we have, after cancellation,

$$\frac{v}{l} = \tanh \frac{2x' - d}{2c}$$

which, when solved for x'/c, yields

$$\frac{x'}{c} = \frac{d}{2c} + \tanh^{-1} \frac{v}{l}. \tag{6}$$

Substituting this into (5) gives

$$l = c \sinh\left(\frac{d}{2c} + \tanh^{-1} \frac{v}{l} \right) + c \sinh\left(\frac{d}{2c} - \tanh^{-1} \frac{v}{l} \right), \tag{7}$$

which is the equation to use for determining c when the unknown variable is g.

From Figure 1, $y' = c + g$. From (3) we get

$$c + g = c \cosh \frac{x'}{c}, \tag{8}$$

so, knowing c, g can be determined.

For example, suppose that we wish to find the sag below the upper support of a cable 110′ long suspended between supports that are separated by 100′ horizontally and 20′ vertically. Then from (7),

$$110 = c \sinh\left(\frac{100}{2c} + \tanh^{-1} \frac{20}{110} \right) + c \sinh\left(\frac{100}{2c} - \tanh^{-1} \frac{20}{110} \right).$$

Here is where the calculator comes into play. Using its ability to solve equations, or to graph them, we get that $c = 72.288$. Substitution into (6) gives $x' = 63.291$ and using this in (8) gives $g = 29.523$ feet.

Similarly, equations may be derived for dealing with the cases where the unknown variable is v, d, or l.

———o———

Second Order Iterations

Joseph J. Roseman (roseman@math.tau.ac.il) and Gideon Zwas (zwas@math.tau.ac.il), Tel Aviv University, Tel Aviv 69978, Israel

Introduction

In calculus courses, the concept of iterative computation is introduced to students as they learn to solve problems with the aid of calculators and computers.

In such a computation one starts with an initial approximation x_0 and then iterates an approximative procedure which gradually produces improved approximations of the result. Such iterative processes are similar to parking a car between two parked cars, tuning a guitar, writing a poem, debugging a computer program, fitting a tailored suit to a customer, etc.

The criterion for terminating the iterations is usually met when an appropriate error bound (and therefore the error) is found to be sufficiently small, thereby assuring the desired accuracy.

Our purpose is to illustrate iterative computations by means of an unusual but enlightening example which we derive by **elementary methods**. After learning some additional calculus one can also derive these results via Newton's iterative method.

Initialization

As our example, we shall now describe "iterative division," a nonstandard technique for the computation of division. We emphasize that our real purpose is not to replace division as we know it (which the better "robots" perform by means of their hardware devices), but rather to illustrate the concept of iteration through an example.

Let us assume that we have a robot which is capable of adding, subtracting, and multiplying numbers that are stored in its memory as $(p \cdot 2^q)$, where $1 \le |p| < 2$, and q is an appropriate integer (just as we humans use $c \cdot 10^j$, where $1 \le |c| < 10$ and j is an integer). This unusual form reflects the fact that inside the computer the calculations are done in binary form. However, we are using a mixed form in which p is a decimal. For example, the ordinary real number 7 would be understood by our robot as 1.75×2^2. Our aim is to "teach" this robot to divide and for simplicity, let us assume that $p > 0$. Now, dividing by $a = p \cdot 2^q$ is equivalent to multiplying by $1/a = (1/p) \cdot 2^{-q}$, where $1/2 < 1/p \le 1$, and therefore we shall concentrate on "teaching" the robot to find $1/p$ to its complete accuracy. (Adding $-q$ to the exponent of 2 presents no problem to our robot.)

Since $1/2 < 1/p \le 1$, we can choose as a simple initial value $x_0 = 0.75$, thus being sure that

$$\left| x_0 - \frac{1}{p} \right| \le \frac{1}{4} = \lambda. \tag{1}$$

The parameter λ is an indication of the initial value's accuracy.

Quadratic Behavior

What are we really after? We want a formula of the form $x_{n+1} = g(x_n)$ such that if we start with x_0, we get x_1 and from x_1 we get x_2, etc. and such that the gap $|x_n - \frac{1}{p}|$ decreases towards zero as n gets large.

Now, if there exist a positive integer r and a positive constant K such that the inequality

$$\left| x_{n+1} - \frac{1}{p} \right| = \left| g(x_n) - \frac{1}{p} \right| \le K \left| x_n - \frac{1}{p} \right|^r \tag{2}$$

holds, then the iterative process described by $x_{n+1} = g(x_n)$ is said to be of order r. For certain values of K and r, (2) indicates how the iterations approach the value $1/p$.

If $r = 1$, we have iterations with linear behavior and it follows from (2) that we need $K < 1$ in order to get an error decrease. We see later that the rate of error decrease in the linear case is very slow and inefficient and we try for something better. Namely, we will look for a second order iterative formula (quadratic behavior) and, if successful, even a third order formula (cubic behavior).

The problem which we now face is the following: Given a number p, $1 \le p < 2$, we seek an iterative formula $x_{n+1} = g(x_n)$ for which we can find a constant K

such that

$$\left|x_{n+1}-\frac{1}{p}\right| \le K\left|x_n-\frac{1}{p}\right|^2. \tag{3}$$

Suppose we have found such a $g(x_n)$ and that we have an initial value x_0 which satisfies

$$\left|x_0-\frac{1}{p}\right| \le \lambda,$$

what will the error bound look like after n iterations?

Using (3), we can write

$$\begin{aligned}
\left|x_1-\frac{1}{p}\right| &\le K\left|x_0-\frac{1}{p}\right|^2 \le K\lambda^2,\\
\left|x_2-\frac{1}{p}\right| &\le K\left|x_1-\frac{1}{p}\right|^2 \le K(K\lambda^2)^2 = K^3\lambda^4,\\
\left|x_3-\frac{1}{p}\right| &\le K\left|x_2-\frac{1}{p}\right|^2 \le K(K^3\lambda^4)^2 = K^7\lambda^8,\\
&\cdots\\
\left|x_n-\frac{1}{p}\right| &\le K^{2^n-1}\lambda^{2^n} = \frac{1}{K}(K\lambda)^{2^n}.
\end{aligned} \tag{4}$$

This last step should be checked and justified by mathematical induction.

We see that the error decay parameter is $K\lambda$, that we need $K\lambda < 1$ to ensure that the error decreases, and that the exponentiation by 2^n determines the speed of this decay. For linear behavior, the error bound would only be λK^n for $K < 1$, giving a much slower error decay than that of (4). Now, can we find $g(x_n)$ with the quadratic behavior shown by the inequalities (4)? Our suggestion is to ask the following question. Is there an expression E such that the equality

$$x_{n+1}-\frac{1}{p} = E\cdot\left(x_n-\frac{1}{p}\right)^2 \tag{5}$$

will give rise to an acceptable iterative formula $x_{n+1} = g(x_n)$? Rewriting (5) in the form

$$x_{n+1} = Ex_n^2 - 2\frac{E}{p}x_n + \frac{E+p}{p^2} \tag{6}$$

shows us that in order to get an acceptable formula, we need an E that will rid us completely of divisions. After all, our robot still does not know how to divide. Examining (6) we see that $E = -p$ is a good choice that will leave us with the iterative division-free formula

$$x_{n+1} = x_n(2-px_n). \tag{7}$$

This formula can also be obtained by applying Newton's method to $f(x) = p - 1/x$. Every calculation with (7) uses two multiplications and one subtraction. Substituting $E = -p$ into (5) and taking absolute values yields

$$\left|x_{n+1}-\frac{1}{p}\right| = |p|\left|x_n-\frac{1}{p}\right|^2.$$

Since $1 \leq p < 2$, it follows that

$$\left| x_{n+1} - \frac{1}{p} \right| \leq 2 \left| x_n - \frac{1}{p} \right|^2,$$

thus achieving the predesigned quadratic behavior of (3) and an error bound (cf. (4))

$$\left| x_n - \frac{1}{p} \right| \leq \frac{1}{2}(2\lambda)^{2^n}. \tag{8}$$

Suppose we want $1/p$ up to nine places after the decimal point, i.e. an error less than $(\frac{1}{2})10^{-9}$. Then with $\lambda = \frac{1}{4}$, it follows from (1) that we need only $n = 5$ iterations. We see that with quadratic behavior only a few iterations are needed.

From the error bound (8) we see that further improvement of the initial value will pay off, since λ is raised to the power of 2^n. An improved initial value can be obtained from a linear approximation of the hyperbola $y = 1/p$, $1 \leq p \leq 2$. This hyperbola lies below the secant line joining points $(1,1)$ and $(2,1/2)$ and above the tangent line at $(\sqrt{2}, 1/\sqrt{2})$ having the same slope $-1/2$. Using the line halfway between these two leads to the initial estimate x_0 given by

$$x_0 = -0.5p + (0.75 + 0.5\sqrt{2})$$

for which one can obtain a value of $\lambda = 1/23$. This would further accelerate the convergence. The combination of such an improved initialization together with the second order iterations will display a sufficiently effective rate of convergence.

Third Order Iterations

A challenge for students is to examine the possibility of third order iterations having cubic behavior. It turns out that such an iterative division-free formula does indeed exist. An analysis similar to the one that gave rise to (7) leads to

$$x_{n+1} = x_n[px_n(px_n - 3) + 3],$$

together with the inequality

$$\left| x_{n+1} - \frac{1}{p} \right| \leq 4 \left| x_n - \frac{1}{p} \right|^3.$$

This shows cubic behavior with $K = 4$, and the error bound

$$\left| x_n - \frac{1}{p} \right| \leq \frac{1}{\sqrt{K}}(\sqrt{K}\lambda)^{3^n} = \frac{1}{2}(2\lambda)^{3^n}.$$

The number of correct digits now triples with each additional iteration and therefore even fewer steps are needed to reach a desired accuracy. This saving does however not compensate for the increased computational work at each iterative step, which requires about twice the computations than that of a second order iteration. In the total efficiency competition, third order iterations lag behind second order ones. For iterations of order four or more, the efficiency is even worse.

Reference

1. S. Breuer and G. Zwas, "Computer root extraction by a priori design," *Computers and Education*, **8** (1984) 305.