from *SIAM News*, Volume 33, Number 4

The Best of the 20th Century: Editors Name Top 10 Algorithms

By Barry A. Cipra

Algos is the Greek word for pain. *Algor* is Latin, to be cold. Neither is the root for *algorithm*, which stems instead from al-Khwarizmi, the name of the ninth-century Arab scholar whose book *al-jabr wa'l muqabalah* devolved into today's high school algebra textbooks. Al-Khwarizmi stressed the importance of methodical procedures for solving problems. Were he around today, he'd no doubt be impressed by the advances in his eponymous approach.

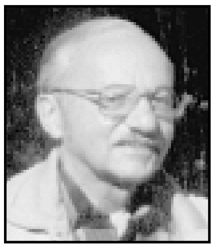
Some of the very best algorithms of the computer age are highlighted in the January/February 2000 issue of *Computing in Science & Engineering*, a joint publication of the American Institute of Physics and the IEEE Computer Society. Guest editors Jack Don-garra of the University of Tennessee and Oak Ridge National Laboratory and Fran-cis Sullivan of the Center for Comput-ing Sciences at the Institute for Defense Analyses put togeth-er a list they call the "Top Ten Algorithms of the Century."

"We tried to assemble the 10 al-gorithms with the greatest influence on the development and practice of science and engineering in the 20th century," Dongarra and Sullivan write. As with any top-10 list, their selections—and non-selections—are bound to be controversial, they acknowledge. When it comes to picking the algorithmic best, there seems to be no best algorithm.

Without further ado, here's the CiSE top-10 list, in chronological order. (Dates and names associated with the algorithms should be read as first-order approximations. Most algorithms take shape over time, with many contributors.)

1946: John von Neumann, Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the Metropolis algorithm, also known as the **Monte Carlo method**.

The Metropolis algorithm aims to obtain approximate solutions to numerical problems with unmanageably many degrees of freedom and to combinatorial problems of factorial size, by mimicking a random process. Given the digital computer's reputation for deterministic calculation, it's fitting that one of its earliest applications was the generation of random numbers.


In terms of widespread use, George Dantzig's simplex method is among the most successful algorithms of all time.

1947: George Dantzig, at the RAND Corporation, creates the **simplex method for linear programming**.

In terms of widespread application, Dantzig's algorithm is one of the most successful of all time: Linear programming dominates the world of industry, where economic survival depends on the ability to optimize within budgetary and other constraints. (Of course, the "real" problems of industry are often nonlinear; the use of linear programming is sometimes dictated by the computational budget.) The simplex method is an elegant way of arriving at optimal answers. Although theoretically susceptible to exponential delays, the algorithm in practice is highly efficient—which in itself says something interesting about the nature of computation.

1950: Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of **Krylov subspace iteration methods**.

These algorithms address the seemingly simple task of solving equations of the form $Ax = b$. The catch, of course, is that A is a huge $n \times n$ matrix, so that the algebraic answer $x = b/A$ is not so easy to compute. (Indeed, matrix "division" is not a particularly useful concept.) Iterative methods—such as solving equations of the form $Kx_{i+1} = Kx_i + b - Ax_i$ with a simpler matrix K that's ideally "close" to A—lead to the study of Krylov subspaces. Named for the Russian mathematician Nikolai Krylov, Krylov subspaces are spanned by powers of a matrix applied to an initial "remainder" vector $r_0 = b - Ax_0$. Lanczos found a nifty way to generate an orthogonal basis for such a subspace when the matrix is symmetric. Hestenes and Stiefel proposed an even niftier method, known as the conjugate gradient method, for systems that are both symmetric and positive definite. Over the last 50 years, numerous researchers have improved and extended these algorithms. The current suite includes techniques for non-symmetric systems, with acronyms like GMRES and Bi-CGSTAB. (GMRES and Bi-CGSTAB premiered in *SIAM Journal on Scientific and Statistical Computing*, in 1986 and 1992, respectively.)

Alston Householder

1951: Alston Householder of Oak Ridge National Laboratory formalizes the **decompositional approach to matrix computations**.

The ability to factor matrices into triangular, diagonal, orthogonal, and other special forms has turned out to be extremely useful. The decompositional approach has enabled software developers to produce flexible and efficient matrix packages. It also facilitates the analysis of rounding errors, one of the big bugbears of numerical linear algebra. (In 1961, James Wilkinson of the National Physical Laboratory in London published a seminal paper in the *Journal of the ACM*, titled "Error Analysis of Direct Methods of Matrix Inversion," based on the LU decomposition of a matrix as a product of lower and upper triangular factors.)

1957: John Backus leads a team at IBM in developing the **Fortran optimizing compiler**.

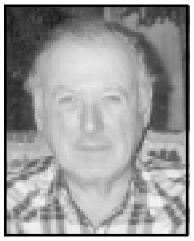
The creation of Fortran may rank as the single most important event in the history of computer programming: Finally, scientists (and others) could tell the computer what they wanted it to do, without having to descend into the netherworld of machine code. Although modest by modern compiler standards—Fortran I consisted of a mere 23,500 assembly-language instructions—the early compiler was nonetheless capable of surprisingly sophisticated computations. As Backus himself recalls in a recent history of Fortran I, II, and III, published in 1998 in the *IEEE Annals of the History of Computing*, the compiler "produced code of such efficiency that its output would startle the programmers who studied it."

1959–61: J.G.F. Francis of Ferranti Ltd., London, finds a stable method for computing eigenvalues, known as the **QR algorithm**.

Eigenvalues are arguably the most important numbers associated with matrices—and they can be the trickiest to compute. It's relatively easy to transform a square matrix into a matrix that's "almost" upper triangular, meaning one with a single extra set of nonzero entries just below the main diagonal. But chipping away those final nonzeros, without launching an avalanche of error, is nontrivial. The QR algorithm is just the ticket. Based on the QR decomposition, which writes A as the product of an orthogonal matrix Q and an upper triangular matrix R, this approach iteratively changes $A_i = QR$ into $A_{i+1} = RQ$, with a few bells and whistles for accelerating convergence to upper triangular form. By the mid-1960s, the QR algorithm had turned once-formidable eigenvalue problems into routine calculations.

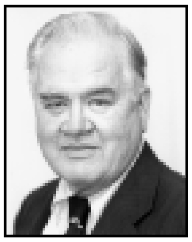
1962: Tony Hoare of Elliott Brothers, Ltd., London, presents **Quicksort**.

Putting N things in numerical or alphabetical order is mind-numbingly mundane. The intellectual challenge lies in devising ways of doing so quickly. Hoare's algorithm uses the age-old recursive strategy of divide and conquer to solve the problem: Pick one element as a "pivot," separate the rest into piles of "big" and "small" elements (as compared with the pivot), and then repeat this procedure on each pile. Although it's possible to get stuck doing all $N(N-1)/2$ comparisons (especially if you use as your pivot the first item on a list that's already sorted!), Quicksort runs on average with $O(N \log N)$ efficiency. Its elegant simplicity has made Quicksort the pos-terchild of computational complexity.


James Cooley

1965: James Cooley of the IBM T.J. Watson Research Center and John Tukey of Princeton University and AT&T Bell Laboratories unveil the **fast Fourier transform**.

Easily the most far-reaching algo-rithm in applied mathematics, the FFT revolutionized signal processing. The underlying idea goes back to Gauss (who needed to calculate orbits of asteroids), but it was the Cooley–Tukey paper that made it clear how easily Fourier transforms can be computed. Like Quicksort, the FFT relies on a divide-and-conquer strategy to reduce an ostensibly $O(N^2)$ chore to an $O(N \log N)$ frolic. But unlike Quick- sort, the implementation is (at first sight) nonintuitive and less than straightforward. This in itself gave computer science an impetus to investigate the inherent complexity of computational problems and algorithms.


John Tukey

1977: Helaman Ferguson and Rodney Forcade of Brigham Young University advance an **integer relation detection algorithm**.

The problem is an old one: Given a bunch of real numbers, say $x_1, x_2, \ldots, x_n$, are there integers $a_1, a_2, \ldots, a_n$ (not all 0) for which $a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$? For $n = 2$, the venerable Euclidean algorithm does the job, computing terms in the continued-fraction expansion of x_1/x_2. If x_1/x_2 is rational, the expansion terminates and, with proper unraveling, gives the "smallest" integers a_1 and a_2. If the Euclidean algorithm doesn't terminate—or if you simply get tired of computing it—then the unraveling procedure at least provides lower bounds on the size of the smallest integer relation. Ferguson and Forcade's generalization, although much more difficult to implement (and to understand), is also more powerful. Their detection algorithm, for example, has been used to find the precise coefficients of the polynomials satisfied by the third and fourth bifurcation points, $B_3 = 3.544090$ and $B_4 = 3.564407$, of the logistic map. (The latter polynomial is of degree 120; its largest coefficient is 257^{30}.) It has also proved useful in simplifying calculations with Feynman diagrams in quantum field theory.

1987: Leslie Greengard and Vladimir Rokhlin of Yale University invent the **fast multipole algorithm**.

This algorithm overcomes one of the biggest headaches of N-body simulations: the fact that accurate calculations of the motions of N particles interacting via gravitational or electrostatic forces (think stars in a galaxy, or atoms in a protein) would seem to require $O(N^2)$ computations—one for each pair of particles. The fast multipole algorithm gets by with $O(N)$ computations. It does so by using multipole expansions (net charge or mass, dipole moment, quadrupole, and so forth) to approximate the effects of a distant group of particles on a local group. A hierarchical decomposition of space is used to define ever-larger groups as distances increase. One of the distinct advantages of the fast multipole algorithm is that it comes equipped with rigorous error estimates, a feature that many methods lack.

What new insights and algorithms will the 21st century bring? The complete answer obviously won't be known for another hundred years. One thing seems certain, however. As Sullivan writes in the introduction to the top-10 list, "The new century is not going to be very restful for us, but it is not going to be dull either!"

Barry A. Cipra is a mathematician and writer based in Northfield, Minnesota.

"An intuitive algebraic approach for solving Linear Programming problems"

Source: Zionts [1974] (or many others).

$$\begin{array}{rrrll} [\max] z = & 0{,}56x_1 & +0{,}42x_2 & & \\ \text{s. to} & x_1 & +2x_2 & \leq & 240 \\ & 1{,}5x_1 & +x_2 & \leq & 180 \\ & x_1 & & \leq & 110 \end{array} \qquad \{1\}$$

$\boxed{\textbf{A}}$[1]

$$\begin{array}{rrrllll} [\max] z = & 0{,}56x_1 & +0{,}42x_2 & & & & \\ & x_1 & +2x_2 & +\{x_3\} & & = & 240 \\ & 1{,}5x_1 & +x_2 & & +\{x_4\} & = & 180 \\ & x_1 & & & +\{x_5\} & = & 110 \end{array} \qquad \{2\}$$

This has (always) an obvious, sure solution. Let

$$x_1, x_2 = 0 \qquad \{3\}$$

Then

$$\begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 240 \\ 180 \\ 110 \end{bmatrix} \qquad \{4\}$$

$$z = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 240 \\ 180 \\ 110 \end{bmatrix} = 0 \qquad \{5\}$$

Is this optimal ? How to improve ?

There does not appear (Dantzig) to be a systematic way of setting *all* the nonbasic variables simultaneously to optimal values —hence, an *iterative*[2] method.

Choose the variable that increases the objective function *most* per unit (this choice is arbitrary), in the example, x_1, because its coefficient (0,56) is the largest.

According to the constraints, x_1 can be increased till:

$\boxed{\textbf{B}}$

$$\begin{array}{rl} x_1 & = 240 \\ 1{,}5x_1 & = 180 \\ x_1 & = 110 \end{array} \quad \rightarrow \quad \begin{array}{rl} x_1 & = 240 \\ x_1 & = 120 \\ x_1 & = 110 \end{array} \qquad \{6\}$$

The *third* equation (why?) in {2} leads to $x_1 = 110$ and $x_5 = 0$. The variable x_1 will be the *entering* variable and x_5 the *leaving* variable:

[1] A, B, C identify the iteration, as summarized below.

[2] *Iterative:* involving repetition; relating to *iteration*. *Iterate* (from Latin *iterare*), to say or do again (and again). Not to be confused with *interactive*.

$$\boxed{\mathbf{C}} \qquad x_1 = 110 - x_5 \qquad \{7\}$$

Substituting for x_1 everywhere (except in its own constraint), we have

$$\begin{array}{lllllll} [\max]z = & 0{,}56(110 - x_5) & +0{,}42x_2 & & & & \\ & (110 - x_5) & +2x_2 & +x_3 & & = & 240 \\ & 1{,}5(110 - x_5) & +x_2 & & +x_4 & = & 180 \\ & x_1 & & & +x_5 & = & 110 \end{array} \qquad \{8\}$$

$$\boxed{\mathbf{A}} \qquad \begin{array}{llllllll} [\max]z = & & 0{,}42x_2 & & & -0{,}56x_5 & +61{,}6 & \\ & & +2x_2 & +\{x_3\} & & -x_5 & = & 130 \\ & & x_2 & & +\{x_4\} & -1{,}5x_5 & = & 15 \\ & \{x_1\} & & & & +x_5 & = & 110 \end{array} \qquad \{9\}$$

which is of course equivalent to Eq. {2}.

We now have a **new** (equivalent) LP problem, **to be treated as the original was.** The process can continue *iteratively*.

$$\begin{bmatrix} x_1 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 110 \\ 130 \\ 15 \end{bmatrix} \qquad \{10\}$$

From Eq. {2} or Eq. {9}, respectively,

$$z = [0{,}56 \quad 0 \quad 0] \begin{bmatrix} 110 \\ 130 \\ 15 \end{bmatrix} = 61{,}6 \qquad \{11\}$$

$$z = [0 \quad 0 \quad 0] \begin{bmatrix} 110 \\ 130 \\ 15 \end{bmatrix} + 61{,}6 = 61{,}6 \qquad \{12\}$$

Now, x_2 is the new entering variable. According to the constraints, it can be increased till:

$$\boxed{\mathbf{B}} \qquad \begin{array}{lllll} 2x_2 & = 130 & & x_2 & = 65 \\ x_2 & = 15 & \rightarrow & x_2 & = 15 \\ 0x_2 & = 110 & & x_2 & = \infty \end{array} \qquad \{13\}$$

$$\boxed{\mathbf{C}} \qquad x_2 = 15 - x_4 + 1{,}5x_5 \qquad \{14\}$$

Substituting for x_2 everywhere (except its own constraint), we have

$$\begin{array}{lllllll} [\max]z = & 0{,}42(15 - x_4 + 1{,}5x_5) & & -0{,}56x_5 & & +61{,}6 & \\ & +2(15 - x_4 + 1{,}5x_5) & +x_3 & -x_5 & = & 130 & \\ & x_2 & +x_4 & -1{,}5x_5 & = & 15 & \\ x_1 & & & +x_5 & = & 110 & \end{array} \qquad \{15\}$$

$$\boxed{\text{A}} \qquad \begin{array}{llllllll} [\max]z = & & & -0{,}42x_4 & +0{,}07x_5 & & +67{,}9 \\ & & \{x_3\} & -2x_4 & +2x_5 & = & 100 \\ & \{x_2\} & & +x_4 & -1{,}5x_5 & = & 15 \\ \{x_1\} & & & & +x_5 & = & 110 \end{array} \qquad \{16\}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 110 \\ 15 \\ 100 \end{bmatrix} \qquad \{17\}$$

Now, x_5 is the new entering variable. According to the constraints, it can be increased till:

$$\boxed{\text{B}} \qquad \begin{array}{ll} 2x_5 & = 100 \\ -1{,}5x_5 & = 15 \\ x_5 & = 110 \end{array} \rightarrow \begin{array}{ll} x_5 & = 50 \\ x_5 & = \ldots \\ x_5 & = 110 \end{array} \qquad \{18\}$$

$$\boxed{\text{C}} \qquad x_5 = 50 - \frac{1}{2}x_3 + x_4 \qquad \{19\}$$

Substituting for x_5 everywhere (except its own constraint), we have

$$\begin{array}{llllllll} [\max]z = & & & -0{,}42x_4 & +0{,}07\left(50 - \frac{1}{2}x_3 + x_4\right) & & +67{,}9 \\ & & x_3 & -x_4 & +x_5 & = & 50 \\ & x_2 & & +x_4 & -1{,}5\left(50 - \frac{1}{2}x_3 + x_4\right) & = & 15 \\ x_1 & & & & +\left(50 - \frac{1}{2}x_3 + x_4\right) & = & 110 \end{array} \qquad \{20\}$$

$$\boxed{\text{A}} \qquad \begin{array}{llllllll} [\max]z = & & -0{,}035x_3 & -0{,}35x_4 & & & +71{,}4 \\ & & x_3 & -x_4 & +\{x_5\} & = & 50 \\ & \{x_2\} & +0{,}75x_3 & -0{,}5x_4 & & = & 90 \\ \{x_1\} & & -0{,}5x_3 & +x_4 & & = & 60 \end{array}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_5 \end{bmatrix} = \begin{bmatrix} 60 \\ 50 \\ 50 \end{bmatrix} \qquad \{21\}$$

Now, no variable produces an increase. So, this is a *maximum.*

In sum:

A In the system of equations, find the identity matrix (immediate solution).
B search for an *entering* variable (or finish)
C consequently, find a *leaving* variable (if wrongly chosen, negative values will appear).

References:

– ZIONTS, Stanley, 1974, “Linear and integer programming”, Prentice-Hall, Englewood Cliffs, NJ (USA), p 5. (IST Library.) ISBN 0-13-536763-8.

– See others on the course webpage *(http://web.ist.utl.pt/mcasquilho)*.

❖

Mar-2011 **Zionts, "An intuitive algebraic approach for solving LP problems"**

$$[\max]\, z = 0{,}56x_1 + 0{,}42x_2$$
$$\text{s.to} \quad x_1 + 2x_2 \le 240$$
$$1{,}5x_1 + x_2 \le 180$$
$$x_1 \le 110$$

Solver model:

X =	60	90				[max] *z* =	***71,4***	7
(0)	0,56	0,42						
(Constr.:)	x_1	x_2			Value		**RHS**	
(1)	1	2	<=	240	240	**<=**	240	TRUE
(2)	1,5	1	<=	180	180	**<=**	180	100
(3)	1	0	<=	110	60	**<=**	110	100

Let's solve it manually.

		Structural—...		Slack—...				MAXIMIZE			
		x_1	x_2	x_3	x_4	x_5					
		0,56	0,42	0	0	0			Ratio		
0.1	x_3	1	2	1	0	0		**240**	240		
0.2	x_4	1,5	1	0	1	0		**180**	120		
0.3	x_5	1	0	0	0	1		**110**	***110***	Smallest **Leaves**	
0.0	Coeffs	**0,56**	**0,42**	0	0	0					
		Enters					*z* =	0	61,6	= next *z*	
1.1	x_3	0	2	1	0	-1		**130**	65	b) {0.1} - {1.3} ×	1
1.2	x_4	0	1	0	1	-1,5		**15**	***15***	c) {0.2} - {1.3} ×	1,5
1.3	x_1	1	0	0	0	1		**110**	-1	a) Pivot {0.3} /	1
1.0	Coeffs	0	0,42	0	0	-0,56	Indep. term.:	*61,6*		d) {0.0} - {1.3} ×	0,56
			Enters				*z* =	61,6	67,9	= next *z*	
2.1	x_3	0	0	1	-2	2		**100**	***50***	b) {1.1} - {2.2} ×	2
2.2	x_2	0	1	0	1	-1,5		**15**	-1	a) Pivot {1.2} /	1
2.3	x_1	1	0	0	0	1		**110**	110	c) {1.3} - {2.2} ×	0
2.0	Coeffs	0	0	0	-0,42	0,07	Indep. term.:	*67,9*		d) {1.0} - {1.3} ×	0,42
						Enters	*z* =	67,9	71,4	= next *z*	
3.1	x_5	0	0	0,5	-1	1		**50**		a) Pivot {2.1} /	2
3.2	x_2	0	1	0,75	-0,5	0		**90**		b) {2.2} - {3.1} ×	-1,5
3.3	x_1	1	0	-0,5	1	0		**60**		c) {2.3} - {3.1} ×	1
3.0	Coeffs	0	0	-0,035	-0,35	0	End	Indep. term.: *71,4*		d) {1.0} - {1.3} ×	0,07
							z =	71,4	OPTIMUM		

Mar-2011 **H&L, Wyndor Probl.**

$$[\max] z = 3x_1 + 5x_2$$
$$\text{s.to} \quad x_1 \le 4$$
$$+2x_2 \le 12$$
$$3x_1 + 2x_2 \le 18$$

								Solver model
X =	2	6						
(0)	3	5				[max] z =	***36***	36
(Constr.:)	x_1	x_2			Value		**RHS**	2
(1)	1	0	<=	4	2	**<=**	4	TRUE
(2)	0	2	<=	12	12	**<=**	12	100
(3)	3	2	<=	18	18	**<=**	18	100

Let's solve it manually.

		Structural—...		Slack—...			——Artificial—— **NOT USED**			MAXIMIZE			
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8				
		3	5	0	0	0	-1000	-1000	-1000		Ratio		
0.1	x_3	1	0	1	0	0	**1**	0	0	**4**	-1		
0.2	x_4	0	2	0	1	0	0	**1**	0	**12**	***6***	Smallest Leaves	
0.3	x_5	3	2	0	0	1	0	0	**1**	**18**	9		
0.0	Coeffs	3	**5**	0	0	0	0	0	0				
			Enters						z =	**0**	30	= next z	
1.1	x_3	1	0	1	0	0				**4**	4	b) {0.1} - {1.2} ×	0
1.2	x_2	0	1	0	0,5	0				**6**	-1	a) Pivot {0.2} /	2
1.3	x_5	3	0	0	-1	1				**6**	**2**	c) {0.3} - {1.2} ×	2
1.0	Coeffs	3	0	0	-2,5	0			Indep. term.:	*30*		d) {0.0} - {1.2} ×	5
		Enters							z =	30	36	= next z	
2.1	x_3	0	0	1	0,33333	-0,3333				**2**		b) {1.1} - {1.3} ×	1
2.2	x_2	0	1	0	0,5	0				**6**		c) {1.2} - {1.3} ×	0
2.3	x_1	1	0	0	-0,3333	0,33333				**2**		a) Pivot {1.3} /	3
2.0	Coeffs	0	0	0	-1,5	-1	End		Indep. term.:	*36*		d) {1.0} - {1.3} ×	3
									z =	36	OPTIMUM		

Now follow a different path !

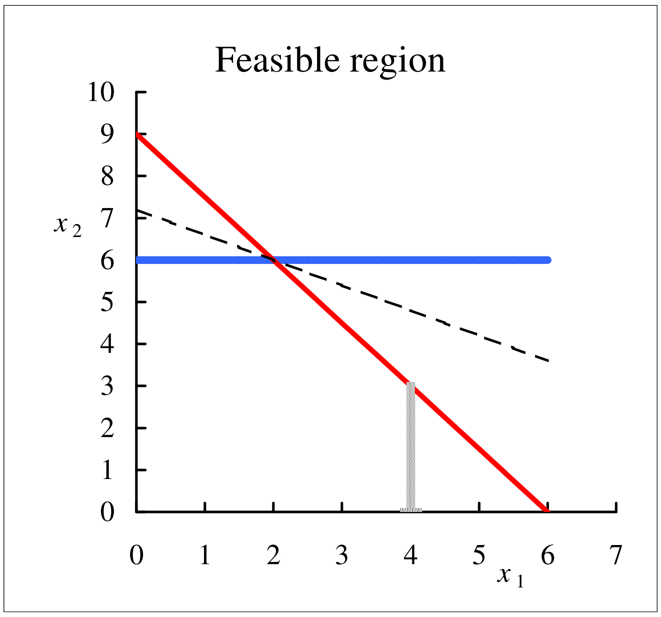
		Structural—...		Slack—...			MAXIMIZE			
		x_1	x_2	x_3	x_4	x_5				
		3	5	0	0	0		Ratio		
0.1	x_3	1	0	1	0	0	**4**	***4***	Smallest Leaves	
0.2	x_4	0	2	0	1	0	**12**	-1		
0.3	x_5	3	2	0	0	1	**18**	6		
0.0	Coeffs	3	5	0	0	0				
		Enters				z =	0	12	= next z	
1.1	x_1	1	0	1	0	0	**4**	-1	a) Pivot {0.1} /	1
1.2	x_4	0	2	0	1	0	**12**	6	b) {0.2} - {1.1} ×	0
1.3	x_5	0	2	-3	0	1	**6**	**3**	c) {0.3} - {1.1} ×	3
1.0	Coeffs	0	5	-3	0	0	Indep. term.: *12*		d) {0.0} - {1.1} ×	3
			Enters			z =	12	27	= next z	
2.1	x_1	1	0	1	0	0	**4**	4	b) {1.1} - {2.3} ×	0
2.2	x_4	0	0	3	1	-1	**6**	**2**	c) {1.2} - {2.3} ×	2
2.3	x_2	0	1	-1,5	0	0,5	**3**	-1	a) Pivot {1.3} /	2
2.0	Coeffs	0	0	4,5	0	-2,5	Indep. term.: *27*		d) {1.0} - {1.3} ×	5
				Enters		z =	27	36	= next z	
3.1	x_1	1	0	0	-0,3333	0,33333	**2**		b) {2.1} - {3.2} ×	1
3.2	x_3	0	0	1	0,33333	-0,3333	**2**		a) Pivot {2.2} /	3
3.3	x_2	0	1	0	0,5	0	**6**		c) {2.3} - {3.2} ×	-1,5
3.0	Coeffs	0	0	0	-1,5	-1 End	Indep. term.: *36*		d) {2.0} - {3.2} ×	4,5
						z =	36	OPTIMUM		

Path 1	(0, 0, z=0), (0, 6, z=30), (2, 6, z=36)
Path 2	(0, 0, z=0), (4, 0, z=12), (4, 3, z=27), (2, 6, z=36)

Try: 27 or 36

Δx = 0,5 z = ? 36

x	y_1	y_2	y_3	y_z
0		6	9	7,2
0,5		6	8,25	6,9
1		6	7,5	6,6
1,5		6	6,75	6,3
2		6	6	6
2,5		6	5,25	5,7
3		6	4,5	5,4
3,5		6	3,75	5,1
4	3	6	3	4,8
4,5		6	2,25	4,5
5		6	1,5	4,2
5,5		6	0,75	3,9
6		6	0	3,6



Artificial variables in Linear Programming

Adapted from H&L [2005] and Taha [1992]

Equality constraints [H&L, p 125]

Suppose a modification to the original *Wyndor* problem, as follows ({1}).

$$\begin{array}{llll} [\max] z = & 3x_1 & +5x_2 & \\ \text{s.to} & x_1 & & \le 4 \\ & & 2x_2 & \le 12 \\ & 3x_1 & +2x_2 & =18 \end{array} \qquad \{1\}$$

with $\boldsymbol{x} \ge 0$. Thus, the third constraint is now an equality. This can become

$$\begin{array}{lllllll} (0) & z & -3x_1 & -5x_2 & & & =0 \\ (1) & & x_1 & & +x_3 & & =4 \\ (2) & & & 2x_2 & & +x_4 & =12 \\ (3) & & 3x_1 & +2x_2 & & & =18 \end{array} \qquad \{2\}$$

However, these equations **do not have** an obvious initial (basic feasible) solution. So, the **artificial variable technique** is applied. With M a very high number (+∞) — this is the **Big M method**[*]—, we can ***augment*** the system {2} to obtain

$$\begin{array}{llllllll} (0) & z & -3x_1 & -5x_2 & & & +M\,\bar{x}_5 & =0 \\ (1) & & x_1 & & +x_3 & & & =4 \\ (2) & & & 2x_2 & & +x_4 & & =12 \\ (3) & & 3x_1 & +2x_2 & & & +\bar{x}_5 & =18 \end{array} \qquad \{3\}$$

Converting equation 0 to proper form

In {3}, the (obvious) initial basic variables are x_3, x_4 and $\bar{x}_5$ (non-basic $x_1 = 0$ and $x_2 = 0$). However, this system is not yet in proper form for Gaussian elimination because a basic variable ($\bar{x}_5$) has a **non-zero coefficient** in Eq. 0. Indeed, all the basic variables must be (algebraically) eliminated from Eq. 0 before the simplex method can find the entering basic variable. (This elimination is necessary so that the negative of the coefficient of each non-basic variable will give the rate at which z would increase if that non-basic variable were to be increased from 0 while adjusting the values of the basic variables accordingly.)

To eliminate $\bar{x}_5$ from Eq. 0, we need to subtract from Eq. 0 the product M times Eq. 3:

$$\begin{array}{lllll} z & -3x_1 & -5x_2 & +M\,\bar{x}_5 & =0 \\ & -M(3x_1 & +2x_2 & +\bar{x}_5 & =18) \\ \hline z & -(3M+3)x_1 & -(2M+5)x_2 & & =-18M \end{array} \qquad \{4\}$$

[*] Another method to solve this matter is the “two-phase method”.

In this example, there is only one equation with an artificial variable. If there were **several** equations with artificial variables, we would have to subtract accordingly.

Application of the simplex method

The new Eq. 0 gives z in terms of just the non-basic variables (x_1, x_2):

$$z = -18M + (3M + 3)x_1 + (2M + 5)x_2 \tag{5}$$

Since the coefficient of x_1 is the **best** (greatest), this variable is chosen as the *entering* variable.

The leaving variable, as always, will correspond to the smallest "positive" (non-negative) ratio (from the so-called "minimum ratio test").

Another (more general) example (Taha [1992], p 72)

$$\begin{array}{rrrrr} [\min] z = & 4x_1 & + x_2 & & \\ \text{s.to} & 3x_1 & + x_2 & = 3 \\ & 4x_1 & + 3x_2 & \geq 6 \\ & x_1 & + 2x_2 & \leq 4 \end{array} \tag{6}$$

with $\boldsymbol{x} \geq 0$. The *augmented* standard form is

$$\boxed{\begin{array}{rrrrrrrr} [\min] z = & 4x_1 & + x_2 & + 0x_3 & + 0x_4 & + Ma_1 & + Ma_2 & \\ \text{s.to} & 3x_1 & + x_2 & & & + a_1 & & = 3 \\ & 4x_1 & + 3x_2 & - x_3 & & & + a_2 & = 6 \\ & x_1 & + 2x_2 & & + x_4 & & & = 4 \end{array}} \tag{7}$$

References

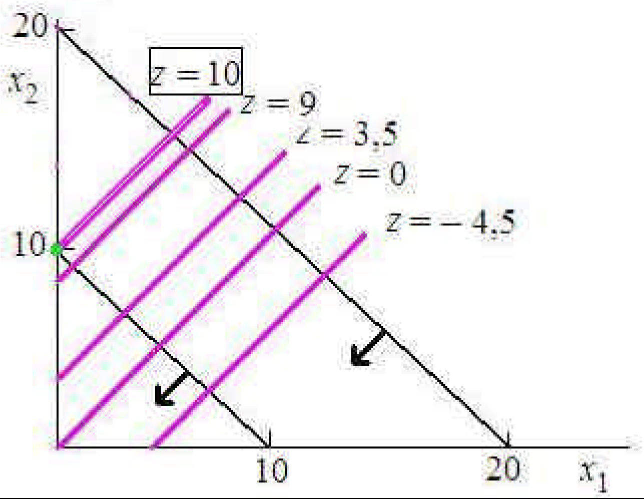
- HILLIER, Frederick S., and Gerald J. LIEBERMAN, 2005, "Introduction to Operations Research", 8.th ed., McGraw-Hill
- TAHA, Hamdy, 1992, "Operations Research: an introduction", 5.th ed., MacMillan Publishing Company

❖

(Blank page)

Redundant ?

	$[\max] x_2 - x_1$ $x_1 + x_2 \le 10$ $x_1 + x_2 \le 20$	{1}



	$$\begin{array}{lllll} [\max] & z = & -x_1 & +x_2 & \\ \text{s.to} & & x_1 & +x_2 & \le 10 \\ & & x_1 & +x_2 & \le 20 \end{array}$$	{2}

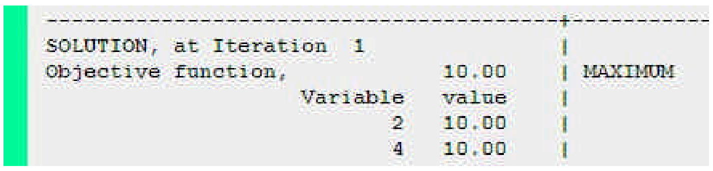
	$$\begin{array}{llllllll} [\max] & z = & -x_1 & +x_2 & +0x_3 & +0x_4 & & \\ \text{s.to} & & x_1 & +x_2 & +x_3 & & = & 10 \\ & & x_1 & +x_2 & & +x_4 & = & 20 \end{array}$$	{3}

Solve:

Go to http://web.ist.utl.pt/~mcasquilho/compute/or/Fx-lp-revised.php

Supply:

Opt.	max
Coefficients	-1 1 0 0
$A \mid B$	1 1 1 0 10 1 1 0 1 20
Artificials	0
Initial basis	3 4



```
-----------------------------------------+-----------
SOLUTION, at Iteration  1                |
Objective function,            10.00     | MAXIMUM
                      Variable  value     |
                             2  10.00     |
                             4  10.00     |
```

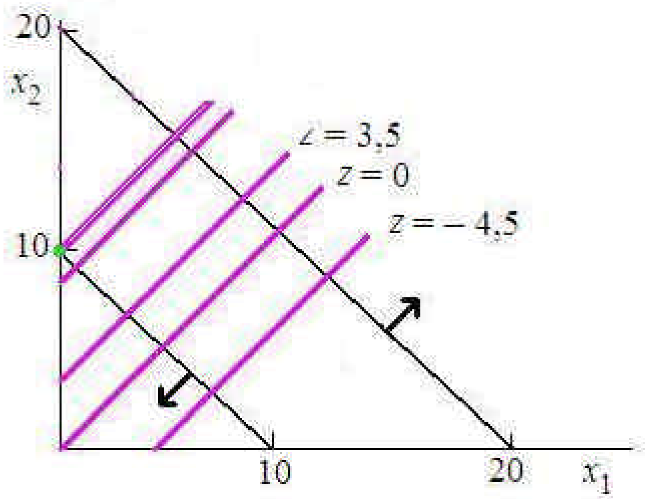
Redundant ? No problem.

❖

2010 — M. Casquilho (IST)
File: {LP_redund.doc}

Impossible ?

	$[\max] x_2 - x_1$ $x_1 + x_2 \le 10$ $x_1 + x_2 \ge 20$	{1}



	$$\begin{array}{lllll} [\max] & z = & -x_1 & +x_2 & \\ \text{s.to} & & x_1 & +x_2 & \le 10 \\ & & x_1 & +x_2 & \ge 20 \end{array}$$	{2}

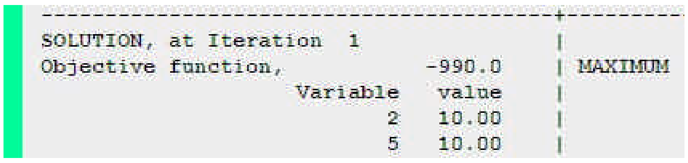
$M \cong +\infty$

	$$\begin{array}{lllllll} [\max] & z = & -x_1 & +x_2 & +0x_3 & +0x_4 & -Mx_5 \\ \text{s.to} & & x_1 & +x_2 & +x_3 & & = 10 \\ & & x_1 & +x_2 & & -x_4 & +x_5 \quad 20 \end{array}$$	{3}

Solve:

Go to http://web.ist.utl.pt/~mcasquilho/compute/or/Fx-lp-revised.php
Supply:

Opt.	max
Coefficients	-1 1 0 0 0
$A \mid B$	1 1 1 0 0 10 1 1 0 -1 1 20
Artificials	5
Big M	1+2
Initial basis	3 5


```
----------------------------------------+----------
SOLUTION, at Iteration  1               |
Objective function,          -990.0     | MAXIMUM
                    Variable   value    |
                           2   10.00    |
                           5   10.00    |
```

Impossible ? No problem.

❖

2010 — M. Casquilho (IST)
File: {LP_imposs.doc}

A “scientific application” (!) of Linear Programming

In Ecker & Kupferschmid [1988], Ch. 2, “LP models and applications”, 2.3, “Some scientific applications of LP”, pp 24–25; example problem from Guttman *et al.* [1982], Ch. 15, “Regression analysis”, 15.5, An example, pp 361–365

A study was instituted to determine the percent of waste solids removed in a filtration system as a function of flow rate, *x*, of the effluentbeing fed into the system. It was decided to use flow rate of 2 (2) 14 gal/min and to observe y^e (“experimental”), the percent of waste solid removed,when each of these flow rates was used. The study yielded the data displayed in Table 1.

The mathematical model $E(y|x) = ax + b$ was proposed.

Find the parameters, *a* and *b*, of the model [Guttman, *et al.*, 1982, p 361].

Table 1

i	x	y^e
1	2	24,3
2	4	19,7
3	6	17,8
4	8	14,0
5	10	12,3
6	12	7,2
7	14	5,5

® Resolution

a) Classical solution

(We will use only points *1*, *4* and *7* of Table 1. With all the points, the source cited gives $\hat{y}=26{,}81-1{,}55x$, “in the sense of least squares”.)

As is well known, the parameters of the problem are obtained minimizing a sum of errors (squared, for convenience), of the form

$$z=\sum_{i=1}^{n}\left(y_i-y_i^e\right)^2 \qquad \{1\}$$

with

z – measure (a sum) of the n errors. ([z] = y^2, see below)
n – number of experiments
y_i – theoretical (or “calculated”) value, $y = ax + b$, of the measured variable, corresponding to x_i (*i* integer, i = 1..n)
a, b – process parameters. (With c and y the dimensions of x and y, respectively, it is [a] = $y\,c^{-1}$ and [b] = y.)
y_i^e – experimental value (a constant, thus) of the measured variable, corresponding to x_i

So, z [1] is a function of only a and b, whose minimum is easy to find by differentiation,

[1] The use of $\sqrt{z}$, as may be concluded, would be more logical, although indifferent from the viewpoint of optimization.

giving for these parameters, as is known,

$$\begin{bmatrix} \hat{a} \\ \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \dfrac{\sum (x_i - \bar{x}) y_i^e}{\sum (x_i - \bar{x})^2} \\ \\ \bar{y}^e - \hat{a}\bar{x} \end{bmatrix} \qquad \{2\}$$

while the optimum of *z* is not relevant.

Table 2

i	x_i	y_i^e	$x_i - \bar{x}$	$(x_i - \bar{x}) y_i^e$	$(x_i - \bar{x})^2$
1	**2**	**24,3**	-6	-145,8	36,00
2	4	19,7			
3	6	17,8			
4	**8**	**14,0**	0	0,0	0,00
5	10	12,3			
6	12	7,2			
7	**14**	**5,5**	6	33	36,00
Sum	24	43,8	(0)	-112,8	72,00
Average	$\bar{x} = 8$	$\bar{y} = 14,6$			

From Table 2, for the points selected, it is

$\hat{a}$ = –1,5(6) (% removed) / (gal/min)
$\hat{b}$ = 27,1(3) (% removed)

(These values are near the values reported for the 7 points, $\hat{a}$ = –1,55 and $\hat{b}$ = 26,81.)

The calculated *y*'s [2] give: $y_1 = 24$ ("low", *vs.* 24,3), $y_4 = 14,6$ ("high", *vs.* 14,0), $y_7 = 5,2$ ("low", *vs.* 5,5).

b) **Solution by Linear Programming**

(We use the same points, *1*, *4* e *7*, from Table 1.)

We propose, now, to obtain the parameters *a* and *b*, of the same model, by another criterion: make the "errors" or deviations be small individually (not as a sum). To this end, we will try to minimize the "worst" (largest) deviation, i.e.,

$$[\min] \quad z = \max_i (|d_i|) \qquad \{3\}$$

with

$$d_i = y_i^e - y_i \qquad \{4\}$$

or, since (in this case) it is $y = ax + b$,

[2] The plural in the form "*x*'s" seems appropriate (better than "*xx*").

$$d_i = y_i^e - ax_i - b \qquad \{5\}$$

Remember that any value of z depends only of a and b, as all other values are constants (experimental values). (This time, the physical dimensions of z are, of course, the same as those of the measured variable, y.)

As z must be the ***maximum*** deviation, it has, equivalently, to satisfy each of the following inequalities

$$z \geq \left| y_i^e - ax_i - b \right| \qquad i = 1..n \qquad \{6\}$$

with n the number of points.

The problem becomes, thus, to find a, b and z such that all the inequalities should be satisfied and z be as small as possible. So, we need to find

$$[\min]\ z \qquad \{7\}$$

subject to

$$z \geq \left| y_i^e - ax_i - b \right| \qquad i = 1..n \qquad \{8\}$$

Now, this problem has the disadvantage of not being a linear programming, in this form, because, of course, the *'absolute value'* [3] *of a linear function is not a linear function* [Ecker *et al.*, 1988]. We can, however, convert it into a linear program through the following elementary fact:

> For any z and w,
> $z \geq |w|$ iff $z \geq w$ and $z \geq -w$.

So, let us replace each (non-linear) inequality by two linear inequalities, to get a linear program:

$$[\min]\ z \qquad \{9\}$$

subject to

$$z \geq +\left(y_i^e - ax_i - b \right) \qquad i = 1..n \qquad \{10a\}$$

$$z \geq -\left(y_i^e - ax_i - b \right) \qquad i = 1..n \qquad \{10b\}$$

$$z \geq 0; \quad a, b\text{: of free sign}$$

As is known, a and b can be replaced by *differences of non-negative variables,* say, $a' - a''$ e $b' - b''$. Incidentally, as we have (possibly good) approximations of the optimum values of a and b, from the previous section, we can simply just replace a by $-a'$ (a' non-negative) —an artifice that must be verified in the end (and which would be under suspicion in case we obtained the boundary value $a' = 0$).

The problem then becomes:

$$[\min]\ z \qquad \{11\}$$

subject to

[3] Or "modulus".

$$x_i a' - b + z \geq -y_i^e \qquad i = 1..n \qquad \{12a\}$$

$$-x_i a' + b + z \geq y_i^e \qquad i = 1..n \qquad \{12b\}$$

or, finally, introducing the numerical values,

$$[\min]\ z \qquad \{13\}$$

subject to

$$\begin{aligned} 2a' - b + z &\geq -24{,}3 \\ 8a' - b + z &\geq -14{,}0 \\ 14a' - b + z &\geq -5{,}5 \\ -2a' + b + z &\geq 24{,}3 \\ -8a' + b + z &\geq 14{,}0 \\ -14a' + b + z &\geq 5{,}5 \end{aligned} \qquad \{14\}$$

In matrix form, it is

$$\begin{array}{llll} [\min] & w & = & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to:} & & \mathbf{Ax} & \geq \mathbf{b} \\ & & \mathbf{x} & \geq \mathbf{0} \end{array} \qquad \{15\}$$

with

$$\mathbf{x} = [a' \quad b \quad z]^{\mathrm{T}}$$

$$\mathbf{c}^{\mathrm{T}} = [0 \quad 0 \quad 1]$$

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 1 \\ 8 & -1 & 1 \\ 14 & -1 & 1 \\ -2 & 1 & 1 \\ -8 & 1 & 1 \\ -14 & 1 & 1 \end{bmatrix} \mathbf{b} = \begin{bmatrix} -24{,}3 \\ -14{,}0 \\ -5{,}5 \\ 24{,}3 \\ 14{,}0 \\ 5{,}5 \end{bmatrix} \qquad \{16\}$$

***i)* Direct resolution**

The problem, as just formulated, has 3 structural variables and 6 constraints. Its manual resolution, thus, faces the practically unfeasible handling of square matrices of order 6, among others. The computer resolution took 5 iterations and gave (as structural variables and objective function):

$$\begin{bmatrix} a' \\ b \\ z \end{bmatrix} = \begin{bmatrix} 1{,}56667 \\ 26{,}9833 \\ 0{,}45 \end{bmatrix} \qquad (z = 0{,}45) \qquad \{17\}$$

So, we have $a = -a' = -1{,}56667$ and $b = 26{,}9833$. Notice that it is $a' \neq 0$ (and, inevitably, $a' > 0$), as expected, which validates the hypothesis made to ease the calculations, so this result is not "suspect".

It is not evident whether this set (a, b) is better or worse than the former (otherwise, it happens that one of the values coincides), a fact that depends on the finalities.

ii) Resolution by the *dual* (brief note)

The LP problems can be grouped in pairs, where one of the problems is the ***primal*** and the other the ***dual*** —an assignment that is arbitrary, although usually the primal corresponds to the original problem. Duality —present in various areas of Mathematics— is important both theoretically and practically in LP, as both problems yield an ***identical optimum*** (if it exists) for the **objective function**. Moreover: indeed, in the complete solution of one of the problems, the complete solution of the other can be read, with the advantage that, frequently, **one of them is computationally** (much) **less difficult**.

The relationship between primal and dual, explored in the LP literature, may be shortly presented as follows, conveniently for theory and application:

Primal	**Dual**
$[\min] \quad z = \mathbf{c}^{\mathrm{T}}\mathbf{x}$ subject to: $\mathbf{Ax} \geq \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	$[\max] \quad z = \mathbf{b}^{\mathrm{T}}\mathbf{y}$ subject to: $\mathbf{A}^{\mathrm{T}}\mathbf{y} \leq \mathbf{c}$ $\mathbf{y} \geq \mathbf{0}$

The case under study corresponds to the classification above; in other cases, the descriptions under the titles *primal* and *dual* would be exchanged.

Among other properties, it can be proved that:

- If one of the problems has an optimum vector (solution), then the other also has one, and the optimum objective function is identical.
- If one of the problems is possible but has no finite optimum, then the other is impossible.
- Both problems can be impossible.
- The optimum vector for the *maximization* has its elements equal to the coefficients of the slack variables of the optimum basis of the *minimization*, and reciprocally.

Therefore, starting from the original problem under study, which has 3 structural variables and 6 constraints (two per each experimental point), its dual can be constructed, having 6 structural variables and only 3 constraints. So, in this case, the dual ***(a)*** evolves by much easier iterations (matrices of order 3, not 6), and ***(b)*** will be, expectedly, less cumbersome, as it will yield about half the iterations (about proportional to 3, not 6). Using the dual would still allow to easily consider *all the experimental points*, even if more numerous, as the number of iterations till the optimum depends essentially on the number of constraints.

The dual would be:

$$[\max]\ (w=)24{,}3s_1+14{,}0s_2+5{,}5s_3-24{,}3s_4-14{,}0s_5-5{,}5s_6 \qquad \{18\}$$

subject to

$$\begin{bmatrix} 2 & 8 & 14 & -2 & -8 & -14 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{bmatrix} \le \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \{19\}$$

The result, in 4 iterations (instead of 5), is (of course)

$$z=0{,}45 \qquad \{20\}$$

and contains —in its so-called *dual variables*— the values

$$\mathrm{D}=\begin{bmatrix}-1{,}567 & -26{,}98 & -0{,}45\end{bmatrix}^{\mathrm{T}} \qquad \{21\}$$

Consequently, this vector (always *negative* —i.e., non-positive— in the optimum of a *maximization*, of course) has as elements the *symmetrical* of the results (*a´*, *b*, *z*) of the primal, already known.

¬

References

– ECKER, Joseph G., Michael KUPFERSCHMID, 1988, “Introduction to Operations Research”, John Wiley & Sons, New York, NY (USA), ISBN 0-471-63362-3.

– GUTTMAN, Irwin, Samuel S. WILKS, J. Stuart HUNTER, 1982, “Introduction to Engineering Statistics”, 3.rd ed., John Wiley & Sons, New York, NY (USA), ISBN 0-471-86956-2.

❖

Bronson & Naadimuthu, 1997, pp 56–57

Chapter 4

Linear Programming: Duality and Sensitivity Analysis

Every linear program in the variables $x_1, x_2, \ldots, x_n$ has associated with it another linear program in the variables $w_1, w_2, \ldots, w_m$ (where m is the number of constraints in the original program), known as its *dual*. The original program, called the *primal*, completely determines the form of its dual.

SYMMETRIC DUALS

The dual of a (primal) linear program in the (nonstandard) matrix form

$$\begin{aligned} \text{minimize:} \quad & z = \mathbf{C}^T\mathbf{X} \\ \text{subject to:} \quad & \mathbf{AX} \geq \mathbf{B} \\ \text{with:} \quad & \mathbf{X} \geq \mathbf{0} \end{aligned} \tag{4.1}$$

is the linear program

$$\begin{aligned} \text{maximize:} \quad & z = \mathbf{B}^T\mathbf{W} \\ \text{subject to:} \quad & \mathbf{A}^T\mathbf{W} \leq \mathbf{C} \\ \text{with:} \quad & \mathbf{W} \geq \mathbf{0} \end{aligned} \tag{4.2}$$

Conversely, the dual of program (*4.2*) is program (*4.1*). (See Problems 4.1 and 4.2.)

Programs (*4.1*) and (*4.2*) are symmetrical in that both involve nonnegative variables and inequality constraints; they are known as the *symmetric duals* of each other. The dual variables $w_1, w_2, \ldots, w_m$ are sometimes called *shadow costs*.

DUAL SOLUTIONS

Theorem 4.1 (Duality Theorem): If an optimal solution exists to either the primal or symmetric dual program, then the other program also has an optimal solution and the two objective functions have the same optimal value.

In such situations, the optimal solution to the primal (dual) is found in the last row of the final simplex tableau for the dual (primal), in those columns associated with the slack or surplus variables (see Problem 4.3). Since the solutions to both programs are obtained by solving either one, it may be computationally advantageous to solve a program's dual rather than the program itself. (See Problem 4.4.)

Theorem 4.2 (Complementary Slackness Principle): Given that the pair of symmetric duals have optimal solutions, then if the kth constraint of one system holds as an inequality—i.e., the associated slack or surplus variable is positive—the kth component of the optimal solution of its symmetric dual is zero.

(See Problems 4.11 and 4.12.)

UNSYMMETRIC DUALS

For primal programs in standard matrix form, duals may be defined as follows:

Primal		***Dual***	
minimize: $z = \mathbf{C}^T\mathbf{X}$ subject to: $\mathbf{AX} = \mathbf{B}$ with: $\mathbf{X} \geq \mathbf{0}$	*(4.3)*	maximize: $z = \mathbf{B}^T\mathbf{W}$ subject to: $\mathbf{A}^T\mathbf{W} \leq \mathbf{C}$	*(4.4)*
maximize: $z = \mathbf{C}^T\mathbf{X}$ subject to: $\mathbf{AX} = \mathbf{B}$ with: $\mathbf{X} \geq \mathbf{0}$	*(4.5)*	minimize: $z = \mathbf{B}^T\mathbf{W}$ subject to: $\mathbf{A}^T\mathbf{W} \geq \mathbf{C}$	*(4.6)*

(See Problems 4.5 and 4.6.) Conversely, the duals of programs *(4.4)* and *(4.6)* are defined as programs *(4.3)* and *(4.5)*, respectively. Since the dual of a program in standard form is not itself in standard form, these duals are *unsymmetric*. Their forms are consistent with and a direct consequence of the definition of symmetric duals (see Problem 4.8).

Theorem 4.1 is valid for unsymmetric duals too. However, the solution to an unsymmetric dual is not, in general, immediately apparent from the solution to the primal; the relationships are

$$\mathbf{W}^{*T} = \mathbf{C}_0^T\mathbf{A}_0^{-1} \qquad \text{or} \qquad \mathbf{W}^* = (\mathbf{A}_0^T)^{-1}\mathbf{C}_0 \tag{4.7}$$

$$\mathbf{X}^{*T} = \mathbf{B}_0^T(\mathbf{A}_0^T)^{-1} \qquad \text{or} \qquad \mathbf{X}^* = \mathbf{A}_0^{-1}\mathbf{B}_0 \tag{4.8}$$

In *(4.7)*, $\mathbf{C}_0$ and $\mathbf{A}_0$ are made up of those elements of $\mathbf{C}$ and $\mathbf{A}$, in either program *(4.3)* or *(4.5)*, that correspond to the *basic variables* in $\mathbf{X}^*$; in *(4.8)*, $\mathbf{B}_0$ and $\mathbf{A}_0$ are made up of those elements of $\mathbf{B}$ and $\mathbf{A}$, in either program *(4.4)* or *(4.6)*, that correspond to the *basic variables* in $\mathbf{W}^*$. (See Problem 4.7.)