George B. Dantzig and Systems Optimization

Philip E. Gill,* Walter Murray,† Michael A. Saunders,‡
John A. Tomlin,§ and Margaret H. Wright¶

Abstract

We pay homage to George B. Dantzig by describing a less well known part of his legacy—his early and dedicated championship of the importance of systems optimization in solving complex real-world problems.

1 Introduction

George B. Dantzig, affectionately known as "GBD", was a remarkable mathematician and an equally remarkable person, famous for many things: inventing the simplex method for linear programming; foundational research on duality, complementarity theory, integer programming, quadratic programming, decomposition algorithms, stochastic programming, and methods for large specially structured linear programs; inspiring generations of students and colleagues; and shaping the entire field of optimization. Although George's lifetime achievements include a substantial body of deep theory, he derived his greatest professional satisfaction from the successful application of theory to real-world problems. In [2], he comments

> ... because my mathematics has its origin in a real problem doesn't make it less interesting to me—just the other way around, I find it makes the puzzle I am working on all the more exciting. I get satisfaction out of knowing that I'm working on a relevant problem.

The five coauthors were all privileged to work with George at the Systems Optimization Laboratory (SOL) in the Operations Research Department at Stanford University. We focus mainly on George's role in defining and supporting systems optimization; we have also included a few reminiscences that typify GBD's endearing personal qualities. Detailed chronological descriptions of GBD's career are given by Dick Cottle in [5, 6].

---

*Department of Mathematics, University of California, San Diego, La Jolla, California 92093 (pgill@ucsd.edu).

†Department of Management Science and Engineering, Stanford University, Stanford, California 94305 (walter@stanford.edu).

‡Department of Management Science and Engineering, Stanford University, Stanford, California 94305 (saunders@stanford.edu).

§Yahoo! Research, Sunnyvale, California 94089 (tomlin@yahoo-inc.com).

¶Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, New York 10012 (mhw@cs.nyu.edu).

2 GBD's perspective on systems optimization

During and after World War II, GBD worked for the United States Air Force, where one of his tasks was to develop mathematical models that could be used to formulate practical planning and scheduling problems. He describes the then-prevailing situation in [2]:

> The ground rules used in planning were expressed in a completely different format from the way we now formulate a linear program. What we did was review these rules one by one and demonstrate that almost all could be reformulated acceptably in linear programming format. . . .

> . . . I learned from Koopmans in 1947 that the economists didn't have an algorithm, and that was bad news. The generals in the Air Force were paying us to solve real planning problems. By hook or crook, we were expected to find a practical way to solve them.

His Air Force work led to the simplex method, which succeeded in solving what would be seen today as very small linear programs, such as a 77-variable diet problem. However, at the time this was an impressive accomplishment [17], requiring 120 man-days using hand-operated desk calculators!

Given the limited computing power available during the 1940s and 1950s, there was no possibility then of solving "realistic" systems-scale linear programs, meaning those with thousands of inequalities and unknowns. But by the 1960s, progress in hardware, algorithms, and software meant that some linear programming problems of this scale *could* be solved in a reasonable time on existing computers, and George's dream of a "systems optimization laboratory", inspired by the pioneering work of Wassily Leontief [2, page 303], began to take shape. Starting in the mid-1960s, GBD spoke about this ambitious concept at many conferences, arguing that there was an urgent need for environments that he described in detail in what we shall call the "SOL paper", entitled "On the need for a systems optimization laboratory".

The SOL paper in fact appeared twice: in 1973 with eight authors (but missing an "s" on "systems") [9], and in 1974, with George as the sole author, in the proceedings of a 1971 NATO conference [7]. Despite having multiple coauthors, the 1973 version contains evidence that it represents primarily George's views, such as "I cite some examples", "it is my belief", and "Philip Wolfe and myself". Hence we are comfortable in describing it as GBD's paper and GBD's vision.

GBD asserts in the SOL paper that, because complex and urgent real-world problems require total system planning, modeling, and optimization, it is necessary to create systems optimization laboratories in which there is a "critical mass" of people and resources so that:

1. Representative problems can be modeled mathematically;

2. General-purpose optimization methods can be devised;

3. Software implementing these methods can be written and systematically tested on representative problems;

4. Insights can be obtained into the nature of the problems and the properties of the general methods;

5. Based on these insights, methods can be developed to take advantage of the special structure of the most interesting and important problems.

An essential part of GBD's perspective was that the fruits of all these activities should be freely available to the wider community.

As justification for systems optimization laboratories, GBD pointed out that many complex problems can be solved only if they are viewed as total systems. His list of examples includes investment planning over time; engineering design and optimization; physical, biological, and ecological systems; urban planning; and transportation systems. GBD presented an ambitious and prophetic concept ahead of its time, recognizing (for what we believe to have been the first time) many of the organizational issues that needed to be solved to make significant progress toward modeling and optimizing complex systems.

3 The early days of the Stanford SOL

In addition to conceiving a grand vision, GBD was determined to found an actual systems optimization laboratory within the Operations Research (OR) Department at Stanford. Several OR faculty, including Dick Cottle, Fred Hillier, and Alan Manne, were supportive of such an activity, but the department did not wish to appoint multiple regular faculty with very similar research interests in numerical optimization and software. Hence George's only hope was to find external funding for full-time researchers.

Unfortunately for the nascent Stanford SOL, in those days United States government agencies were reluctant to fund software development, which was not considered to be fundamental research. GBD nonetheless continued in his low-key but persistent way to seek and obtain funding to establish an SOL. The authors of this paper believe that it is now safe to reveal that he managed to do so by bootstrapping grants in optimization that emphasized mathematical theory without mentioning any of the software-related activities that he planned to include.

The funding that GBD was able to raise covered scientific personnel, but certainly not lavish accommodations or equipment. John Tomlin was the first person to work full-time explicitly at the Stanford SOL. When he arrived in 1970 as an IBM post-doctoral fellow, the largest room in the OR Department was impressively labeled "Model Optimization Laboratory". However, the room was almost empty, containing only a drafting table, a large cabinet with many wide, short drawers, and an IBM "golfball" computer terminal—the sole connection between the OR Department and the computer center across campus. When John returned to Stanford in 1972 as the Assistant Director of SOL, the room had been relabeled as the "Systems Optimization Laboratory", but there was a large gap between his imposing title and the unchanged reality of an organization that still possessed a drafting table (whose purpose was never clarified), but no official funding and very little structure.

Despite the sparse environment and uncertain financial support, admirable results were produced almost immediately. In particular, John used the golfball terminal to write the initial versions of his linear programming "LPMx" codes, intended to test basis updating methods [35]. Ph.D. students with practical inclinations also became involved with SOL activities, although they were warned by George that they needed to "math it up" to pass muster with the primarily theoretical OR Department.

The next step toward creation of the critical mass envisaged by GBD was the arrival in

1975 of Michael Saunders, whose 1972 Ph.D. thesis in the Computer Science Department at Stanford treated linear algebraic issues in the simplex method (with GBD as a member of his thesis committee). Michael stayed at SOL for two years, then went back to New Zealand for two years, and finally returned to Stanford in 1979. Margaret Wright arrived in 1976, another Ph.D. from Stanford's Computer Science Department; John Tomlin departed in 1977; and Philip Gill and Walter Murray joined SOL in 1979 from the National Physical Laboratory in Teddington, England.

We note for completeness that, in addition to GBD, Martin Beale (Imperial College), Gene Golub (Stanford), and Jim Wilkinson (National Physical Laboratory) influenced the early SOL work on numerical software by the present authors.

Many changes have taken place at Stanford since those early years. Gerd Infanger's arrival in 1989 was an especially important event for GBD, who collaborated closely with Gerd on stochastic programming until the late 1990s. (See [29] for information about today's Stanford SOL.)

4 GBD and optimization software

GBD refers to himself in [2] as "expert at programming planning methods using the only 'computing machines' we had then ... hand-operated desk calculators" during 1941–1945; John Tomlin recalls that, at their first meeting in 1969, George wore a button saying "My computer understands me". GBD appreciated very early the impact and potential of computers as tools to solve real-world problems. (For this reason, he had a joint appointment in Stanford's Computer Science Department.) In happy (for us) contrast to some of his colleagues who regarded the design and writing of software as trivial or uninteresting, he dedicated vast amounts of his time and energy to generating support for, nurturing, and protecting software-related activities.

The problems that GBD loved the most, without question, were linear programs, preferably stochastic with staircase structure. In the hallway outside his office, he had posted an imitation Charles Schulz cartoon depicting the "Peanuts" character Linus sucking his thumb and holding his blanket, with the caption "Happiness is assuming the world is linear". Hence it is no surprise that the first software produced at the Stanford SOL (by John Tomlin) was the linear programming code LPM1 [35], making George's vision of freely available SOL software a reality.

Despite GBD's personal focus on linear programming, he was entirely supportive of the Stanford SOL's expansion into nonlinearity. In the mid-1970s, Michael Saunders, together with Bruce Murtagh, produced the first version of the code MINOS (Modular In-core Nonlinear Optimization System). Originally designed for problems with a nonlinear objective function and linear constraints [25], this code has morphed into today's all-nonlinear MINOS [26], and is probably the software most associated with the Stanford SOL.

Other software developed at the Stanford SOL during its first decade includes LCPL for linear complementarity problems [37, 38]; QPSOL for nonconvex quadratic programming; LSSOL for linearly constrained linear least-squares problems; and NPSOL for nonlinearly constrained optimization [14].

4

5 An historical aside: GBD, the Stanford SOL, Khachiyan, and Karmarkar

Throughout his career, George described his invention of the simplex method with characteristic modesty [8]:

> I thought the method might be efficient [in the sense that it required a small number of steps relative to the exponential number of combinations] but not practical, so I continued to look for a better algorithm. About a year later ...my group asked me why I continued to look elsewhere when the simplex algorithm was working out so well on the test problems. ... I certainly did not anticipate that it would turn out to be so terrific.

Never one to rest on his laurels, GBD keenly followed theoretical and practical advances in linear programming, especially the quest to discover a polynomial-time method. He mused on various occasions about the contrived nature of the famous Klee-Minty example of exponential-time performance by the simplex method using the "textbook" pivot rule [20], and he wondered about the existence of a polynomial-time simplex pivot rule (a question still open in 2007) .

In 1979, Leonid Khachiyan astounded the mathematical programming community by showing [19] that linear programs could be solved in polynomial time using the ellipsoid algorithm, which had been developed in 1976–77 for nonlinear convex programs independently by Shor and by Yudin and Nemirovski. Khachiyan's breakthrough ran completely contrary to the long-dominant worldview that, because of the power of the simplex method, it was undesirable to "nonlinearize" linear programs by applying techniques from nonlinear optimization.

GBD was delighted with Khachiyan's result, especially because the proof techniques (verifying that the optimal solution lies within a sequence of shrinking ellipsoids) were radically different from those traditionally associated with the simplex method. GBD's respect for Khachiyan led him to ask that a photograph be taken of the two of them together at a 1990 Asilomar workshop. (By a sad coincidence, Khachiyan died at the age of 52 on April 29, 2005, two weeks before GBD. An appreciation of Khachiyan's work may be found in [34].)

Stanford SOL researchers were among the earliest to investigate the performance of the ellipsoid algorithm applied to linear programs, and (as is now well known) the method turned out to be extremely slow in practice. GBD was first and foremost a scientist, but he was, not surprisingly, pleased that the simplex method retained its pre-eminence as an LP solution technique. He was also fascinated, as were many, by the enormous gap between practice (the typically rapid speed of the simplex method) and achievable worst-case complexity. This dichotomy, which is not unique to the simplex method, continues to lead to new results. A centerpiece of recent work by Spielman and Teng [30, 31] is a proof that the simplex method has polynomial "smoothed complexity".

Five years after Khachiyan's result, Stanford SOL researchers played an interesting role during a tumultuous period in linear programming. In 1984 Narendra Karmarkar, a mathematician at AT&T Bell Labs, announced his discovery of an algorithm for linear programming that was both polynomial-time and consistently 50 times faster than the simplex method. Even more than Khachiyan, Karmarkar was the subject of wide publicity in the

mainstream press, including the *New York Times* [16]. Karmarkar's proof of polynomial complexity, accompanied by a mathematical description of the algorithm, appeared at the end of 1984 [18]. However, because the software implementing his method was proprietary, researchers outside AT&T were unable to perform comparative numerical tests as they had with Khachiyan's method.

GBD, who was unfailingly gracious and warm to young researchers, was delighted to learn about the latest development in linear programming. He personally congratulated Karmarkar for his accomplishment and invited him to visit Stanford. During Karmarkar's Stanford talk, the SOL researchers, trained in nonlinear optimization in general and barrier methods in particular [23, 24], observed the strong similarity between the equations in Karmarkar's method and those arising in the 1960s logarithmic barrier method of Fiacco and McCormick [12].

By the summer of 1985, the Stanford SOL "gang of four"—Gill, Murray, Saunders, Wright—and their former colleague John Tomlin, with help from Irvin Lustig (then a Stanford Ph.D. student working with GBD), had proved that there was a formal mathematical connection between Karmarkar's method and the log barrier method. In addition, a "projected Newton barrier" code had been written and compared with the simplex method (as implemented in MINOS) on a suite of representative LPs (see Section 6). To the surprise of many (including the authors of this paper), the nonlinear barrier method was obviously competitive with the simplex method, providing the first confirmation outside AT&T of the promise of barrier-based methods for LP. (For details, see [15].)

The so-called "interior-point revolution" in linear, quadratic, convex, and nonlinear optimization began with Karmarkar's seminal work. It also seems fair to say that, without George's guiding influence, commitment, and support, the central role of barrier methods (see [22]) would have taken much longer to establish.

Unfortunately, for reasons that seem in retrospect incomprehensible, the first few years after Karmarkar's announcement were marred by inaccurate and unwarranted comments about the simplex method. For example, a *Time* magazine article in late 1984 [32] asserted that "Before the Karmarkar method, linear equations [sic] could be solved only in a cumbersome fashion, ironically known as the simplex method, devised by mathematician George Dantzig in 1947"; a speaker at the 1988 International Symposium on Mathematical Programming ended his talk on an interior-point method by saying "The simplex method is still running". George's equanimity and sweetness of temperament were never more in evidence than during this time.

Because of limited space, this paper cannot begin to summarize the long-term effects of interior methods on optimization, except to note that linear, nonlinear, and integer programming have all changed in significant ways since 1984. Mike Todd's survey [33] gives a nice overview of linear programming from several perspectives. Readers seeking a picture of the recent state of the art in LP software should consult Bob Bixby's 2002 study [3], which carefully analyzes algorithmic features of simplex-based and barrier methods as well as the effects of computer hardware. Several textbooks ([28, 39, 41], to name three) discuss the history of interior-point methods as well as issues of complexity. In our view, a central and welcome change has been elimination of the formerly widespread article of faith that linear and nonlinear programming are completely different. (This aspect of LP history is described in, for example, the opening sections of [40].)

6   Collecting representative test problems

From the beginning, a Stanford SOL activity initiated by GBD was the collection of test problems, primarily LP problems. These came from a variety of sources, including industrial contacts and large-scale energy-economic models. In particular, George and colleagues (including, during 1975–1990, Pat McAllister, Shail Parikh, John Stone, and others) had developed PILOT, a multi-period investment and production model of the United States economy. George was especially fond of the associated linear programs not only because they display the staircase structure that never failed to engross him but also because the PILOT model is, in his words, "the real McCoy—a powerful tool for making policy decisions" [2].

Another step toward the systematic collection of representative problems occurred at the 1973 International Mathematical Programming Symposium, held at Stanford, when SOL hosted a meeting of researchers (including Harlan Crowder, Larry Haverly, Rick Jackson, and Mike Powell) who were interested in developing standards for testing optimization algorithms (see [36]). The resulting discussions culminated in the formation of the Mathematical Programming Society's Committee on Algorithms (COAL).

As noted in Section 5, the frenzy surrounding Karmarkar's method led many researchers to the view, propounded by GBD in the SOL paper [9], that a standard collection of linear programming test problems was essential to enable experimental comparisons of the simplex method and the flood of newly proposed interior-point methods. Email was in wide use by 1984, and David Gay of Bell Labs began a concerted effort to gather a suite of interesting LP problems [13]. The resulting collection of LP problems, called the "netlib test set" after the *netlib* system for electronic software distribution [11], has grown substantially since 1984 and remains the standard benchmark for testing LP methods. Several of the first *netlib* problems were collected by the Stanford SOL and we list them for LP aficionados: `25fv47`, `adlittle`, `afiro`, `bandm`, `beaconfd`, `brandy`, `capri`, `czprob`, `e226`, `etamacro` (based on Alan Manne's nonlinear energy model), `israel`, `pilot` (in several variations), `share1b`, `share2b`, `shell`, and `stair`. In addition, long before the availability of tools for doing so, Irv Lustig analyzed and created graphical representations of the matrix structures in the first 53 *netlib* problems [21].

In the immediate post-Karmarkar era, leading researchers began to report computational results on the same set of problems, and consequently it became known which problems were "easy" and which were "hard" for which methods. A nice example of GBD's goal of improving methods by learning from numerical results occurred when early interior-point methods were observed to be noticeably slower than the simplex method on the *netlib* problem `israel`. The difficulty was caused by fill-in from a few dense columns in the constraint matrix (see, e.g., Figures 25 and 26 in [1]), and linear algebraic techniques were soon developed to address this situation.

Today, researchers in LP have access to a large array of test problems, some huge (with tens of millions of variables), along with analysis of characteristics that influence the performance of algorithmic options; see [3].

7   Today

Many changes have transformed the landscape of optimization methods and software since GBD first proposed establishment of a systems optimization laboratory. Thanks to the

Internet and the World Wide Web, it is no longer necessary for the critical mass of people to be co-located, since researchers and users can exchange code electronically as well as run problems on a machine in a remote location using software written by someone else. Even so, GBD's concept of a systems optimization laboratory lives on; there are at least twenty organizations in universities around the world (Canada, France, Israel, Japan, Korea, and the United States) whose names are small variants of "systems optimization laboratory". Not all organizational problems have been resolved, however; obtaining sustained government funding for software development remains a challenge.

We mention two complementary resources for today's optimization researchers, software developers, and users, both reflecting GBD's philosophy of providing publicly available access to the latest and best.

- The NEOS (Network-Enabled Optimization Server) [27] fulfills several elements in GBD's "wish list" for a systems optimization laboratory. NEOS enables optimization software to be invoked electronically by those with problems to solve. Users can submit problems to NEOS, expressed in a selection of modeling languages, and receive the computed solutions within moments. The structure of NEOS permits authors of proprietary software to make their codes available as solvers without providing access to the source, so that commercial software can be used on a trial basis.

- The COIN-OR (Computational Infrastructure for Operations Research) project, launched in 2000 by IBM Research and managed today by a foundation [4], aims "to create for mathematical software what the open literature is for mathematical theory". A key difference from NEOS is that COIN-OR uses an "open source" model in which researchers submit their own source code and have access to the source code of others. The structure of COIN-OR also makes it possible for software to be peer-reviewed.

Turning now to the "big picture" that motivated GBD during his entire career, despite tremendous progress the problems of modeling and optimizing complex systems are not yet solved. In fact, a 2004 report on multiscale mathematics by the United States Department of Energy includes a sentence that could have been written by GBD in the 1970s: "Unfortunately, the ability to simulate complete systems does not follow immediately or easily from an understanding, however comprehensive, of the component parts. For that, we need to know and to faithfully model how the system is connected and controlled at all levels" [10].

8   GBD's legacy

In our view, GBD deserves enormous credit—which, in his unassuming way, he would never have claimed for himself—for his pioneering and enduring contributions to modeling and optimizing complex systems. George was an inspiration to people in his field as well as to people who never met him; to his amusement, his famous "two unsolved problems" homework incident once served as the theme of a radio sermon about positive thinking given by a minister he met on an airplane [2]. GBD was quiet and gentle, but utterly unforgettable. One can almost hear his soft voice and see his eyes sparkling with enthusiasm as he says [8]:

> We have come a long way ... but much work remains to be done. The final test will come when we can solve the practical problems which originated the field back in 1947.

It is impossible to convey the extent to which George is and will be missed.

Acknowledgements

References

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga (1989), Data structures and programming techniques for the implementation of Karmarkar's algorithm, *ORSA Journal on Computing* 1, 84–106.

[2] D. J. Albers and C. Reid (1986), An interview with George B. Dantzig: the father of linear programming, *The College Mathematics Journal* 17, 292–314.

[3] R. E. Bixby (2002), Solving real-world linear programs: a decade and more of progress, *Operations Research* 50, 3–15.

[4] `www.coin-or.org`.

[5] R. W. Cottle (2005), George B. Dantzig: operations research icon, *Operations Research* 53, 892–898.

[6] R. W. Cottle (2006), George B. Dantzig: A legendary life in mathematical programming, *Mathematical Programming Series A* 105, 1–8.

[7] G. B. Dantzig (1974), "On the need for a systems optimization laboratory", in *Optimization Methods for Resource Allocation* (Proceedings of a 1971 NATO conference), R. W. Cottle and J. Krarup (eds.), English Universities Press, London.

[8] G. B. Dantzig (1991), "Linear programming", in *History of Mathematical Programming: A Collection of Personal Reminiscences*, J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver (eds.), Elsevier Science Publishers, Amsterdam.

[9] G. B. Dantzig, R. W. Cottle, B. C. Eaves, F. S. Hillier, A. S. Manne, G. H. Golub, D. J. Wilde, and R. B. Wilson (1973), "On the need for a system optimization laboratory", in *Mathematical Programming*, T. C. Hu and S. M. Robinson (eds.), Academic Press, New York and London.

[10] J. Dolbow, M. A. Khaleel, and J. Mitchell (2004), *Multiscale Mathematics Initiative: A Roadmap, Report from the Third DOE Workshop on Multiscale Mathematics*, `www.sc.doe.gov/ascr/mics/amr/Multiscale`.

[11] J. Dongarra and E. Grosse (1987), Distribution of mathematical software via electronic mail, *Communications of the ACM* 30, 403–407.

[12] A. V. Fiacco and G. P. McCormick (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York. Republished by Society for Industrial and Applied Mathematics, Philadelphia, 1990.

[13] D. M. Gay (1985), Electronic mail distribution of linear programming test problems, *COAL Newsletter* 13, 10–12.

[14] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright (1998), User's guide for NPSOL 5.0: a Fortran package for nonlinear programming, Report NA 98-2, Department of Mathematics, University of California, San Diego.

[15] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright (1986), On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, *Mathematical Programming* 36, 183–209.

[16] J. Gleick (1984), "Breakthrough in problem solving", *New York Times*, November 18, 1984 (page 1).

[17] `www.ifip.org/IFIP_MEMORIES/DANTZIG.htm`.

[18] N. K. Karmarkar (1984), A new polynomial time algorithm for linear programming, *Combinatorica* 4, 373–395.

[19] L. G. Khachiyan (1979), A polynomial algorithm in linear programming (in Russian), *Doklady Akedamii Nauk SSR* 244, 1093–1096.

[20] V. Klee and G. J. Minty (1972), How good is the simplex method?, in *Inequalities III*, O. Shisha (ed.), Academic Press, London and New York, 159–175.

[21] I. J. Lustig (1989), An analysis of an available set of linear programming test problems, *Computers and Operations Research* 16, 173–184.

[22] R. Marsten, R. Subramanian, M. Saltzman, I. Lustig, and D. Shanno (1990), Interior point methods for linear programming: just call Newton, Lagrange, and Fiacco and McCormick!, *Interfaces* 20, 105–116.

[23] W. Murray and M. H. Wright (1976a), Efficient linear search algorithms for the logarithmic barrier function, Technical Report SOL 76-18, Systems Optimization Laboratory, Department of Operations Research, Stanford University. Also published as Murray, W. and Wright, M. H. (1994), Line search procedures for the logarithmic barrier function, *SIAM Journal on Optimization* 4, 229–246.

[24] W. Murray and M. H. Wright (1976b), Numerical aspects of trajectory methods for nonlinearly constrained optimization, in *Proceedings of the Bicentennial Conference on Mathematical Programming*, Gaithersburg, Maryland, 1976.

[25] B. A. Murtagh and M. A. Saunders (1978), Large-scale linearly constrained optimization, *Mathematical Programming* 14, 41–72.

[26] B. A. Murtagh and M. A. Saunders (1982), A projected Lagrangian method and its implementation for sparse nonlinear constraints, *Mathematical Programming Study* 16, 84–117.

[27] `neos.mcs.anl.gov`.

[28] C. Roos, T. Terlaky, and J.-Ph. Vial (1997), *Theory and algorithms for linear optimization: an interior point approach*, John Wiley and Sons, New York.

[29] `www.stanford.edu/group/SOL`.

[30] D. A. Spielman and S.-H. Teng (2004), Why the simplex method usually takes polynomial time, *Journal of the ACM* 51, 385–463.

[31] `www.cs.yale.edu/homes/spielman/SmoothedAnalysis/index.html`.

[32] "Folding the perfect corner", *Time*, December 3, 1984.

[33] M. J. Todd (2002), The many facets of linear programming, *Mathematical Programming* 91, 417–436.

[34] M. J. Todd (2005), Leonid Khachiyan, 1952–2005: An appreciation, *SIAM News*, December 1, 2005. `www.siam.org/news/news.php?id=197`.

[35] J. A. Tomlin (1972), Maintaining a sparse inverse for the simplex method, *IBM J. of Research and Development* 16, 415–423.

[36] J. A. Tomlin (1973), Computational standards for the mathematical programming society, *ACM SIGMAP Newsletter* 15, 22–24.

[37] J. A. Tomlin (1976), User's guide to LCPL—a program for solving linear complementarity problems using Lemke's method, Technical Report 76–16, Systems Optimization Laboratory, Department of Operations Research, Stanford University.

[38] J. A. Tomlin (1978), Robust implementation of Lemke's method for the linear complementarity problem, *Mathematical Programming Studies* 7, 55–60.

[39] R. J. Vanderbei (2001), *Linear programming: foundations and extensions* (second edition), Springer, New York.

[40] M. H. Wright (2005), The interior-point revolution in optimization: history, recent developments, and lasting consequences, *Bulletin of the American Mathematical Society* 42, 39–56.

[41] Y. Ye (1997), *Interior-point algorithms: theory and analysis*, John Wiley and Sons, New York.