

Variations on a Theme of Newton

ROBERT M. CORLESS

University of Western Ontario
London, Ontario
Canada N6A 5B7

Introduction

We use a particularly simple example function¹, and the computer algebra system Maple, to try to learn something about Newton's method. The discussion here presumes only a minimal amount of calculus—including the standard introduction to Newton's method, such as is found in [2, Sec. 2.10]—and some algebraic fluency. This discussion, though aimed at undergraduate students, contains surprises (perhaps even for instructors), items not found in the usual calculus course, and pointers to many more such items. The intention is to provoke or reinforce an interest in pure and applied mathematics. If this works, *everyone* will take something new away.

Newton's method

Newton's method is for approximately solving nonlinear equations $f(x) = 0$. Applied mathematics problems usually lead to nonlinear equations—we cannot rely on everything being linear. Some examples of applied problems requiring Newton's method or an equivalent are:

- so-called “implicit” numerical methods for the solution of ordinary differential equations.
- practically any engineering design problem, where instead of being asked to calculate the behavior of a machine or system as given, you are asked to calculate the design parameters that will make the system behave in a certain desired way. For example, many problems in robotic control fall into this category.
- computer-aided design uses piecewise polynomials to model physical objects. Calculating their intersection points requires the solution of systems of polynomial equations. Even if initial approximations to the solutions are arrived at by other means, Newton's method can be used to “polish” the roots.

The basic idea behind Newton's method is that if you can't solve $f(x) = 0$ for x , replace f with a simpler function F , namely, the best linear approximation to $f(x)$ near some initial guess point x_0 . This approximation is $F(x) = f(x_0) + f'(x_0)(x - x_0)$, and we *can* solve $F(x) = 0$ to get $x_1 = x_0 - f(x_0)/f'(x_0)$, provided $f'(x_0) \neq 0$. Repeating this with the new approximation x_1 to get x_2 and so on gives us the iterative formula

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

¹Our example function $f(x) = x^2 - a$ is indeed particularly simple, and this is important: if it were not so simple, we wouldn't be able to go anywhere near as far as we do. Hold on to your seat!

We will explore this formula with an extremely simple nonlinear function, namely $f(x) = x^2 - a$, in order to learn something about Newton's method (and some computer tools). It is clear that the zeros of $f(x) = x^2 - a$ are just $x = \sqrt{a}$ and $x = -\sqrt{a}$, so Newton's method is not really required for this problem. Even worse, we are later going to specify $a = 1$, so we will be using Newton's method to find the square root of 1! Our iteration is, for general a ,

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} \quad (1)$$

or, mathematically equivalent but slightly less numerically stable,

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right).$$

A Maple program The following program, written in the computer algebra language Maple (see [1], for example, for an accelerated introduction to Maple), will be used to compute iterates of Newton's method for the rest of this discussion. The routine `normal` just simplifies expressions.

```
Newton := proc(a, x0, n) local xn;
  xn := x0;
  to n do
    xn := normal(xn - (xn^2 - a) / (2*xn))
  od
end;
```

Numerical tests If we choose $a = 2$, then our function is $f(x) = x^2 - 2$ and we are looking for $\sqrt{2}$. Choosing an initial guess of $x_0 = 1$, the program `Newton` produces Table 1.

TABLE 1 Newton iterates of $f(x) = x^2 - 2$.

| n | x_n | error |
|-----|---------------------------|----------------------|
| 0 | 1 | -1.0 |
| 1 | 3/2 | $2.5 \cdot 10^{-1}$ |
| 2 | 17/12 | $6.0 \cdot 10^{-3}$ |
| 3 | 577/408 | $6.0 \cdot 10^{-6}$ |
| 4 | 665857/470832 | $4.5 \cdot 10^{-12}$ |
| 5 | 886731088897/627013566048 | $2.5 \cdot 10^{-24}$ |

REMARKS

1. The error reported in the above table is the so-called "residual" error $r_n = f(x_n)$. If r_n is zero, then of course x_n is a root; if r_n is "small," then, in some sense, x_n is "close" to a root. This type of measure of accuracy is always available, even when the exact answer is not known. For "well-conditioned" problems it gives the same information as the difference between the approximate answer and the true answer; this problem is well-conditioned because $x_n - \sqrt{a} = (x_n^2 - a)/(x_n + \sqrt{a}) \approx r_n/(2\sqrt{a})$ and so the *relative* error here ($a = 2$) is about $(x_n - \sqrt{a})/\sqrt{a} \approx r_n/4$.
2. Exact arithmetic *costs a lot*. We notice that the *length* of the answer approximately doubles each time; a quick calculation shows that the answer after 30

iterations would take a few gigabytes of memory to store. This is why people instead use arithmetic with a fixed number of decimals (*i.e.*, floating-point).

3. We can simplify our problem by the *nondimensionalization*² $u_n = x_n/\sqrt{a}$, at least for the purpose of understanding what is happening. Of course, for actual calculations we can't nondimensionalize by \sqrt{a} which we don't know. If we use this conceptual scaling, then the Newton iteration becomes

$$u_{n+1} = \frac{1}{2} \left(u_n + \frac{1}{u_n} \right).$$

This is exactly the same iteration but with $a = 1$. Thus the scaled iteration uses Newton's method to compute the square root of 1. But the relative error in x_n is $(x_n - \sqrt{a})/\sqrt{a} = u_n - 1$ and so this iteration really does tell us something about Newton's method, and we will keep it in mind. It is easy to see that if $u_n = 1 + e_n$ where e_n represents the error after n iterations, then

$$e_{n+1} = \frac{e_n^2}{2(1 + e_n)} \approx \frac{1}{2} e_n^2.$$

This is called *quadratic convergence*. Using this formula shows that after about 30 iterations we will have about 1 billion digits of \sqrt{a} correct, if we start with roughly one correct digit.

4. If we convert these rational numbers to "continued fraction form" (using the Maple routine `convert(17/12, confrac)`) where a *continued fraction* is something of the form

$$n_0 + \frac{1}{n_1 + \frac{1}{n_2 + \frac{1}{\ddots}}}$$

we see the quite remarkable patterns

$$\begin{aligned} 1 &= 1 \\ 3/2 &= 1 + \frac{1}{2} \\ 17/12 &= 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2}}} \\ 577/408 &= 1 + \frac{1}{2 + \frac{1}{\ddots + \frac{1}{2}}} \end{aligned}$$

where the length of the continued fraction is 2^n , and every entry is 2. This is the beginning of an interesting foray into number theory.

²If a has units, say square meters, this scaling removes them.

Symbolic initial guess If the program `Newton` given earlier had been written in C or FORTRAN, then calling it with a symbol (say g) for the initial guess would generate an error message. But here,

```
> Newton(1, g, 1);
```

in Maple, returns $(g^2 + 1)/(2g)$. We can ask Maple to continue, giving the results in Table 2.

TABLE 2 Newton iterates for $x^2 - 1$ with a symbolic initial guess, g .

| n | x_n |
|-----|--|
| 0 | g |
| 1 | $\frac{g^2 + 1}{2g}$ |
| 2 | $\frac{1}{4} \frac{g^4 + 6g^2 + 1}{g(g^2 + 1)}$ |
| 3 | $\frac{1}{8} \frac{g^8 + 28g^6 + 70g^4 + 28g^2 + 1}{g(g^4 + 6g^2 + 1)(g^2 + 1)}$ |

In FIGURE 1 we plot the first few results from `Newton`. We see that these rational functions are trying to approximate a step function; as n increases, we see clear evidence that these functions converge. The moral of this section is that the error message that FORTRAN would have given us would have concealed an insight, namely that the result of n iterations of Newton's method is a rational function of the initial guess g . Further, we have learned that this rational function looks (for large n) rather like a step function with heights $\pm \sqrt{a}$. Note that the graph in FIGURE 1 works for all a because the axes are scaled—the horizontal axis is the g/\sqrt{a} axis and the vertical axis is the x_n/\sqrt{a} axis.

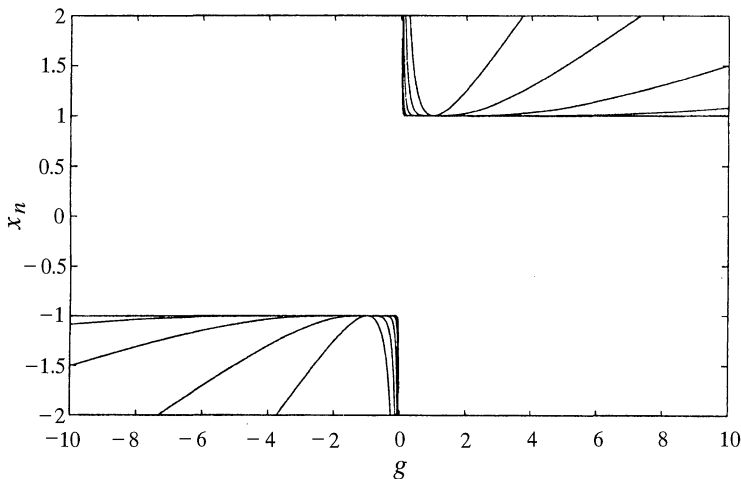


FIGURE 1

Newton iterates with a symbolic initial guess, plotted together. As n increases we must have $x_n/\sqrt{a} \rightarrow \pm 1$, and we can see that the convergence is rapid near $g/\sqrt{a} = \pm 1$, as we expect.

Symbolic a Now let us choose instead $x_0 = 1$ (we will discuss this choice of initial guess in a moment) and look at the results from Maple if we input a symbolic a to the program. The first few of these are presented in Table 3.

TABLE 3 Rational approximations obtained by using a symbolic a .

| n | x_n |
|-----|---|
| 0 | 1 |
| 1 | $\frac{1}{2} + \frac{1}{2}a$ |
| 2 | $\frac{1}{4} \frac{1 + 6a + a^2}{1 + a}$ |
| 3 | $\frac{1}{8} \frac{1 + 28a + 70a^2 + 28a^3 + a^4}{(1 + 6a + a^2)(1 + a)}$ |

When we plot these³, we get a sequence of rational (in a) approximations to \sqrt{a} , as is quite evident in FIGURE 2.

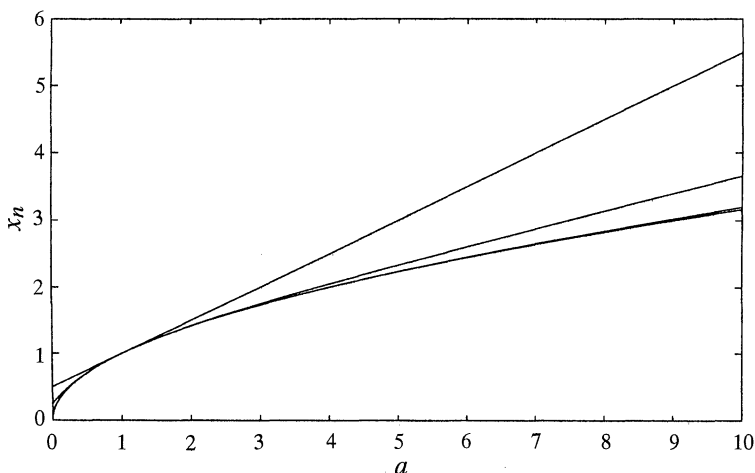


FIGURE 2

The first few iterates of Newton's method on $f(x) = x^2 - a$ with symbolic a give quite good rational approximations to \sqrt{a} .

REMARKS

1. Nondimensionalization shows that choosing $x_0 = 1$ is perfectly general. Put $x_n = x_0 v_n$ in equation 1, and simplify to get

$$v_{n+1} = \frac{1}{2} \left(v_n + \frac{a/x_0^2}{v_n} \right).$$

³Both FIGURE 1 and FIGURE 2 were actually prepared using Matlab, not Maple, because Matlab plots look slightly nicer; moreover, the graphs were generated by giving a *vector* of g values and a *vector* of a values to a Matlab implementation of Newton's method, much like the Maple symbolic version.

This is just the iteration for finding the square root of a/x_0^2 . Therefore, the graph in FIGURE 2 differs from the graph of the approximations we would get with some other initial guess (say $x_0 = 2$) *only in the scale of the axes*—in particular the shape of the curves remains the same. If we label the y -axis with x_0 where 1 is now, and likewise $2x_0$ for 2 and so on, and label the x -axis with x_0^2 where 1 is now, etc., then FIGURE 2 represents the first few iterates of the general case. That is, all the curves with general initial guess *collapse onto the same curve*. This shows the true power of nondimensionalization.

2. We can replace `normal` in the routine `Newton` with a call to Maple's `series` command, and execute Newton's method in the domain of power series. Quadratic convergence in this domain means that the number of correct terms in the power series doubles each time.
3. We can show with Maple that the error in our rational functions of a above are proportional to $(a - 1)^{2^k}$; for example, after three iterations the error is

$$f_3(a)^2 - a = \frac{1}{64} \frac{(a - 1)^8}{(1 + 6a + a^2)^2(1 + a)^2}.$$

As before the difference between $f_3(a)$ and \sqrt{a} will be about $1/(2\sqrt{a})$ times this.

4. We can convert the rational approximations in Table 3 to continued fraction form; indeed these approximations are one step towards *approximation theory* which underlies much of scientific computing.
5. Again FORTRAN would give us an error message if we tried this in that language. We begin to suspect that whenever a language gives us an error message, there is something to learn.

Chaotic dynamics

Now we choose $a = -1$ and see what happens. We are trying to find an x such that $x^2 + 1 = 0$, and if we start with a real x_0 we are doomed to failure. However, the failure is very interesting.

A few experiments show us that some initial guesses ($x_0 = 0$, $x_0 = 1$, $x_0 = 1 - \sqrt{2}$, etc.) lead to division by zero. We ignore these minor annoyances. A few more experiments show that most initial guesses don't lead (immediately) to division by zero, but rather wander all over the x -axis, without showing any kind of pattern.

Since the x_n appear random in this case, we consider looking at a *frequency distribution* of them. We divide the axis up into bins—the bins are chosen according to a rule given by an advanced theory, namely a rule depending on the theoretical probability density function—and count the number of x_n that appear in each bin. The results appear in Table 4.

To explain the theoretical probability density function would take us to the boundaries of *ergodic theory*, which is a “main artery,” if you will, of statistical mechanics, dynamical systems, and indeed probability theory.

Symbolic n If we call the Maple program not with symbolic a or x_0 but rather with symbolic n , the number of iterations, we get the error message

Error, (in Newton) unable to execute for loop.

TABLE 4 Frequency distribution for x_n where $x_{n+1} = (x_n - 1/x_n)/2$ and $x_0 = 0.4$ (10,000 iterates). The bin boundaries b_k , $0 \leq k \leq 10$ are chosen so that $b_0 = -\infty$ and $\int_{b_{k-1}}^{b_k} 1/(\pi(x^2 + 1)) dx = 1/10$. According to theory, there should be roughly the same number of x_n in each bin.

| k | b_k | number of x_n in (b_{k-1}, b_k) |
|-----|----------|--|
| 1 | -3.0777 | 1001 |
| 2 | -1.3764 | 999 |
| 3 | -0.7265 | 1006 |
| 4 | -0.3249 | 1000 |
| 5 | 0.0000 | 986 |
| 6 | 0.3249 | 1000 |
| 7 | 0.7265 | 1007 |
| 8 | 1.3764 | 980 |
| 9 | 3.0777 | 986 |
| 10 | ∞ | 1035 |

As we have discovered, an error message indicates that we have something to learn. Maple might not be able to do this problem for a symbolic n , *but we can* (in this case). Assume first that $u_0 > 1$ (this corresponds to $x_0 > \sqrt{a}$). Put $u_n = \coth \theta_n$. (The *hyperbolic functions* $\sinh \theta = (\exp(\theta) - \exp(-\theta))/2$, $\cosh \theta = (\exp(\theta) + \exp(-\theta))/2$, $\tanh \theta = \sinh \theta / \cosh \theta$ and so on, are strongly related to the ordinary trig functions.) We have

$$\begin{aligned} \coth \theta_{n+1} = u_{n+1} &= \frac{1}{2} \left(\frac{\cosh \theta_n}{\sinh \theta_n} + \frac{\sinh \theta_n}{\cosh \theta_n} \right) \\ &= \frac{\cosh 2\theta_n}{\sinh 2\theta_n} \\ &= \coth 2\theta_n \end{aligned}$$

where we have used $\cosh^2 \theta + \sinh^2 \theta = \cosh 2\theta$ and $2\sinh \theta \cosh \theta = \sinh 2\theta$ to simplify. Taking \coth^{-1} of both sides, we see $\theta_{n+1} = 2\theta_n$, which is easily solved to get

$$\theta_n = 2^n \theta_0.$$

Therefore $u_n = \coth(2^n \theta_0)$, if $u_0 > 1$.

For the case when $0 < u_0 < 1$, we note that we will immediately have $u_1 = (u_0 + 1/u_0)/2 > 1$ (for example, by elementary calculus we see the minimum of u_1 occurs when $u_0 = 1$). Thereafter the previous analysis applies. The case of $u_0 < 0$ is symmetric to the positive case. So we can say $u_n = \coth 2^{n-1} \theta_1$, regardless of what u_0 is.

Similarly, it is an elementary exercise to show in the complex case, with $a = -1$, that $u_n = \cot(\theta_n)$ gives $\theta_{n+1} = 2\theta_n$ or

$$u_n = \cot(2^n \theta_0).$$

This lays bare all of the chaotic dynamics of this iteration in the complex case. See [3] for more discussion of this case.

REMARKS

1. Now we have the solution for symbolic n , we can answer the question “What do you get if you do half a Newton iteration?” For this problem, we get $u_{3/2} = \coth(\sqrt{2} \theta_1)$ (by definition). This doesn’t have any apparent application, but in more complicated dynamical systems finding such an interpolation is very useful indeed.

2. The *Lyapunov exponent* in the chaotic case is $\ln 2$. The formula (3) also tells us how to find the theoretical probability density function alluded to earlier.
3. No “fractals” appear in this problem, unless it is on the imaginary axis (where the chaos is). However, looking at Newton’s method for solving $f(x) = x^3 - 1 = 0$, we get fractals in \mathbb{C} immediately. See [3], and the other papers in that same issue of the *College Mathematics Journal*.
4. The “asymptotics” of $\coth 2^n \theta_0$ tell us how quickly the iterates approach 1. By Maple,

$$u_n = 1 + 2e^{-1 \cdot 2^n \theta_1} + 2e^{-2 \cdot 2^n \theta_1} + O(e^{-3 \cdot 2^n \theta_1})$$

which tells us everything about how fast u_n approaches 1 (and by extension how fast x_n approaches \sqrt{a}).

Concluding remarks

In this discussion we have stepped outside the normal route to mathematics. By asking just slightly different questions about Newton’s method than is usual in a calculus class—using a very simple example, just trying to understand it better—we have used or discovered links to nondimensionalization, numerical analysis, complexity theory, continued fractions, approximation theory, series algebra, asymptotics, ergodic theory, and dynamical systems (chaos and fractals). One hopes the student will be stimulated to search out other references on these subjects (one might begin with the references in [3], and the other papers in that same issue of the *College Mathematics Journal*).

The discussion in this paper also suggests that it might have been premature to drop Newton’s method (for computing the square root) from the high-school curriculum, as it has been dropped in some districts, merely because calculators can compute square roots with the press of a button. The important thing may not ever have been to compute a square root, but rather to provide a nice introduction to Newton’s method, from which “central trunk” we may move on to other significant areas of modern mathematics.

Probably the most significant concept used in this discussion is nondimensionalization. From a practical viewpoint, it is an invaluable tool in the management of large numbers of variables; from the pure mathematical viewpoint it is an overture to the theory of symmetry, itself a vigorous and powerful branch of modern mathematics.

But even just on its own, Newton’s method is an extremely important and well-studied tool in applied mathematics, used every day for the solution of systems of nonlinear equations. It is surprising how easy it is to find new questions to ask about it.

Acknowledgment. Many of these ideas are due to Charles M. Patton, and I first heard them in his workshop at the 4th International Conference on Technology in Education, in Portland, Oregon, 1991. This paper also benefited from discussions with Peter Poole, David Jeffrey, and Bob Bryan.

REFERENCES

1. Robert M. Corless, *Essential Maple*, Springer-Verlag, New York, NY, 1994.
2. J. Stewart, *Calculus*, Brooks/Cole, Pacific Grove, CA, 1991.
3. Gilbert Strang, A chaotic search for i , *College Math. Journal* 22 (1991), pp. 3–12.