2

A Few Concepts from Numerical Analysis

A systematic treatment of numerical methods is provided in conventional courses and textbooks on numerical analysis. But a few very common issues, that emerge in similar form in many numerical methods, are discussed here.

2.1 Root finding: fast and unreliable

We consider the problem of solving a single equation, where a function of one variable equals a constant. Suppose a function $f(x)$ is given and we want to find its root(s) x^*, such that $f(x^*) = 0$.

A popular method is that of Newton. The tangent at any point can be used to guess the location of the root. Since by Taylor expansion $f(x^*) = f(x) + f'(x)(x^* - x) + O(x^* - x)^2$, the root can be estimated as $x^* \approx x - f(x)/f'(x)$ when x is close to x^*. The procedure is applied iteratively: $x_{n+1} = x_n - f(x_n)/f'(x_n)$. For example, it is possible to solve $\sin(3x) = x$ in this way, by finding the roots of $f(x) = \sin(3x) - x$. Starting with $x_0 = 1$, the procedure produces the numbers shown in column 2 of table 2-I. The sequence quickly approaches a root.

Newton's method is so fast, computers may use it internally to calculate the square root, $x = \sqrt{y}$, with $f(x) = x^2 - y$. This only requires substractions, divisions, and multiplications. For example, the square root of 2 can be quickly calculated using only divisions and subtractions with $f(x) = x^2 - 2$. With a starting value of $x_0 = 2$, we obtain the numbers shown in Table 2-II. Eight significant digits are achieved after only four iterations.

n	x_n	
	$x_0 = 1$	$x_0 = 2$
0	1	2
1	0.7836...	3.212...
2	0.7602...	2.342...
3	0.7596...	3.719...
4	0.7596...	-5.389...

Table 2-I: *Newton's method applied to* $\sin(3x) - x = 0$ *with two different starting values.*

n	x_n
0	2
1	1.5
2	1.41666...
3	1.414216...
4	1.4142136...
$\sqrt{2}$	1.4142136...

Table 2-II: *Newton method used to obtain a square root*

But Newton's method can easily fail to find a root. Returning to the previous example, with $x_0 = 2$ the iteration never converges, as indicated in the last column of table 2-I.

Robustness may be preferable to speed. Is there a method that is certain to find a root? The simplest and most robust method is bisection, which follows the "divide-and-conquer" strategy. Suppose we start with two x-values where the function $f(x)$ has opposite signs. Any continuous function *must* have a root between these two values. We then evaluate the function halfway between the two endpoints and check whether it is positive or negative there. This restricts the root to that half of the interval on whose ends the function has opposite signs. Table 2-III shows an example. With the bisection method the accuracy is only doubled at each step, but the root is found for certain.

There are more methods for finding roots than the two just mentioned. Each method has its advantages and disadvantages. Bisection is

n	x_{lower}	x_{upper}
0	0.1	2
1	0.1	1.05
2	0.575	1.05
3	0.575	0.8125
4	0.6938...	0.8125
5	0.7531...	0.8125
⋮	⋮	⋮
15	0.7596...	0.7597...
16	0.7596...	0.7596...

Table 2-III: *Bisection method applied to* $\sin(3x) - x = 0$.

most general but is awfully slow. Newton's method is less general but much faster. Such a trade-off between generality and efficiency is often inevitable. This is so because efficiency is often achieved by exploiting a specific property of a system. For example, Newton's method makes use of the differentiability of the function; the bisection method does not and works equally well for functions that cannot be differentiated.

The bisection method is guaranteed to succeed only if it brackets a root to begin with. There is no general method to find appropriate starting values, nor do we generally know how many roots there are. For example, a function can reach zero without changing sign; our criterion for bracketing a root does not work in this case. In principle, a continuous function can in any interval drop rapidly, cross zero, and then increase again, making it impossible to exclude the existence of roots.

The problem becomes even more severe for finding roots in more than one variable, say under the simultaneous conditions $g(x, y) = 0, f(x, y) = 0$. Newton's method can be extended to several variables, but bisection cannot. How could one be sure all zeros are found in this vast space? There is no method that is guaranteed to find all roots. This is not a deficiency of the numerical methods, but is due to the intrinsic nature of the problem. Unless a good, educated initial guess can be made, finding roots in more than a few variables may be fundamentally and practically impossible. (This is in stark contrast to solving a system of *linear*

equations, which is an easy numerical problem.)

Root finding can be a numerically difficult problem, because there is no method that always succeeds.

2.2 Error propagation and numerical instabilities

Numerical problems can be difficult for other reasons too.

When small errors in the input data, of whatever origin, can lead to large errors in the resulting output data, the problem is called "numerically badly-conditioned" or if the situation is especially bad, "numerically ill-conditioned." An example is solving the system of linear equations

$$x - y + z = 1, \quad -x + 3y + z = 1, \quad y + z = 2.$$

Suppose there is an error ϵ in one of the coefficients such that the last equation becomes $(1 + \epsilon)y + z = 2$. The solution to these equations is easily worked out as $x = 4/\epsilon$, $y = 1/\epsilon$, $z = 1 - 1/\epsilon$. Hence, the result depends extremely strongly on the error ϵ. The reason is that for $\epsilon = 0$ the system of equations is linearly dependent: the sum of the left-hand sides of the first two equations is twice that of the third equation. The right-hand side does not follow the same superposition. Consequently the unperturbed equations ($\epsilon = 0$) have no solution. The situation can be visualized geometrically. Each of the equations describes an infinite plane in a three-dimensional space and the point at which they intersect represents the solution. For small ϵ, the planes intersect at a small angle and the intersection moves to infinity as ϵ goes to zero. This is a property of the problem itself, not the method used to solve it. No matter what method is utilized to determine the solution, the uncertainty in the input data will lead to an uncertainty in the output data. If a linear system of equations is linearly dependent, it is an ill-conditioned problem.

The theme of error propagation has many facets. Errors introduced during the calculation, namely by roundoff, can also become critical, in particular when errors are amplified not only once, but repeatedly. Let me give one such example for the successive propagation of inaccuracies.

Consider the difference equation $3y_{n+1} = 7y_n - 2y_{n-1}$ with the two starting values $y_0 = 1$ and $y_1 = 1/3$. The analytic solution to this

equation is $y_n = 1/3^n$. If we iterate numerically with initial values $y_0 = 1$ and $y_1 = 0.3333$ (which approximates $1/3$), then column 3 of table 2-IV shows what happens. For comparison, the second column in the table shows the numerical value of the exact solution. The numerical iteration breaks down after a few steps.

n	y_n		
	$y_n = 1/3^n$	$y_1 = 0.3333$	$y_1 = 1./3.$
0	1	1	1
1	0.333333	0.3333	0.333333
2	0.111111	0.111033	0.111111
3	0.037037	0.0368777	0.0370372
4	0.0123457	0.0120257	0.0123459
5	0.00411523	0.00347489	0.00411569
6	0.00137174	9.09399E-05	0.00137267
7	0.000457247	-0.0021044	0.000459095
8	0.000152416	-0.0049709	0.00015611
9	5.08053E-05	-0.0101958	5.81938E-05
10	1.69351E-05	-0.0204763	3.17123E-05
11	5.64503E-06	-0.0409809	3.51994E-05
12	1.88168E-06	-0.0819712	6.09904E-05
13	6.27225E-07	-0.163946	0.000118845
14	2.09075E-07	-0.327892	0.000236644

Table 2-IV: *Numerical solution of the difference equation* $3y_{n+1} = 7y_n - 2y_{n-1}$ *with initial error (third column) and roundoff errors (last column) compared to the exact numerical values (second column).*

The reason for the rapid accumulation of errors can be understood from the analytic solution of the difference equation with general initial values: $y_n = c_1(1/3)^n + c_2 2^n$. The initial conditions for the above example are such that $c_1 = 1$ and $c_2 = 0$, so that the growing branch of the solution vanishes, but any error seeds the exponentially growing contribution. Indeed, the last few entries in the third column of table 2-IV double at every iteration; they are dominated by the 2^n contribution.

Even if y_1 is assigned exactly $1/3$ in the computer program, using

single-precision numbers, the roundoff errors spoil the solution (last column in table 2-IV). This iteration is "numerically unstable"; the numerical solution quickly grows away from the true solution. Numerical instabilities are due to the method rather than the mathematical nature of the equation being solved. (Contrary to the previous example, where the ultimate source of error was the coefficients in the equations themselves.) For the same problem one method might be unstable while another method is stable.

In summary, we have encountered a number of issues that come up in numerical computations. There is commonly a tradeoff between robustness and speed. There may be no algorithm that succeeds for certain, as for example root finding in one variable (in principle) and root finding in many variables (in practice). The propagation of errors in input data or due to roundoff can lead to difficulties. Solutions can be very sensitive to uncertainties in the input data, e.g. a system of linear equations that is nearly degenerate. This sensitivity can be assessed, but that involves generally more work than obtaining the solution. Difficulties may be intrinsic to the problem or intrinsic to the method. Another concept, one barely touched on here, is computational efficiency. It is discussed in chapters 8 to 10.

Recommended Reading: A practically oriented classic is Press, Teukolsky, Vetterling & Flannery, *Numerical Recipes*. This voluminous book describes a broad and selective collection of methods and provides insightful lessons in numerical analysis; see www.nr.com for further information.

Brainteaser: An illustrative example of a function that suddenly crosses zero is $f(x) = 3\pi^4 x^2 + \ln[(x-\pi)^2]$, which dips below zero only within a tiny interval. Although it is easy to show that $f(x)$ has two roots, graphing the function with any software will not reveal the sign change.