Chaotic Convergence of Newton's method

Allen, Jont B.

Wednesday 19th October, 2022

Abstract

Problem statement: In 1680 Newton proposed a algorithm for finding roots of polynomials. His method has since evolved, but the core concept remains intact. Here we briefly review this evolution, and consider the question of convergence. First, and most important, does the method always converge? Second, if not, what are the conditions necessary to force it to converge? Namely here we investigate the conditions for convergence. In the following we assume a monic polynomial of degree N.

Methods: Convergence requires the important concept of the *Region of convergence* (RoC) for *Netwon's method*. Each starting point on the complex plan must converge to one of the roots of the polynomial because every analytic point on the complex plan has a region of convergence. We show that on boundaries of the RoC regions, the method becomes hyper-sensitive to the initial guess. A slight change in the first guess can cross the RoC boundary, causing the itteration to change the path to a different root. When this happens the change in the step is unpredictable, possibly even chaotic. It is this condition that is the source of a convergence discontinity possibly leading to chaotic convergence. This behaviour is the topic of this document.

Findings: Under certain conditions, non-linear (NL) limit-cycles appear, resulting in a reduced rate of convergence to a root. Since Netwon's method is inherently a discrete linear recursive method, it is important to determine the source of this non-linearity. We conclude that the NL effect is due to violations of the Nyquist Sampling theorem, known as aliasing, which is due to under-sampling near the poles. The conditions and method for uniform convergence are explored. We introduce a complex step size $\alpha = ae^{\jmath\phi}$. The introduction of an adaptively adjusted *step-size* $\alpha \in \mathbb{C}$ greatly reduces the nonlinear effects of aliasing. This method is known as the *damped Newton's method* (Galántai, 2000, p. 25).

The N *regions of convergence* (RoC) are investigated. From the definition of the RoC, these are naturally existing regions defined for every point $s_o \in \mathbb{C}$. Every starting point in the plane comes from one of the N RoC regions, equal in number to the degree of the polynomial. Besides reducing a, one can modify the angle ϕ, redirecting the trajectory away from any RoC boundary so that it does not cross it. The magnitude of the stepsize a can also be reduced to avoid crossing the boundary. This can be highly effective, but will naturally slow the convergence.

Conclusions: *Colorized plots* (Allen, 2020) of the logarithmic derivative are provided to explain when the solution will result in a limit-cycle. Reducing the step-size seems to result in a more rapid, stable convergence, however this is not proved. We show that when $\alpha = 1$, the solution can cross an RoC boundary. In such cases the target root will change, resulting in a chaotic trajectory. As long as the RoC region remains the same, the iteration will converge. We show that when it approaches a limit-cycle, the convergence rate must decrease (it takes more steps to converge). This is because limit cycles introduce the nonlinear process, causing the length of the path to be greater. In the worst case, the path oscillates and the path integral can become arbitrarily long. If the oscillation causes the RoC region to change, the target root correspondingly must change. In theory the RoC could change on each iteration, resulting in chaos. Thus it is helpful to delineate these RoC boundaries, to avoiding NL limit-cycles. We propose using the *ratio test* to detect the transition to a different RoC.

1 Introduction

Newton's method (NM) for finding roots of a polynomial $P_N(s)$ of degree N is a venerable mathematical algorithm. However this view is controversial. NM is openly stated to be unstable,[1] in the sense that it can fail to converge to a root (Stewart, 2012, p. 347). It has properties related to *dynamic analysis* a mathematical science started by Poincaré.[2] We shall show that this view was premature, and in my view, even wrong. Within 10 years Newton's method was renamed the *Newton-Ralphson method*, first published in 1690. There seems to be no difference, other than the name change.

This question was recently explored in Allen (2020), and no instability or limit-cycles were observed. An explanation is due. Newton's method was modified by applying an adaptive *step-size* α, a widely recognized contemporary technique in the signal processing literature.

It is shown that when the roots are complex, random jumps and limit-cycles can occur when the iteration is restricted to real numbers (Allen, 2020). If the initial guess was declared to be complex, and the step-size α was sufficiently small,

[1] https://en.wikipedia.org/wiki/Newton's_method#Failure_of_the_method_to_converge_to_the_root
[2] https://mathinsight.org/newtons_method_refresher

the itteration would always converged as long as the trajectory stayed within the RoC. Two examples were provided using a random initial guess, one of which is presented in Fig. 1 (LEFT) (Allen, 2020, Fig. 3.2). The details of the step-size used by Allen (2020) were not discussed.

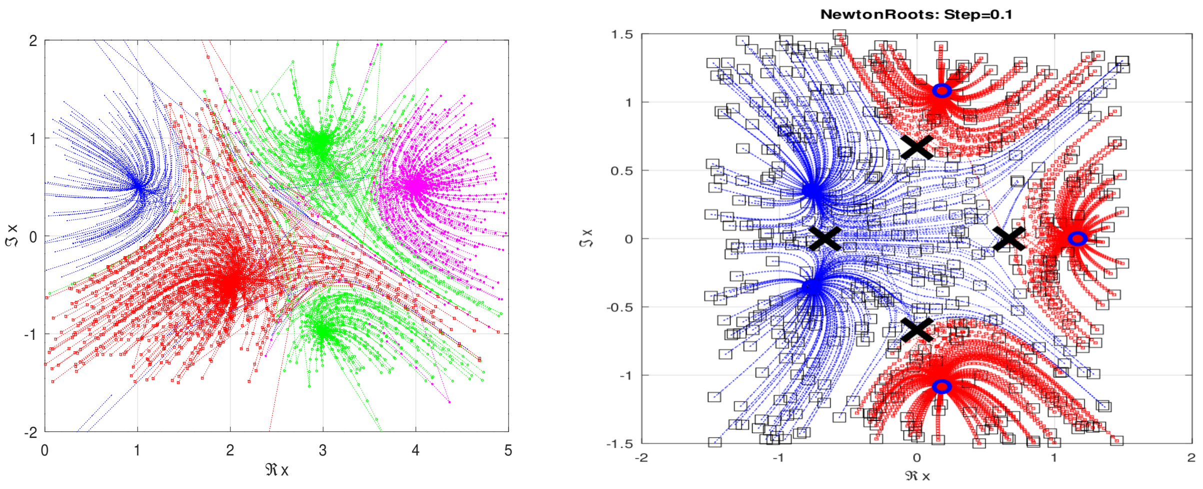

Figure 1: **LEFT:** *This figure is taken from Allen (2020, p. 78). It is a plot of a thousand trajectories for Newton's method, with a random initial guess, taken from the complex plane, between [0,5] along the real axis and ± 1.5 along the imaginary axis, for a polynomial having $N = 5$ complex coefficients. Note the long straight lines that occasionally appear in the figure.* **RIGHT:** *This figure shows the poles and zeros of the polynomial having coefficients $[1000-1-1]$ for 200 itterations of Newton's Method, with random starting points. In this case the roots (o) and poles (x) are superimposed on top of the trajectories of Newton's method. The step size is $\alpha = 0.1$ to reduce the NL effect aliasing. Then the trajectory come close to a pole, the step can jump to a random point in the complex plane.* **./Demo-JPD/ DemoJPD2 or NewtonJPD.m or ZvizPoly.m**

While most of the curves seem to converge to the roots, there are some small percentage of cases where the trajectory take huge jumps to random locations in the complex plane. We shall show that these jumps occur when the trajectory approaches any of the poles of Newton's method, which are at the roots of $P'_N(s) = \frac{d}{ds}P_N(s)$. Near a pole the step can be arbitrarily large, depending on how close the step comes to the pole. This can happen for carefully designed starting values for the itteration. We shall show that these poles (roots of $P'(s)$) are the source of these divergent points and even the limit cycles.

We shall show these jumps in convergence are easily detected using the ratio test between two neighboring steps.

In Fig. 1, the five RoC regions are color coded. Each RoC region is associated with one of the roots. For each initial guess s_o, the itteration will converge to a root. However this depends on the path not approaching a pole. For each pole there are initial conditions that will head for one of the $N-1$ poles. The initial guess is hypersensitive to this initial guess, and the step size, as we shall explicitly show. These RoC regions are compact, well defined, (non-overlapping) complex-valued analytic regions, with each paired with one of the roots of $P'_N(s)$. The poles always lie on the RoC boundaries. These observations are supported by numerical examples. For now, the theory behind this remains open, at least to the author. But the theory of the circle centered on the root, is analytic out to the nearest pole. This concept remains in force for these numerical calculations. The boundaries between each RoC region contains fractial regions which diminish in size as the step size is reduce.

In the literature there are many claims that Newton's method can develop limit cycles. In our initial numerical results (Fig. 1) we did not see any limit cycles. However the simplest example is for monic polynomial

$$P_2(s) = s^2 - s - 1 = (s - s_+)(s - s_-) \leftrightarrow [1, -1, -1],$$

where $s_\pm = (1 \pm \sqrt{5})/2$. NM also requires $P'(s) = 2s - 1$, which has a root at $s_r = \pm 1/2$. For this very simple case, the RoC's are the right and left planes centered on the root of $P'(s_p) = 0$, where $s_p = 1/2$. NM will limit-cycle as long as the initial guess is on the line $s_o = 1/2 + b\jmath$, $b \in \mathbb{R}$, which goes through the root of $P'(s_r) = 0, s_r = 1/2$. here NM for this Carefully Verify! example is

$$s_{n+1} = s_n - \frac{P_2(s_n)}{P'(s_n)} = s_n - \frac{\alpha}{2} \cdot \frac{s_n^2 - s_n - 1}{s_n - 1/2},$$

where $n \in \mathbb{N}$ and $|\alpha < 1|$ is the step size. The initial guess is $s_1 = s_o$.

It is well know that limit cycles are nonlinear. Newton's method on the other hand is a linear recursion equation, with poles and zeros in the complex plane. The the main research question is therefore "How does a linear equation become nonlinear?" proofed to here We show how the these NL limit-cycles may be easily avoided by removing (linearizing) the NL recursion when it is detected.

The suggested procedure will result in a net convergence speed-up, because the NL limit-cycle adds meandering rambling NL steps to the recursion. Thus removing the NL behavior must speedup the iteration. Initially these appear to be conflicting requirments. We hope to convince the reader that there is no conflict, once the behaviour is explained.

Veritasium discussion of chaos from the simple recursion[3] $x_{n+1} = \alpha x_n(1-x_n)$ with $\alpha > 1$.

1.1 Derivation of Newton's method

Consider the polynomial $P_N(s)$, with $s, s_r, c_n \in \mathbb{C}$ and $N \in \mathbb{N}$:

$$P_N(s) = c_N(s-s_r)^N + c_{N-1}(s-s_r)^{N-1} + \cdots + c_1(s-s_r) + c_0, \tag{1.1}$$

where Taylor's formula is used to determine the coefficients

$$c_k = \frac{1}{k!}\frac{d^k}{ds^k}P_N(s)\bigg|_{s=s_r}. \tag{1.2}$$

It is both good practice and convenient to take $c_N = 1$, by dividing the polynomial by c_N, forming the *monic form* of the polynomial. This transformation has no impact on the roots of since $\frac{1}{c_N}P_N(s_r) = 0$. In the following we will assume that $P_n(s)$ is in monic form ($c_N = 1$).

To derive Newton's method we note that every root has a neighborhood called the *Region of convergence* (RoC).

We assume an initial guess s_n for $n = 0$, within the RoC of one of the roots s_r where $s_n \in \mathbb{C}$ within the RoC, namely such that $r(s) \equiv |s_n - s_r| < 1$. It follows that powers of $r^n(s) < r(s)$ for $n \geq 2$, since $|s - s_r| < 1$.

Seting s to this initial guess s_n gives

$$P_N(s_{n+1}) \approx c_1(s_{n+1} - s_n) + c_o. \tag{1.3}$$

This is possible because the higher order terms $k > 1$ are smaller than the linear term $k = 1$.

We now form a recursion s_{n+1} starting from our initial random guess s_n starting from $n = 0$. Let s_r be renamed s_n and s_{n+1} be our next estimate of s_r. Since $r < 1$, s_{n+1} converges to the root s_r in the RoC. We shall confirm this converges in the next section, with several of examples.

In summary

$$P_N(s_{n+1}) \approx c_1(s_{n+1} - s_n) + c_o.$$

Given that $r(s_n) < 1$, $s_n \to s_r$ as $n \to \infty$.

Next replace the two coeficiengs c_1, c_0 by their definitions Eq. 1.2. As $n \to \infty$, $P_N(s_r) \to 0$. Next we use the definitions of c_1 and c_0.

Since c_0 and c_n are given by Eq. 1.2, and replacing s with s_{n+1} and s_r by s_n gives Newton's Method.

$$P_N(s_{n+1}) \approx (s_{n+1} - s_n)\frac{dP(s)}{ds} + P(s_n).$$

Letting $s_{n+1} \to s_r$ the RHS goes to zero. Thus Newton's method is

$$\cancelto{0}{P_N(s_{n+1})} \approx (s_n - s_r)\frac{dP(s)}{ds} + P(s_n).$$

Thus

$$(s_n - s_{n+1})\frac{dP(s)}{ds} + P(s_n) = 0.$$

Solving for s_{n+1} gives

$$s_{n+1} = s_n - \frac{P(s_n)}{P'(s_n)},$$

where $P'(s_n)$ is $\frac{P(s)}{ds}$.

If our initial guess for the root s_1 is close to a root s_r (i.e., $s_1 - s_r$ is within the RoC),

$$|(s_1 - s_r)^k| \ll |(s_1 - s_r)| \tag{1.4}$$

for $k \geq 2 \in \mathbb{N}$. In this case we may linearize the equation $P_N(s_1)$ and solve for s_1

$$\begin{aligned} P_N(s_1) &\approx (s_1 - s_r)\frac{d}{ds}P_N(s)\bigg|_{s_r} + P_N(s_r) \\ &= (s_1 - s_r)P'_N(s_r) + P_N(s_r), \end{aligned}$$

[3] https://www.youtube.com/watch?v=ovJcsL7vyrk&t=1s

where $P'_N(s_r)$ is shorthand for $dP_N(s_r)/ds$. This *complex analytic linearization step* is the key to Newton's method. It will only be true if the difference equation remains linear, which requires the RoC condition Eq. 1.4 to be true.[4] That this, Newton's method is a linear approximation that depends critically on the RoC condition (Eq. 1.4).

One more key transformation is essential: We next modify this linearized RoC approximation to be a recursion, by replacing $s_1 \to s_{n+1}$ and $s_r \to s_n$. Newton's method follows from a recursion where s_{n+1} is closer to the root s_r than $|s_{n+1}/s_n| < 1$, giving

$$P_N(s_{n+1}) = (s_{n+1} - s_n)P'_N(s_n) + P_N(s_n) \to 0. \tag{1.5}$$

Assuming linearity, $s_n \to s_r$, thus $P_N(s_{n+1}) \to 0$. It is clear that this relation only hold under the linearity assumption, since we started by dropping the higher order terms in the Taylor series expansion.

Solving for s_{n+1} recovers Newton's method as the linear difference equation in s_n

$$s_{n+1} = s_n - \frac{1}{N}\frac{P_N(s_n)}{P'_N(s_n)}. \tag{1.6}$$

It is helpful to denote $L_N(s) \equiv P(s)/P'(s)$ as the *step*, and its magnitude as the *step-size*, in terms of two monic polynomials

$$L_N(s_n) = \frac{s_n^N + c_{N-1}s_n^{N-1} + \cdots + c_0}{s_n^{N-1} + \frac{N-1}{N}c_{N-1}s_n^{N-2} \ldots + \frac{1}{N}c_1}. \tag{1.7}$$

The poles or zeros of the step are unmodified by the use of monics. and it reduces the step-size by $1/N$, which improves convergence.The roots of $P'(s)$ (the poles of $L_N(s)$) are invariant to rescaling P and P' into monics. Everything on the right is known; thus when s_n is within the RoC, s_{n+1} will converge to the root s_r as $n \to \infty$.

Equation 1.6 is the solution to the linear difference equation. It has many variants (Galántai, 2000), the simplest being

$$s_{n+1} = s_n - \frac{\alpha}{N}L_N(s_n), \tag{1.8}$$

by introduced a *step-size gain* $|\alpha| < 1 \in \mathbb{C}$, stabilizing the iteration when s_n is in the neighborhood of a pole. Near any pole, the step-size $|L_N(s_n)|$ will become arbitrary large, introducing non-linearity into the iteration. We shall explore one way to detect that the trajectory is moving into one of these unstable regions. We shall show that reducing the step size is not a panacea, and while it helps, it can not solve the core problem of nonlinear limit cycles.

The use of Linear Pre
can find the poles fro
distance given an ana
function. Better: her
know the order to pre

Rational complex analytic functions, such as $L_N(s)$, consisting of poles and zeros, which when expanded as partial fractions, have RoCs that are circles in the complex plane, centered on the poles, with an RoC out to the nearest pole. The RoC's of Fig. 1 are obviously quite different. As long as the RoC condition (Eq. 1.4 holds, Newton's method will converge, as s_{n+1} will be closer to the root than s_n.

However the RoC condition will fail if the value of s_{n+1} crosses an RoC boundary, stepping into a different RoC region. These RoC regions are labeled by different colors in Fig. 1, each of which is centered on one of the five complex roots.

To solve the difference equation we can form the partial fraction expansion (residue expansion) about the $N-1$ roots s_p of $P'(s_p) = 0$, which are the poles of $L_N(s)$, and solve for the eigen-vectors, which form the basis for the linear solution of Eq. 1.6.

To show this let

$$ds \equiv (s_{n+1} - s_n) = -\lim_{\alpha \to 0} \alpha \left(\frac{1}{N}\frac{P_N(s_n)}{P'_N(s_n)}, \right) \tag{1.9}$$

thus

$$\frac{d}{ds}L_N(s) = \sum_{n=1}^{N-1} \frac{R_n}{s - s_p^n} \leftrightarrow \sum_{n=1}^{N-1} R_n e^{-s_p^n t}. \tag{1.10}$$

Here s_p^n is the nth root of $P'(s)$ and $\leftrightarrow$ indicates the inverse Laplace transform.

The inverse Laplace transform of this residue expansion is the solution to this linear difference equation. An alternative is to seek the inverse z transform, which gives the exact solution to the linear difference equation. In both cases the eigen-values are the $N-1$ roots of $P'(s_p) = 0$.

To complete the proof of the failure to converge, one may take the inverse Z transform of Eq. 1.10. Under certain conditions, as $\alpha \to 1$ the imaginary poles can become real, at which point they are no longer conjugate pairs. When this happens, the pole pair split and one pole goes to 0 and the other to ∞. This is the condition which results in an unstable limit cycle (Allen and Sondhi, 1979, Sec. V, p. 126).

[4]Given some basic algebra, Eq. 1.4 is equivalent to $|\frac{s_n}{s_r}| < 1$, which follows from the triangle inequality for Hilbert vectors $s_r - s_n$ and $s_r - s_{n+1}$ within the RoC (Allen, 2020, p. 130). This needs to be verified, not simply stated. Specifically, is $||s/s_n|| < ||s_n/s_{n-1}||$? For example, is $s_{n+1} - 2s_n + s_{n-1}$ "small"? Cite *Mathematical physics*, Balakrishnan (2020), page 508, Eq. 22.32, on the Ratio test for the RoC of any analytic function. This is exactly what I'm trying to say. But it needs to be tested numerically for verification.

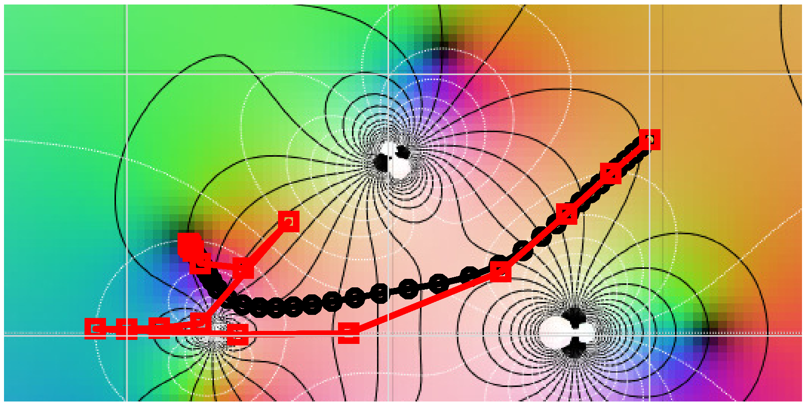

Figure 2: *A* colorized plot *(Allen, 2020, p. 168) of $L_N(s)$ for $P_5(s) = s^5 - s - 1$. This is a blow-up of Fig. 1 (RIGHT) showing the region in the upper half plane, showing the 3 poles and three zeros, in a colorized plot format, as discussed by Allen (2020). The magnitude of $L_N(s)$ is represented by the brightness and the phase $\angle(L_N(s))$ is represent by the color (hue). Thus the dark regions are zeros (roots of $P_n(s)$) and the white regions are poles of $L_N(s)$ (roots of $P'(s)$). Two trajectories of Newton's method are shown, as the black circles and red squares. The starting value for both cases is $s_o = 1 + 0.75\jmath$. The black circles are for to $\alpha = 0.1$ while the red squares are for $\alpha = 0.5$. It is clear that the limit cycle depends on the step-size α, and becomes nonlinear when $\alpha = 1$, leading to a NL limit cycle. With the smaller step-size $\alpha = 0.1$, the recursion remains complex analytic, thus is linear, and converges to root #3 $(-0.765 + 0.3525\jmath)$. The vertical white lines are at -1, 0 ,1 and the horizontal lines are at 0, 1. The polynomial coefficients are $P_5(s) = [1, 0, 0, 0, -1, -1]$ with roots $[1.167, 0.181 \pm 1.084\jmath]$, $-0.765 \mp 0.3525\jmath$. The roots of $P'_v(s) = [5, 0, 0, 0, -1]$ are ± 0.669, and $\pm 0.669\jmath$.* /NewtonsMethod/Demo-Jared/ZvizDemo.m

1.2 Numerical examples of Newton's method

Examples of failures of Newtons Method: Example 1 from the citation[5] looks at the case of $f(x) = x^3 - x + 1$ as an example where Newton's method appears to fail. In this example has two imaginary roots $0.662 \pm 0.5633\jmath$, and a real root -1.325.

The initial guess was assumed to be 1. Since the initial guess has no imaginary part, the recursion proceeds using real arithmetic. As a result the itteration cannot converge to the one of the two complex roots, close to the starting point of $x_0 = 1$. Due to the restriction that the computation must be real, it is forced to the real line where it limit cycles between 1.155 and 0.694, for a 13 step nonlinear limit cycle, Step 13 it breaks free of the limit cycle because it enters the RoC of the real root. On the 14 sample the itteration jumps to -0.74249, within the RoC of the real root, at which point the sequence converges within a few step to the real root at $x_r = -1.3247$.

The 13 limit cycle iterations were wasted computations.

If I start the itteration with an imaginary component at $1 + je - 6$, the itteration converged to the imaginary root at $0.122\jmath$ in 13 steps.

If $x = \jmath$, the solution converges in 3 steps to the upper complex root $x_3(3) = 0.639379 + 0.509792\jmath$

Example 5 from[6] thus $s_r = \ln 2 \approx 61/88 \approx 0.69318$. looks at $f(x) = e^x - 1$, where $x \in \mathbb{R}$, The exact solution is $x_o = ln(2) = 0.69315$.

Applying Newton's method gives

$$x_{n+1} = x_n - \frac{e_n^x - 2}{e_n^x} = s_n - (1 - 2e^{-s_n}).$$

Since e^x is entire, there are no converence issues. Starting from x=1, after three steps it converges to within 4 decimal places.

If x is taken to be complex, the imaginary part quickly decays to zero and depending on the starting condition, approaches one of the infinite number of solutions, within four steps.

For example, the region corresponding to the root near $(2.2-0.3\jmath)$ has an interesting long narrow "RoC stream," converging to the lower-right quadrant, last seen at $(4.5 - 1\jmath)$. There is a second narrow neighboring related parallel (green) stream just north of the red stream, last seen at $(4.5 - 0.9\jmath)$. While this may seem obvious, even expected, I am not aware of any discussion of this obviously important distortion of the RoC's, bound to Newton's method. Presently there is no way to predict the remapping of the RoC regions, due to the normalization by $P'(s)$. However methods for doing so seem within reach using straight-forward modern analysis methods of the RoC boundaries.

Determining the RoCs for Eq. 1.7 is difficult, since the function $L_N(s_n)$ has poles, confounding the locations of the RoC regions. Based on Fig. 1, they are complicated. If s_n approaches one of these the poles, the update can become arbitrarily

[5] https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/newton/complete.html Note: The example given in the above review is obtained using Newton's method for $P_2(x) = x^2 + 2$, but they obtain the wrong answer for the root $x_r = \sqrt{2}$ because they used real arithmetic. It uses the program *Demo-MathInsight/NewtonExp1.m*

[6] https://www.quantamagazine.org/how-mathematicians-make-sense-of-chaos-20220302/

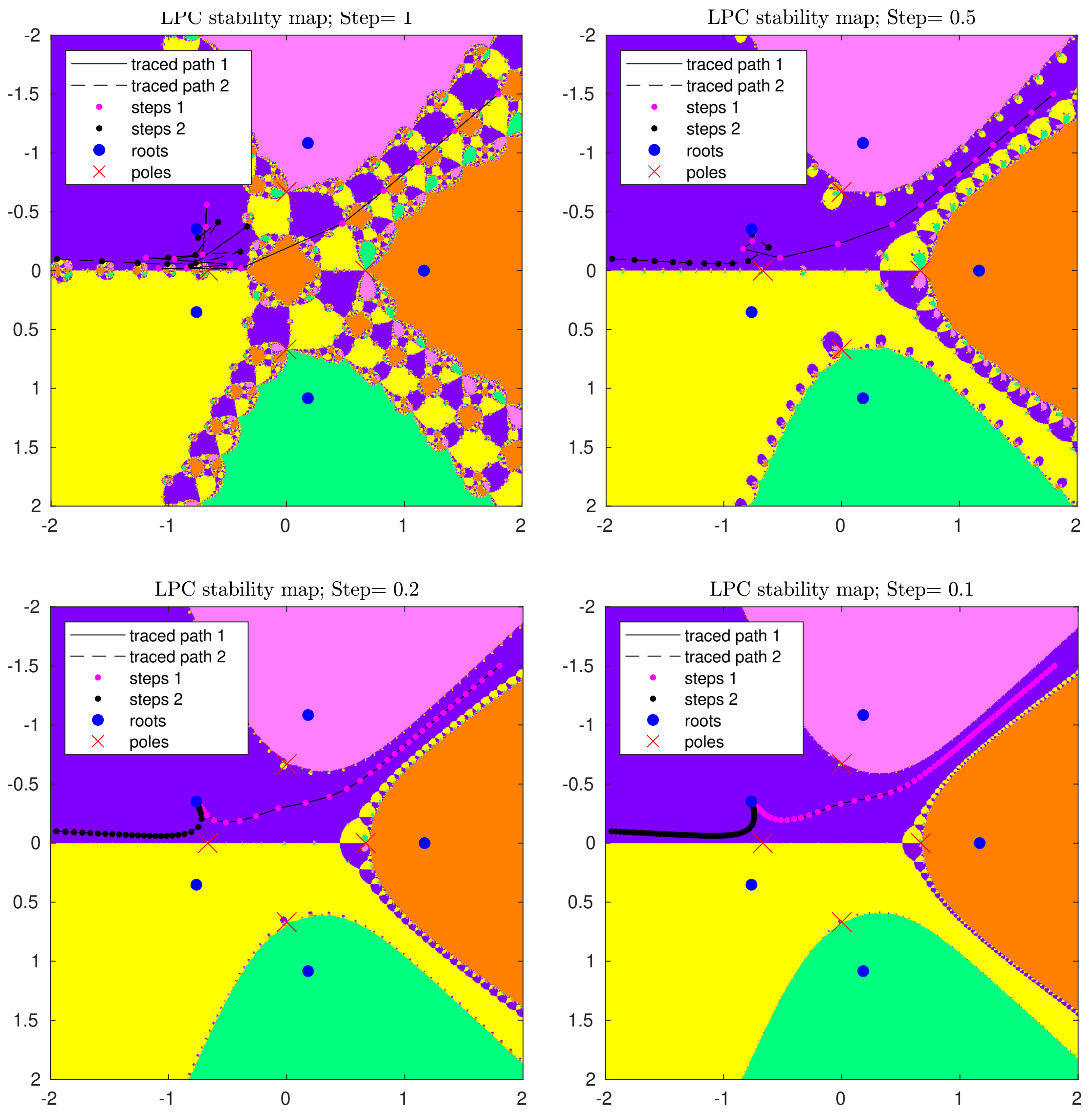

Figure 3: *Four colorized plots showing regions of convergence (RoC) as a function of the step size, as stated in the title of each figure. The fractial regions reside on the RoC boundaries. The size of this fractial boundry dependents on the step size with a step size of 1 resulting in a very large fractal region. Reducing the step size to 1/2 dramatically reduces the fractial regions, by much more that 2. Reducing the step size to 1/10 causes them to almost disappear, except for the small region at* $0.5 + 0\jmath$*. In the RoC, two trajectories are shown. For the Stepsize of 1, the limit cycle can be seen for both the red and black trajectories. For step sizes less that 1/2, the limit cycles are gone. As the trajectories approach the pole, labled as the red X, the trajectory heads for the blue root at* $-0.76 + .352\jmath$*. Limit cycles are wasted steps, easily fixed by reducing the step size. Given a smaller step size, the fractial regions shrink, but never disappear. A simple solution for avoiding the limit cycles is to detect when the boundary has crossed, then backing up to the previous itterate, and reducing the stepsize, and then continue the itteration from that previous point. Detecting the onset of limit cycles near a pole is easy, because the path reverses (oscillates).*

large, depending on how close s_n is to the pole. If the step-size $L_N(s_n)$ is within the RoC, this does not seem occur. When the value of s_{n+1} falls outside the RoC there can be an arbitrary increase in step-size. Normally this will not happen, since when s_n approaches a pole, since s_{n+1} is naturally "pushed" away from the pole, may be seen in Fig. 2. Here step is

$$L_N(s) = -\frac{1}{5}\frac{s^5 - s - 1}{s^4 - 1/5}. \tag{1.11}$$

Limit-cycles can occur if the initial (starting) condition s_0 is close to an RoC boundary, or even worse, on top of (i.e., *very* close to) a root of $P'(s)$ (a pole s_p of $L_N(s)$). The natural convergence factor $1/N$ in Eq. 1.8 can improve the convergence, but it will not remove the problem if the trajectory comes close to a pole. This is the case of these "rare" (seeming-random) circumstance, which depend on s_o and α.

While both trajectories start at s_o, the influenced by the two poles at $\pm 0.669\jmath$, is much greater for the larger step size (red squares). The smaller step-size (black-circles) result in a linearized trajectory, which drifts smoothly away from both poles, eventually converging on the zero of L_N at $-0.766 + 0.352\jmath$. The large step-size (red square) trajectory jumps erratically over the pole at $-0.766 + 0.35\jmath$, resulting in a nonlinear limit-cycle oscillation. When $\alpha = N$ (not shown) there is an even larger limit-cycle. The coefficients of $P_5(s)$ are $c_n = [1, 0, 0, 0, -1, -1]$ having its five roots at $\{1.1673, 0.18123 \pm 1.08395\jmath, -0.76588 \pm 0.35247\jmath\}$. The roots four for $P'(s)$ are $\sqrt[4]{0.2}(\pm 1, \ \pm \jmath)$ where $0.66874^4 = 0.2$. The Octave code is available from the author.

I think this now fixed: trajectory labeling. S thing is wrong here. the middle (or last) r

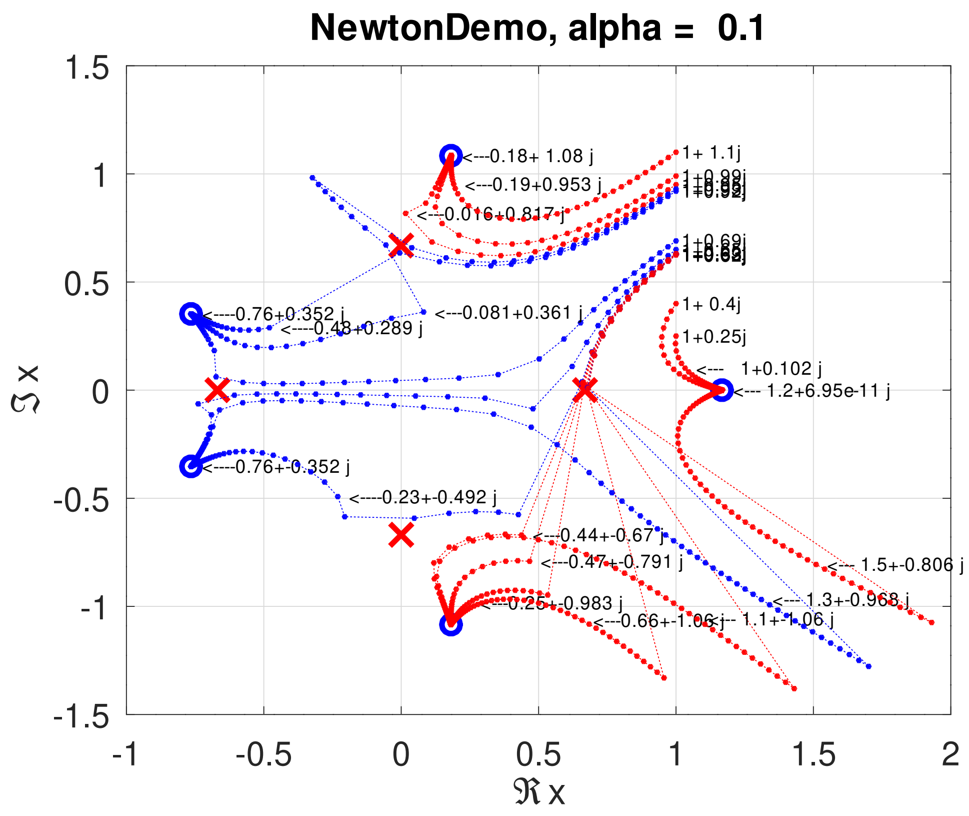

Figure 4: *This numerical experiment for polynomial coefficients* $[1, 0, 0, 0, -1, -1]$ *(the same as shown on the right panel of Fig. 1) reveals the inner workings of Newton's method. Seventeen different starting values* (s_o) *have been carefully chosen, to find the RoCs associated with each* s_o*. All the starting values are of the form* $s_o = 1 + \jmath\beta$*, where each* β *and the converged root are indexed in Table 1. The root index is indexed goes from 1 to 5, counting counter clockwise from the root #1 at 1.2. The scattering angle is determined by the residue of the scattering pole. Each curve is labeled twice, once at the starting point and once at another point somewhere along the trajectory. The most interesting case is for starting points between* $1 + 0.62\jmath$ *and* $1 + 0.5999\jmath$ *which converge to dramatically different RoCs due to squarly hitting the positive real pole* $0 + .6\jmath$ *having initial values* $1 + \jmath\, 0.6 \pm 0.001$*.* /NewtonsMethod/Demo-Jared/Demo-Jarde.m

1.2.1 Source of the limit-cycles

Based on the results of Fig. 2, it is clear that the NL limit-cycles are dependent on the step size, and are significantly reduced for a smaller step-size ($\alpha = 0.1$). However by taking a different value of s_o, the limit-cycle problem will return, once the trajectory is comes closer to the pole, which is controlled by s_o. This effect is shown in considerable detail in Fig. 4. The value to s_o will need to be finely tuned, and the trajectory will intercept the pole, resulting in a new NL limit-cycle.

The Gauss-Lucas theorem[7] (Allen, 2020) come into play at this point. This theorem says that the convex hull of the roots of a polynomial bound the roots of its derivative,. This important theorem, seems highly relevant to the convergence of Newton's method (Galántai, 2000). This publication has 75 highly relevant citations, many on the problem address here, but apparently none solving it.. I will argue that the key to avoiding the troublesome limit-cycles, is to detect them. This seems easily implemented.

The following quote may be found on (Galántai, 2000, p. 39):

> The possibility that a small change in s_o can cause a drastic change in convergence indicates the nasty nature of the convergence problem. The set of divergence points of the Newton method is best described for real polynomials.

As demonstrated in Fig. 4, it is clear that we agree on Galántai's first point. The second remains open.

One difference in opinion is that it should be trivial to detect these arbitrary large non-analytic jumps $s_{n+1} - s_n$, thus once detected, a strategy can be developed to avoid the problem, say by slightly moving s_o to change the trajectory, or by making the step-size α smaller, say by a factor of 2. As long as the step size is small enough, the trajectory will converge because $L_N(s)$ is everywhere complex analytic except at the poles.

One method that seems useful is to use Newton's method to find the poles of $P'_N(s)$. In this case the poles will be zeros, since we are looking for the roots of $P'_N(s_r) = 0$. In this case the poles will be the roots of $P''_N(s_r) = 0$. As higher derivatives are taken, the largest root or roots, can drop out, since the roots of the derivative are also inside the convex hull of the polynomial we are studying. Each time we take a derivative, the convex hull will shrink. It may be interesting to look at some numerical examples graphically to see the family of nested convex hulls. This could be especially important when the roots lie on a line, such as the famous Wilkinson example having real roots from 1 to 20, or even more interesting, the roots of the Riemann zeta function, where the roots seem to lie on a line.This case is more complicated, because there are an infinite number of zeros. However the roots of $\zeta'(s)$ must lie between the roots of $\zeta(s)$. Likewise, the roots of $\zeta''(s)$ would lie between the roots of $\zeta'(s)$. If this logic proves to be correct, it provides some important topological structure for these roots.

Examples of such iterations, for hundreds of random initial conditions, are shown in Fig. 1. This figure clearly delineate the five RoC regions (not discussed in Allen (2020)). Also not discussed are cases where the initial conditions lead to an instability in the iteration.[8]

For example in Fig. 1, the red "stream" corresponding to the root near $(2.2 - 0.3\jmath)$ has an interesting long narrow "RoC-stream," converging from the lower-right quadrant, first seen at $(4.5 - 1\jmath)$. There is a second neighboring parallel (green) RoC-stream just north of the red stream, last seen at $(4.5 - 0.9\jmath)$. While this may seem obvious, even expected, I am not aware of any discussion of this obviously important distortion of the RoC regions, bound to Newton's method. Presently there is no way to predict the conformal remapping of the RoC regions due to the normalization by $P'(s)$. Methods for doing so seem within reach using straight-forward modern analysis techniques of the RoC boundaries, as will be explored in Sec. 2.

1.3 Addition of a step-size

If we modify Eq. 1.6 by introducing the step-size $\alpha < 1$, we obtain Eq. 1.7.

$$s_{n+1} = s_n - \frac{\alpha}{N} L_N(s_n). \tag{1.12}$$

The poles and zeros of the update $L_N(s)$ remain the same. The effect of the reduced step-size is to force the trajectory to be more sensitive to the influence of the poles, rather than stepping over them. The smaller step-size stabilizes (i.e., eliminates) the nonlinear limit-cycles.

When the initial value for the iteration s_o is close to the cross-over of two RoCs, $s_n \to s_{n+1}$ can cross over an RoC boundary, radically changing the limit point (root it converges to). The likelihood of this depends critically on α. A limit cycle can happen when s_n comes close to one of the poles of $L_N(s_n)$. At a pole, the value of L_N can become arbitrary large, causing the unmodified ($\alpha = 1$) update $L_N = s_{n+1} - s_n$ to fail to satisfy the required RoC convergence condition (Eq. 1.4).

This is effect is best show with an example, using the same trajectory, but with a second smaller step-size. The convergence factor α is not necessary when the initial estimate of the pole does not lead to a limit-cycle. It is easy to detect limit-cycles because they are nonlinear (and for some reason, tend to be real).

One popular strategy for detecting the pole is to look at the magnitude of the step. If $\hat{s}_{n+1}$ becomes large, the assumption that the RoC condition fails and the step is reverted back to s_n and the step-size is reduced by $\alpha < 1$, to reduce the probability of a limit-cycle. This then repeated until the RoC condition ($|s_n| > |\hat{s}_{n+1}|$), avoiding the limit cycle.

Based on our existing numerical results, the addition of the convergence factor α seems unnecessary when the the initial value is well within the RoC, as required by Eq. 1.4. The main question is when (and why) the limit-cycles are created

[7] `https://en.wikipedia.org/wiki/Gauss-Lucas_theorem`
[8] http://jontalle.web.engr.illinois.edu/MISC/NewtonRoots.22/

Table 1: *Table showing the imaginary part of $s_0 = 1 + \beta\jmath$, along with the RoC target root index number.*

β	root
0.25, 0.4	#1
0.95, 0.99, 1.1	#2
0.69, 0.92, 0.93	#3
0.65, 0.632	#4
0.63, 0.631	#5

with Newton's method. This question is at least partial explored in the example of Fig. 2. As long as the RoC condition is maintained, each step will progress closer to a root, and in the limit, as $n \to \infty$,

$$\frac{P_N(s_n)}{P'_N(s_n)} \to 0, \tag{1.13}$$

since $s_n \to s_r$ as $n \to \infty$.

1.4 A numerical example

To study the convergence and limit-cycles it is helpful to use numerical methods and look at specific starting points, such that the trajectory approaches a pole (root of $P'_n(s)$).

Note that if s_n is at a root, the numerator term $P_N(s_n) \to 0$, thus and the update goes to zero ($s_{n+1} \to s_n \to s_r$). This property can be useful in detecting the stopping condition. Note that s_n cannot be near a pole (which cannot be near the zeros, as required by the Gauss-Lucas theorem).

In practice, once near a root, it takes only a few steps to converge. In experimental trials (see Fig. 2) fewer than 10 steps can give double-precision floating-point machine accuracy. If any value s_n is close to a root of P'_N, the recursion fails, giving a large random value for L_N, forcing the method to restart at some random point far from the roots. In such cases the solution typically converges to a different root (RoC). It should not be difficult to detect these large non-convergent steps by monitoring $|s_{n+1} - s_n|$, which must always be monotonically decreasing.

Figure 2 shows two paths with the same initial condition but different step-sizes. The utility of the reduced step-size is clear from Fig. 2.

Starting at $s_o = 0.7 + 1.5\jmath$, the path develops into a NL limit-cycle near the second pole at $-0.766 + 0.352\jmath$. It is a combination of the large steps and the proximity to the negative real pole that results in the nonlinear limit-cycle. On the 10 step it comes out of the limit cycle and after a 10 more steps, it has converged to the root at $-0.766 + 0.352\jmath$. When the step-size α is reduced from 0.5 to 0.1, as the path approaches the positive pole, it moves away, avoid the limit-cycle. With steps sizes of 0.2, 0.3 it becomes captured by the pole. It still converges, but much more slowly as the NL becomes greater. With the step-size of 0.9 (not shown), the trajectory is similar to that of 0.5, but after 5 steps it is in well within the RoC of the zero at $-.8 - .3\jmath$. After 20 steps, the error is less than 1%. The convergence time may be a crude quantitative measure of the NL. The spectral properties or smoothness of the trajectory may be more appropriate. The red-squares is an example of a smooth analytic trajectory while the red-squares is chaotic. This behavior would show up in a time-series analysis.

If one assumes that the initial guess is real ($s_o \in \mathbb{R}$) Octave/Matlab evaluates the polynomial using real arithmetic, forcing the estimate $s_{n+1} \in \mathbb{R}$ (Allen, 2020). Thus the iteration cannot converge when $s_r \in \mathbb{C}$.

Roots $s_r \in \mathbb{C}$ may be found by a recursion that defines a sequence $s_n \to s_r \in \mathbb{C}$, $n \in \mathbb{N}$, such that $P_N(s_n) \to 0$ as $n \to \infty$. As shown in Fig. 2, solving for s_{n+1} using Eq. 1.6 always gives one of the roots, due to the analytic behavior of the complex *logarithmic derivative* $P'_N/P_N = d\log(P_n(s)$.

With every step, s_{n+1} is closer to the root, finally converging to the root in the limit. As it comes closer, the linearity assumption becomes more accurate, resulting in a better approximation and thus a faster convergence.

Equation 1.6 depends on the log-derivative $d\log P(x)/dx = P'(x)/P(x)$. It follows that even for cases where fractional derivatives of roots are involved (Allen, 2020, p. 197, #5); Newton's method should converge, since the log-derivative linearizes the equation.[9]

2 A proof of convergence

We define a monic polynomial $P_N(s) = s^N + \sum_{n=N-1}^{0} c_n s^n$ (i.e, $c_N = 1$) having roots $r_k \in \mathbb{C}$, such that $P_N(r_k) = 0$ where $s = \sigma + \jmath\omega$ is called the Laplace frequency. We also define the *step-size* $\alpha = |\alpha| e^{\jmath\angle\alpha}$ (see Eq. 1.6) where $a \leq 1$, and

[9]This seems like a way to understand fractional, or even irrational, roots.

$\phi = \angle\alpha$ and $0 \le \phi \le 2\pi$.

Here we present the hypothesis we wish to prove true, and outline a proof, which can be verified numerically, but only within the limits of the computers numerical accuracy, Thus we have only proved the hypothesis for rational roots $r_k \in \mathbb{Q}$.

However we believe it to be true within these numerical limits (rational roots) ($r_k \in \mathbb{F} \equiv n/m$), where $n, m \in \mathbb{N}$, $P_n(s) = \Pi_{k=1}^{N}(s - r_k)$ and $P_N(r_k) = 0$. We seek a generalization for irrational roots $r_k \in \mathbb{I}$.

Theorem 1
Given a monic polynomial $P_N(s)$ we apply Newton's method (Eq. 1.6) starting with $n = 0$

$$\begin{aligned} s_{n+1} &= s_n - \frac{\alpha}{N} L_N(s_n) \\ n &= n+1, \end{aligned} \tag{2.1}$$

and an arbitrary starting point $s_o \in \mathbb{C}$, including the point at $s = \infty$. We shall prove that every s_0 lines within a unique *Region of convergence* (RoC). There are N of these regions, one for each root r_k.

After the first step we have $s_1 = s_0 - \frac{1}{N} L_N(s_0)$. If $|s_1/s_0| < 1$ we have satisfy the RoC assumption built into Newton's method, thus we have moved closer to the root r_k associated with s_o. Thus s_1 is in the same RoC as s_o. In this case we must repeat this calculation for $n > 1$, and at each step verify that the RoC condition ($s_{n+1}/s_n < 1$) holds.

If the RoC condition fails for n, we must repeating Eq.2.1, restarting from s_n, the last point inside the RoC. We have two options at this point:

1. reduce the step size a, and repeat Eq. 2.1
2. change $\phi = \angle\alpha$ leaving $a = 1$.

A third more aggressive option is to vary both a and ϕ such that $|\frac{\alpha}{N} L_n(s_n)| \approx 1$. The locus of such points identifies the RoC boundary between that containing s_0 and a neighboring RoC region. As the angle $phi = \angle\alpha$ is varied, the RoC condition with vary from greater than 1 at $\phi = 0$, because in this direction the RoC condition failed, to less than 1 when $phi > 0$. This appears to be a variant on the conjugate-gradient method for Newton's Method. It seems likely, but presently untested, that the best approach is to vary ϕ to minimize s_{n+1}/s_n, and then step in that direction. Such an approach should give a faster convergence because the RoC condition will be smaller. This approach eliminates the need to make $a < 1$, thus further increasing the convergence rate.

Our claim is, that by changing the angle of the step size, we will find a range of angles that optimize the RoC condition (find the smallest ratio with respect to ϕ).

Moving in the direction that $s_{n+1}/s_n \approx 1$ delineates the RoC boundary. The prediction is, that by moving in the direction where RoC is 1, will trace out the RoC boundary region. Crossing the boundary and starting with s_{n+1}, Eq. 2.1 should converge to a different root.

Numerically this is easily demonstrated, providing an empirical proof of our hypothesis (theorem).

The RoC regions may be seen in Fig. 1 as the different colored regions. The long dashed lines indicate when the trajectory jumped from on RoC to another due to coming under the influence of a pole of $L_N(s)$ (zero of $P'(s)$).

3 Summary

We don't understand many observations in science (math and physics), which with some analysis, can eventually be explained. It is the *reductionist* method in science that explains the success of the scientific method. This might be viewed as a form of evolution: success begets more success, while failure eventually dies off, perhaps slowly.

The process of systematically exploring these seemingly tiny discrepancy, almost always leads to new knowledge. Seeking out these idiosyncratic inconsistencies and trying to explain them is at the heart of the scientific method. When a problem is longstanding and considered fundamental, its resolution can even lead to a paradigm shift. Not surprisingly such deep insights are rarely welcomed by the scientific community, rather they are viewed with great skepticism. This can be good when if doesn't go on for 50 years.

The problem of finding roots using Newton's method is an excellent example. It is a case that can be explained only after careful thought and iterative analysis. I feel we are either close to that understanding, or it has been explained clearly enough that the debate can be stopped, and final conclusions may be reached. However, realize that there is no "final."

Limit cycles do exist in Newton's method, but in my view, they are due to under-sampling the complex plane. This is an example of aliasing, in the Nyquist sense. An under-sampled process becomes nonlinear when the "high frequencies" alias into the "base-band" frequencies. This nonlinear effect is easily removed by increasing the sampling rate above the Nyquist sampling frequency, defined as twice the highest frequency in the signal. While that concept is not clear in the context of Newton's method, it can explain limit-cycles, and slightly (2x-3x) increasing the computation, by decreasing the step-size α, the aliasing may be brought under control, and the problem becomes linear and well behaved. The onset of aliasing is easily detected. This leads to a well know method in signal processing called the *adaptive step-size*, which has been successfully applied in many engineering problems. It is, I believe, well understood and characterized in terms of aliasing (Allen and Sondhi, 1979, Sec. V, p. 126).

4 Acknowledgments

I would like to acknowledge the help of John D'Angelo and Jared Bronski for their dogged perseverance to prove me wrong. Their very negative input was an important psychological impedance to work against. If you side with their views, please talk to me. I'm open to more, constructive push-back. If you wish to better understand any of the concepts describe here, please read Allen (2020, p. 77-80).

5 Some ideas

Treat $L_N(s)$ as a transfer function and expand it in a residue expansion (partial fraction). Then once the roots R are known, find the polynomial coefficients C, and then build the Eigen-matrix (companion matrix) M.

The method is:

1. Use Netwon's method to find the vector of roots R
2. Find the vector of polynomial coefficients $C = poly(R)$
3. Compute the eigen-matrix from C: $M = compan(C)$
4. Finally find the eigenvectors $[E, L] = eig(M)$
5. Verify that $diag(L) == R_{Newton}$ (Roots as determined by Newton method), in case there is numerical error.

The eigenvectors span the space defined by $L_n(s) = 0$. Any solution of the characteristic polynomial (either the denominator or numerator polynomials) spans the space of the roots.

Summary: Given M, find the eigen functions E using `eig(C)`, where `C=poly(R)` are the polynomial coefficients, $R = [r_1, r_2, \cdots]$ is the complex root vector, `[E, L]=poly(C)` and `R=diag(L)`.

References

Allen, J. B. (2020). *An Invitation to Mathematical Physics, and Its History*. Springer.

Allen, J. B. and Sondhi, M. M. (1979). Cochlear macromechanics: Time-domain solutions. *Journal of the Acoustical Society of America*, 66(1):120–132.

Galántai, A. (2000). The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1-2):25–44.

Stewart, J. (2012). *Essential Calculus: Early Transcendentals*. Cengage Learning, Boston, MA. https://books.google.com/books?id=AcQJAAAAQBAJ.