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.)

MONTE CARLO

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.)

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.)

Alston Householder

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

SIMULATION routine

("Monte Carlo Method")

Simulation of random phenomena, based on (pseudo-)random numbers

1) Define the phenomenon to simulate (random variable, x) and ***(a)*** caracterize it through an adequate probability function or ***(b)*** go to **2**.

DISCRETE VARIABLE	**CONTINUOUS VARIABLE**
Let $f(x)$ be the (point) probability function.	Let $f(x)$ be the probability density function.

2) Determine the cumulative (distribution) function.

$F(x) = \sum_{t=x_{\min}}^{x} f(t)$	$F(x) = \int_{x_{\min}}^{x} f(t)\,\mathrm{d}t$

($x_{\min}$ is the least value possible for x.)

Whenever the analytical form of $F(x)$ is unknown or difficult to deduce, it is necessary to tabulate $F(x)$.

3) Adopt uniform random numbers of k digits, with k "compatible" with the values of $F(x)$ to use (same number of significant digits).

(The value of k depends of the circumstances of the problem.)

4) *(Simulation proper)* Extract a random number, N, and divide it by 10^k, to obtain the random number u, reduced to the interval [0, 1).

$$u = N / 10^k$$

5) Solve the equation "$F(x) = u$" for x, i.e.,

$F(x-1) \le u < F(x)$	$F(x) = u$
[with $F(x_{\min} - 1) = 0$] whence x is obtained.	If the function F has a known inverse, $\tilde{F}$, $x = \tilde{F}(u)$

The value x is a simulated value of the random variable under consideration.

6) Return to the simulation procedure —steps **4** and **5**— until a "sufficient" number of values is calculated; or finish.

■

$$y = f(x) = \frac{1}{b-a}$$

$$\mu = E(x) = \int_{-\infty}^{+\infty} x f(x)\,\mathrm{d}x =$$

$$= \int_a^b \frac{x}{b-a}\,\mathrm{d}x = \frac{1}{b-a}\int_a^b x\,\mathrm{d}x = \frac{1}{b-a}\left[\frac{x^2}{2}\right]_a^b =$$

$$= \frac{1}{2}\frac{1}{b-a}\left(b^2 - a^2\right) = \frac{1}{2}\frac{(b+a)(b-a)}{b-a}$$

$$\boxed{\mu = \frac{a+b}{2}}$$

$$\sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)\,\mathrm{d}x =$$

$$= \int_a^b \frac{\left(x - \dfrac{a+b}{2}\right)^2}{b-a}\,\mathrm{d}x$$

Let $u = x - \dfrac{a+b}{2}$.

$$\sigma^2 = \frac{1}{b-a}\int_{-(b-a)/2}^{(b-a)/2} u^2\,\mathrm{d}u = \frac{1}{b-a}\left[\frac{u^3}{3}\right]_{-(b-a)/2}^{(b-a)/2} =$$

$$= \frac{1}{3}\frac{1}{b-a}\left[\left(\frac{b-a}{2}\right)^3 + \left(\frac{b-a}{2}\right)^3\right] = \frac{1}{3}\frac{1}{b-a}\frac{2}{8}(b-a)^3 =$$

$$= \frac{1}{12}(b-a)^2$$

$$\boxed{\sigma = \frac{b-a}{\sqrt{12}}}$$

Physical dimensions match, as expected !

Useful mathematical tools

MIGUEL A. S. CASQUILHO
Technical University of Lisbon, 1049-001 Lisboa, Portugal

Some useful mathematical tools are presented: the Newton-Raphson method; surrogate Gaussian distributions; and notes on the Monte Carlo (simulation) method.

Key words: *Newton-Raphson method, Gaussian distribution, Monte Carlo, Simulation.*

1. Introduction

Useful mathematical tools (some of which you may have forgotten) are presented: the Newton-Raphson method; surrogate Gaussian distributions; and some notes on the Monte Carlo simulation method. Pertinent illustrations are included.

2. The Newton-Raphson method

The Newton-Raphson[1] method is a well-known numerical method to find (approximate) zeros (or "roots") of a function. It is an iterative algorithm[2], which, when successful, converges (usually) rapidly (quadratically, i.e., doubling the number of correct figures in each iteration), but may fail as any other root-finding algorithm.

The method tries to solve an equation in the form of Eq. {1}

$$f(x) = 0 \qquad \{1\}$$

through the iterative procedure of Eq. {2},

$$x^* = x - \frac{f(x)}{f'(x)} \qquad \{2\}$$

from an initial estimate of x, usually called the *initial guess*. Although the notation x^* is often used to indicate the solution to a problem (Eq. {1}), here it is used to mean the next, (hopefully) improved value of the variable, which, in the end, will indeed be the solution.

The bibliographical sources of the method are so numerous that no specific recommendation is made in this opuscule.

Eq. {2} shows that: if we know the solution, i.e., the x for which it is $f(x) = 0$, then, of course, the next x is the same as the current one, so the process is terminated; and if the derivative is null in the solution, i.e., $f'(x) = 0$, the method fails (which happens, namely, for multiple roots). Failure to converge may, of course, happen in

[1] Isaac NEWTON (1643–1727), Joseph RAPHSON (~1648–~1715), English mathematicians ([MacTutor, 2010])
[2] From Abu Jafar Muhamad Ibn Musa Al-Kwarizmi ([MacTutor, 2010])

Prof. Dr. Casquilho is an Assistant Professor in the Department of Chemical Engineering at Instituto Superior Técnico, Technical University of Lisbon, Portugal. His email address is mcasquilho@ist.utl.pt.

any iterative numerical method. Anyway, the convergence of the Newton-Raphson method, when it arises, is typically quite rapid (few iterations).

Illustration 2-A

Solve

$$f(x) = ax + b = 0 \qquad \{3\}$$

from $x = 1$.

Resolution Applying the algorithm, it is

$$f'(x) = a \qquad \{4\}$$

$$x^* = x - \frac{ax+b}{a} = x - x - \frac{b}{a} = -\frac{b}{a} \qquad \{5\}$$

In this case, we did not even have the opportunity to supply the initial guess to get the (exact) solution.

Illustration 2-B

Solve

$$f(x) = 2x^2 - 6x = 0 \qquad \{6\}$$

from $x = \pm 5$.

Resolution Applying the algorithm, with the simple derivative, it is

$$x^* = x - \frac{2x^2 - 6x}{4x - 6} \qquad \{7\}$$

Eq. {7} could be simplified to $x^* = x - x(x-3)/(2x-3)$ —not necessary—, just to show that 0 and 3 are, of course, the roots of the given equation. The function is shown in Fig. 1 and the computations in Table 1.

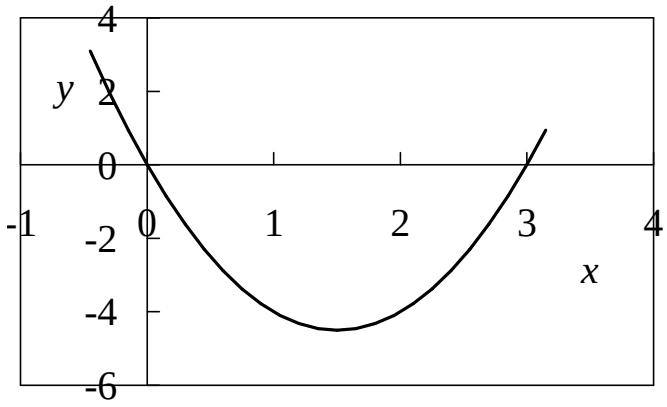

Fig. 1

Table 1

x	$f(x)$	$f'(x)$	new x	x	$f(x)$	$f'(x)$	new x
-5	80	-26	-1,92308	5	20	14	3,571429
-1,92308	18,93491	-13,6923	-0,54019	3,571429	4,081633	8,285714	3,078818
-0,54019	3,824752	-8,16076	-0,07151	3,078818	0,485331	6,315271	3,001967
-0,07151	0,439314	-6,28606	-0,00163	3,001967	0,011812	6,007869	3,000001
-0,00163	0,009768	-6,00651	-8,8E-07	3,000001	7,73E-06	6,000005	3
-8,8E-07	5,29E-06	-6	-2,6E-13	3	3,32E-12	6	3
-2,6E-13	1,55E-12	-6	-2,2E-26	3	0	6	3
-2,2E-26	1,34E-25	-6	0				
0	0	-6	0				

In this simple case, the two zeros (or "roots", a more usual term for polynomials) were obtained from respective "reasonable" initial guesses.

Illustration 2-C

Solve

$$\sin x = 1/2 \qquad \{8\}$$

from $x = 0$. (We know that $x = \arcsin(1/2) = \pi/6 = 0.523...$[3])

Resolution First, drive Eq. {8} to the convenient form (Eq. {2}).

$$f(x) = (\sin x) - 1/2 = 0 \qquad \{9\}$$

Thus,

$$f'(x) = \cos x \qquad \{10\}$$

$$\tilde{x} = x - \frac{\sin x - 1/2}{\cos x} \qquad \{11\}$$

The computations are shown in Table 2.

Table 2

x	$f(x)$	$f'(x)$	Dx	new x
0	-0,5	1	0,5	0,5
0,5	-0,02057	0,877583	0,023444	0,523444
0,523444	-0,00013	0,866103	0,000154	0,523599
0,523599	-6E-09	0,866025	6,87E-09	0,523599
0,523599	0	0,866025	0	0,523599

Illustration 2-D

Solve

$$x + \arctan x = 1 \qquad \{12\}$$

from $x = 0$.

Resolution Drive Eq. {12} to the convenient form.

$$f(x) = x + \arctan x - 1 = 0 \qquad \{13\}$$

[3] The following reference is highly recommended: TOMPSON, Ambler and Barry N. TAYLOR, 2008, "Guide for the use of the International System of units (SI)", (x+78 pp, 2.2 Mb), NIST, Special Publication 811, US Department of Commerce, Gaithesburg, MD (USA). (Available at http://web.ist.utl.pt/mcasquilho/acad/errors/)

$$f'(x) = 1 + \frac{1}{1+x^2}$$

Then,

$$\tilde{x} = x - \frac{x + \arctan x - 1}{1 + 1/(1+x^2)} \qquad \{14\}$$

The function *f* is shown in Fig. 2 and the computations in Table 3.

Table 3

x	*f*(*x*)	*f*'(*x*)	Δx	new *x*
0	-1	2	0,5	0,5
0,5	-0,0363	1,8	0,020196	0,520196
0,520196	-0,00013	1,787027	7,32E-05	0,520269
0,52027	-1,7E-09	1,78698	9,67E-10	0,520269
0,520269	0	1,78698	0	0,520269

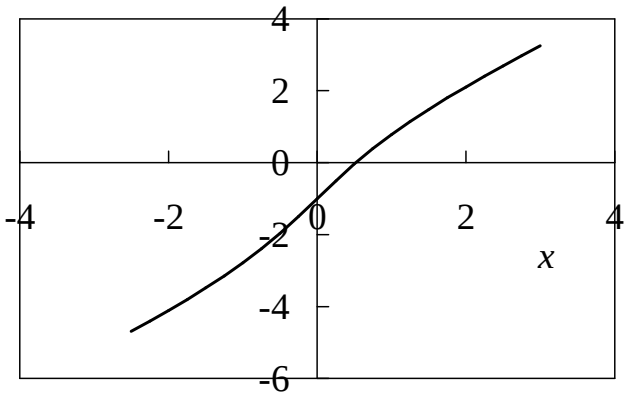

Fig. 2

Illustration 2-E

Solve

$$\Phi(z) = 0.25 \qquad \{15\}$$

from *z* = 0. This will be useful in the *simulation* of a Gaussian variable.

Resolution Remember that Φ is the usual notation for the Gaussian integral,

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\left(-\frac{1}{2}x^2\right) \mathrm{d}\,x \qquad \{16\}$$

Applying the algorithm, with the right-hand side constant of Eq. {15} denoted by *P*, *P* = 0.25, it is

$$f(z) = \Phi(z) - P$$

$$f'(z) = \phi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}z^2\right)$$

Then,

$$z^* = z - \frac{\Phi(z) - P}{\phi(z)} \qquad \{17\}$$

(Blank page)