. Computing*, 9:185–194, 1997.

[8] Åke Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, Pennsylvania, 1996.

[9] Åke Björck and Tommy Elfving. Algorithms for confluent Vandermonde systems. *Numer. Math.*, 21:130–137, 1973.

[10] Åke Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13:176–190, 1992.

[11] Åke Björck and Victor Pereyra. Solution of Vandermonde system of equations. *Math. Comp.*, 24:893–903, 1970.

[12] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. In *Acta Numerica*, pages 1–51. Cambridge University Press, New York, USA, 1995.

[13] James R. Bunch, Linda Kaufman, and Beresford N. Parlett. Decomposition of a symmetric matrix. *Numer. Math.*, 27:95–109, 1976.

[14] James R. Bunch and Beresford N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 8(4):639–655, 1971.

[15] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[16] Tamra J. Carpenter and David F. Shanno. An interior point method for quadratic programs based on conjugate projected gradients. *Computational Optimization and Applications*, 2:5–28, 1993.

[17] Y. Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23:444–466, 1981.

[18] S. Chandrasekaran and I. C. F. Ipsen. On rank-revealing factorisations. *SIAM J. Matrix Anal. Appl.*, 15:592–622, 1994.

[19] P. Chin and A. Vannelli. Iterative methods for the augmented equations in large-scale linear programming. Technical Report UWE&CE-94-01, Department of Electrical and Computer Engineering, University of Waterloo, October 1994.

[20] Charles K. Chui. *An Introduction to Wavelets*. Academic Press, New York, 1992.

[21] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Methods for nonlinear constraints in optimization calculations. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 363–390. Clarendon Press, Oxford, England, 1997.

[22] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Academic Press, New York, 1992.

[23] M. G. Cox. The least squares solution of overdetermined linear equations having band or augmented band structure. *IMA J. Numer. Anal.*, 1:3–22, 1981.

[24] G. Cybenko. Fast Toeplitz orthogonalization using inner products. *SIAM J. Sci. Stat. Comput.*, 8:734–740, 1987.

[25] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[26] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[27] C. J. Demeure. Fast QR factorization of Vandermonde matrices. *Linear Algebra and Appl.*, 122/3/4:165–194, 1989.

[28] J. E. Dennis, Jr. and Jorge J. Morè. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19:46–89, 1977.

[29] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986.

[30] S. W. Ellacott. Aspects of the numerical analysis of neural networks. *Acta Numerica*, 3:145–202, 1994.

[31] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming : Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, 1968. Reprint : Volume 4 of *SIAM Classics in Applied Mathematics*, SIAM Publications, Philadelphia, PA 19104–2688, USA, 1990.

[32] Anthony V. Fiacco, ed. *Mathematical Programming with Data Perturbations*. Marcel Dekker, New York, 1998.

[33] R. Fletcher. *Practical Methods of Optimization*. John Wiley, New York, 1987.

[34] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer J.*, 7:149–154, 1964.

[35] A. Forsgren, P. E. Gill, and J. R. Shinnerl. Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization. *SIAM J. Matrix Anal. Appl.*, 17:187–211, 1996.

[36] Roland W. Freund and Florian Jarre. A QMR-based interior-point algorithm for solving linear programs. Technical report, AT&T Bell Laboratories and Institut für Angewandte Mathematik und Statistik, 1995.

[37] C. F. Gauss. Theoria combinationis observationum erroribus minimis obnoxiae, pars posterior. In *Werke, IV*, pages 27–53. Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1880.

[38] C. F. Gauss. Theoria combinationis observationum erroribus minimis obnoxiae, pars prior. In *Werke, IV*, pages 1–26. Königlichen Gesellschaft der Wissenschaften zu Göttingen, 1880.

[39] Carl Friedrich Gauss. *Theory of the Combination of Observations Least Subject to Errors. Part One, Part Two, Supplement*. SIAM, Philadelphia, Pennsylvania, 1995. Translated from the Latin by G. W. Stewart.

[40] J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Englewood Cliffs, NJ., 1981.

[41] G. Golub. Numerical methods for solving least squares problems. *Numer. Math.*, 7:206–216, 1965.

[42] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Maryland, 1996.

[43] Clovis C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34:167–224, June 1992.

[44] P. C. Hansen and H. Gesmar. Fast orthogonal decomposition of rank deficient Toeplitz matrices. *Numer. Algorithms*, 4:151–166, 1993.

[45] Harwell subroutine library. HSL Office, Culham, Oxon OX14 3ED , United Kingdom, http://www.cse.clrc.ac.uk/Activity/HSL.

[46] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Standards*, 49:409–436, 1952.

[47] Nicholas J. Higham. Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems. *Numer. Math.*, 50:613–632, 1987.

[48] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, Pennsylvania, 1996.

[49] Peter J. Huber. Robust estimation of a location parameter. *Annals of Math. Statist.*, 35:73–101, 1964.

[50] Peter J. Huber. *Robust Statistics.* John Wiley and Sons, New York, 1981.

[51] Thomas Kailath and Ali H. Sayed. Displacement structure: Theory and applications. *SIAM Review*, 37:297–386, 1995.

[52] N. K. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[53] N. K. Karmarkar and K. G. Ramakrishnan. Computational results of an interior point algorithm for large scale linear programming. *Math. Programming*, 52:555–586, 1991.

[54] R. B. Kellogg, T.-Y. Li, , and J. A. Yorke. A constructive proof of the Brouwer fixed-point theorem and computational results. *SIAM J. Numer. Anal.*, 13:473–483, 1976.

[55] L. G. Khachian. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 244:1093–1096, 1979.

[56] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

[57] Tamara Kolda, Dianne P. O'Leary, and Larry Nazareth. BFGS with update skipping and varying memory. *SIAM J. Optimization*, 8:1060–1083, 1998.

[58] H. T. Kung and C. E. Leiserson. *Algorithms for VLSI Processor Arrays.* Addison-Wesley, Reading, MA, 1980.

[59] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[60] A. S. Lewis and M. L. Overton. Eigenvalue optimization. *Acta Numerica*, 5:149–190, 1996.

[61] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and Appl.*, 284:193–228, 1998.

[62] David G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading, Massachusetts, 1984.

[63] I. J. Lustig and E. Rothberg. Gigaflops in linear programming. Technical report, Silicon Graphics, 1995.

[64] Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6(1):1–14, 1994.

[65] Sanjay Mehrotra. Implementation of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. *ORSA Journal on Computing*, 4(2):103–118, 1992.

[66] K. G. Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, Berlin, Germany, 1988.

[67] J. G. Nagy. *Toeplitz Least Squares Computations.* Ph.D. thesis, North Carolina State University, Raleigh, NC, 1991.

[68] Subhash C. Narula. Optimization techniques in linear regression: A review. *TIMS/Studies in the Management Sciences*, 19:11–29, 1982.

[69] S. G. Nash. Preconditioning of truncated-Newton methods. *SIAM J. Sci. Stat. Comput.*, 6:599–616, 1985.

[70] Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35:773–782, 1980.

[71] Jorge Nocedal. Theory of algorithms for unconstrained optimization. In *Acta Numerica*, volume 1, pages 199–242. Cambridge University Press, New York, USA, 1992.

[72] Dianne P. O'Leary. A discrete Newton algorithm for minimizing a function of many variables. *Math. Programming*, 23:20–33, 1982.

[73] Dianne P. O'Leary. Robust regression computation using iteratively reweighted least squares. *SIAM J. Matrix Anal. Appl.*, 11:466–480, 1990.

[74] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York, 1970.

[75] Michael Overton and Henry Wolkowicz. Forward: Special issue on semidefinite programming. *Math. Programming*, 77:105–110, 1997.

[76] H. Park and L. Eldén. Fast and accurate triangularization of Toeplitz matrices. Tech. Report LiTH-MAT-R-1993-17, Department of Mathematics, Linköping University, Sweden, 1993.

[77] G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.*, 13:309–316, 1970.

[78] E. Polak and G. Ribière. Note sur la convergence de methodes de directions conjugées. *Rev. Francaise Informat Recherche Operationelle, 3eAnnée*, 16:35–43, 1969.

[79] L. F. Portugal, M. G. C. Resende, G. Veiga, and J. J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. November 1994.

[80] J. K. Reid. A note on the least squares solution of a band system of linear equations by Householder reductions. *Comput J.*, 10:188–189, 1967.

[81] M. A. Saunders. Cholesky-based methods for sparse least squares: the benefits of regularization. In L. Adams and J. L. Nazareth, editors, *Linear and Nonlinear Conjugate Gradient-Related Methods*, pages 92–100. SIAM, Philadelphia, Pennsylvania, 1996.

[82] ScaLAPACK project. Oak Ridge National Laboratory, Oak Ridge, Tennessee. http://www.netlib.org/scalapack/index.html.

[83] David F. Shanno and Evangelinia M. Simantiraki. Interior point methods for linear and nonlinear programming. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 339–362. Clarendon Press, Oxford, England, 1997.

[84] G. W. Stewart. Modifying pivot elements in Gaussian elimination. *Math. Comp.*, 28(126):537–542, 1974.

[85] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Review*, 35:551–566, 1993.

[86] D. R. Sweet. Fast Toeplitz orthogonalization. *Numer. Math.*, 43:1–21, 1984.

[87] Tamás Terlaky, editor. *Interior Point Methods of Mathematical Programming.* Kluwer Academic Publishers, Boston, 1996.

[88] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York, 1995.

[89] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait. *Automation and Remote Control*, 24:6, 1963.

[90] Richard S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[91] Weichung Wang and Dianne P. O'Leary. Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming. Technical Report CS-TR-4011, Computer Science Department, University of Maryland, College Park, MD, April 1999. http://www.cs.umd.edu/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-4011.

[92] Weichung Wang and Dianne P. O'Leary. Adaptive use of iterative methods in interior point methods for linear programming. Technical Report CS-TR-3560, Computer Science Department, University of Maryland, November 1995. http://www.cs.umd.edu/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-3560.

[93] L. T. Watson, M. Sosonkina, R. C. Melville, A. P. Morgan, and H. F. Walker. Algorithm 777: Hompack90: A suite of Fortran 90 codes for globally convergent homotopy algorithms. *ACM Trans. Math. Software*, 23:514–549, 1997.

[94] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT*, 13:217–232, 1973.

[95] M. H. Wright. Interior methods for constrained optimization. In *Acta Numerica 1992*, pages 341–407. Cambridge University Press, New York, USA, 1992.

[96] Tjalling J. Ypma. Historical development of the Newton-Raphson method. *SIAM Review*, 37:531–551, 1995.