[MATH-13] Numerical solutions of equations

Miguel-Angel Canela

Associate Professor, IESE Business School

Iterative numerical methods

It may sound strange to you, but, for many simple mathematical problems, there is no method for finding a solution, even when one has proved its existence. We must content ourselves with approximations. This is illustrated in this lecture with various methods for (approximately) solving equations.

A **numerical method** is any process which leads to an approximate solution. Common numerical methods for solving equations are **iterative**. An iterative method produces a sequence of approximations x_n ($n = 1, 2, 3, \ldots$), whose limit is an (exact) solution of the equation. The core of the iterative method is a **recurrence formula**, which is a mathematical formula that gives x_n in terms of x_0, …, x_{n-1} (frequently x_n in terms of x_{n-1}). Taking as x_0 an **initial estimate** of the solution, we apply the recurrence formula to obtain x_2 from x_1, again to obtain x_3 from x_1 and x_2, etc. Depending on the complexity of the recurrence formula (it may be very involved in some regression methods actually used in Econometrics, based on maximum likelihood estimation), your computer may spend more or less time doing that.

It is common to distinguish between an iterative method and the algorithm for its implementation, which is a specific set of instructions that include a stopping rule. Thus, an iterative algorithm has a maximum number of iterations, and a tolerance limit for the proximity between the results of successive iterations. When the successive results are very close, we say that the iterative process converges.

Solving equations

Let me consider the problem that is the objective of this lecture, solving an equation. I write the equation as $f(x) = 0$, assuming that f is a function given by a mathematical expression that I can manage without pain. When f is increasing or decreasing, everything is simpler, because there can be at most one solution. If there are several solutions, it is crucial to start the oterative process from an initial estimate that is close to the desired solution.

The first method is very simple, but so slow that it is used only for producing first approximations. It is based on a classical theorem of Calculus, the **Bolzano theorem**.

Theorem. Let $f : [a, b] \longrightarrow \mathbb{R}$ be a continuous function (no jumps), with $f(a) > 0$ and $f(b) < 0$ (or conversely). There exists $a < x < b$ such that $f(x) = 0$.

For functions whose graph is a curve that can be plotted, the theorem is obvious. The application in order to approximate the solution of an equation is straightforward: for any interval where $f(x)$ changes the sign, there must be a solution within it. This is the basis of an elementary method called the **bisection method**. First we find an interval that contains a solution, then we evaluate the function at the midpoint, look at the sign, and consequently pick the left or the right half-interval. Next, we $f(x)$ at the midpoint of the new interval, and divide by two again. Et cetera. The procedure is very slow, but it is very easy to write a program for implementing it in the computer. Let show how this works with an example.

Example 1. Suppose that we are interested in solving the equation

$$f(x) = e^{-x} - x = 0.$$

Then, $f'(x) = -e^{-x} - 1 < 0$, so that f is a decreasing function, and there is, at most, one solution. I start with the interval $[0, 1]$.

$$f(0) = 1, \qquad f(1) = -0.6321.$$

This tells us that there is a solution in the interval $0 < x < 1$. Now, at the midpoint, $f(0.5) = 0.1065$, so that the solution satisfies $0.5 < x < 1$. Next, $f(0.75) = -0.2776$, and I pick the right half, $0.5 < x < 0.75$. Next, $f(0.675) = -0.0897$, so the new interval is $0.5 < x < 0.675$. I stop here. □

The Newton-Raphson method

A more powerful approach, called the **Newton-Raphson method**, starts from an initial estimate x_0 (that can be be found searching for a change of sign) and uses the iterative formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}\,.$$

This method converges very quickly if the initial estimate is chosen near the solution. But the algorithm needs to take derivatives in the symbolic way.

Example 1 (continuation). Back to the equation $e^{-x} - x = 0$. I know that the solution lies between 0 and 1, I can reasonably start from $x_0 = 0.5$. Applying successively the iterative formula, which here is

$$x_{n+1} = x_n + \frac{e^{-x_n} - x_n}{e^{-x_n} + 1}\,,$$

we get the results of Table 1. It seems reasonable to stop after four steps, since the solution is already identified up to the sixth digit. □

TABLE 1. Newton-Raphson method (Example 1)

n	x_n	$f(x_n)$	$f'(x_n)$	$f(x_n)/f'(x_n)$
0	0.5	0.106531	–1.606531	–0.066311
1	0.566311	0.001305	-1.567616	–0.000832
2	0.567143	1.96E-07	–1.567143	–1.25E-07
3	0.567143	4.44E-15	–1.567143	–2.83E-15
4	0.567143	–1.11E-16	–1.567143	7.08E-17

The secant method

An alternative to the preceding iterative procedure is the **secant method**, which starts from two initial estimates x_0 and x_1 and uses the iterative formula

$$x_{n+1} = x_n - f(x_n)\,\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\,.$$

For the secant method, the convergence is slower than for the Newton-Raphson method, but it has the advantage that one does not need an expression for the derivative $f'(x)$. This makes programming easier.

Example 1 (continuation). With the secant method, taking $x_0 = 0$ and $x_1 = 1$, we get the same solution after five iterations (Table 2). □

TABLE 2. Secant method (Example 1)

n	x_n	$f(x_n)$
0	0	1
1	1	–0.63212056
2	0.61269984	–0.07081395
3	0.56383839	0.00518235
4	0.56717036	–4.2419E-05
5	0.56714331	–2.538E-08
6	0.56714329	1.2423E-13
7	0.56714329	0

Homework

A. Prove that the equation $x^3 + x^2 + x - 1 = 0$ has a unique real solution and find it (up to six digits) using the Newton-Raphson method.