EXETER MATHS SCHOOL
FREEDOM FOR THINKERS

Curious Properties of Newton's Method

Tomass Penny, Nirius McDade, Ed Jillians, Oliver Grey, Joshua Scates

Exeter Mathematics School

Introduction

In this project, we investigated the curious properties of Newton's Method. It is a powerful algorithm that was first formulated by Isaac Newton and Joseph Raphson as a way to solve polynomials of degree 5 or greater. The method makes use of iteration to generate answers that eventually converge to polynomial roots. This makes it extremely useful for solving equations that cannot be solved analytically or by using traditional methods. Depending on the nature of the polynomials used and the choice of starting value, Newton's Method can form complex patterns and exhibit strange properties.

Newton's Method

The general formula for Newton's Method (also known as the Newton-Raphson Method) is:

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

This algorithm, when graphed, uses tangents to a point on polynomial to calculate the next point to be used as a guess. This requires the use of differentiation to find the equation of the tangent. To begin the process, an initial guess for the location of a root is made.

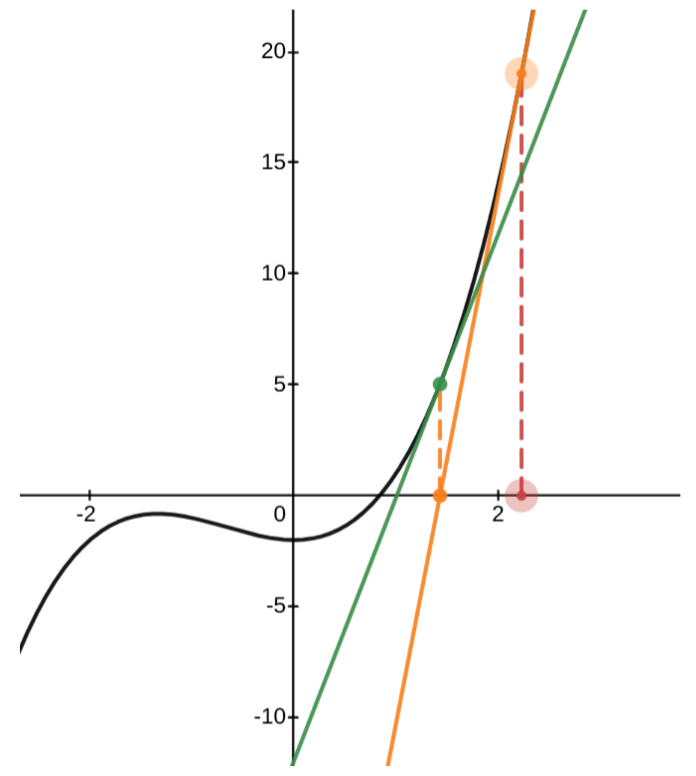


Figure 1. Example of Newton's Method being used on equation $x^3 + 2x - 2$. Solid lines represent the use of tangents to the output whereas the dotted lines represent the first intersection with the line drawn vertically from the x-axis.

The point where the tangent intersects the x-axis becomes the basis of the next iteration. After each iteration, the result becomes closer to the location of one of the roots.

Basins of Attraction

As with most iterative methods, Newton's Method converges to the roots of an equation depending on the first input of the formula. For each root, the set of starting guesses that converge to it are called the "basin of attraction".

Finding the basins of attraction is not as simple as using a formula. It is difficult to predict which root any particular input will converge to without applying the method each time. Hence, using a computer program to test a large range of points is the best way to find the basins of attraction. When creating graphical representations, a colour is assigned to each root and the initial guess is coloured according to which root it converges to. Points that are not in any basin of attraction are coloured black. Interestingly, points extremely close to each other can often be in separate basins of attraction, therefore making Newton's Method an example of sensitive dependence on initial conditions.

However, Newton's Method does not always give useful results. For example when:

- The initial guess is at a critical point. The tangent created using the guess will be parallel to the x-axis and therefore fail to converge to a root.
- The initial point coincides with a cycle. The iteration can get stuck in a loop when 2 tangents loop back onto one another, never converging.
- The function has no real roots. Newton's Method will chaotically move around the x-axis if there are no solutions.
- The function is non differentiable at the point. Newton's Method will collapse.

Complex Numbers and Functions

Complex numbers are numbers that cannot be defined purely in real terms. They contain both imaginary components (denoted by the element i) and real components. Besides that, they can be used and referred to in essentially the same way as standard real numbers. in a similar fashion to real numbers, they can be graphed, although on a different type of graph called an 'Argand diagram', representing the real part of the number on the x-axis and the imaginary part on the y-axis.

In addition, complex numbers can be utilised in functions the same way real numbers can. These are called complex functions (denoted as $f(z)$).

When plotting complex functions, they can be thought of as their components:

$$u + iv = f(x + iy) \qquad (2)$$

Plotting these can be more complicated however, due to the need to represent 4 dimensions (x, y, u, v) It is easier to represent these in 3D graphs, although representing them in 2D is possible with the use of colour mapping. We can encode the magnitude and phase of $f(x+iy)$ onto a 2D colour map using hue and intensity to represent the complex value of the function [3] as shown below

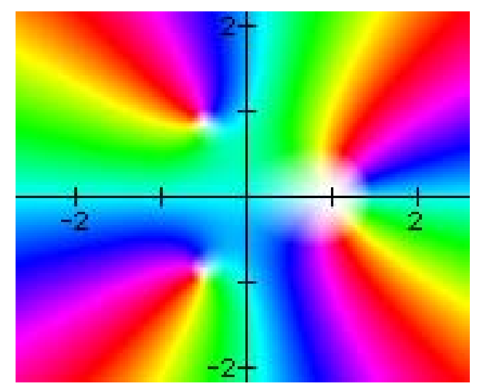


Figure 2. Colour mapping of the complex function $(z^3 - 1)(z - 1)$. When the mapping is black, f(z) is 0 and where it is white, f(z) is stretching off into infinity whilst the hue represents the argument of the function.

Roots of Unity

For a complex number z, the equation $y = z^n$ has n roots of unity. Roots of unity are complex numbers z that satisfy $z^n = 1$. Using polar form, a root of unity can be written as $z = cos\theta + isin\theta$. So the equation becomes $(cos\theta + isin\theta)^n = 1$. Then by De Moivre's Theorem, $(r(\cos\theta + i\sin\theta))^n = r^n(\cos(n\theta) + i\sin(n\theta))$, roots of unity can be found by $cos(n\theta) + isin(n\theta) = 1$, since r must equal 1. Then by comparing the imaginary and real components when $z = 1$, we can find that $n\theta$ must be integer multiples of 2π because the imaginary component must be 0. $n\theta = 2\pi k \Rightarrow \theta = \frac{2\pi k}{n}$. So $z = cos\frac{2\pi k}{n} + isin\frac{2\pi k}{n}$ for $k = 0, 1, 2..., n-1$. You can also use the Euler form,$z = e^{2ki/n}$ to find the roots of unity.

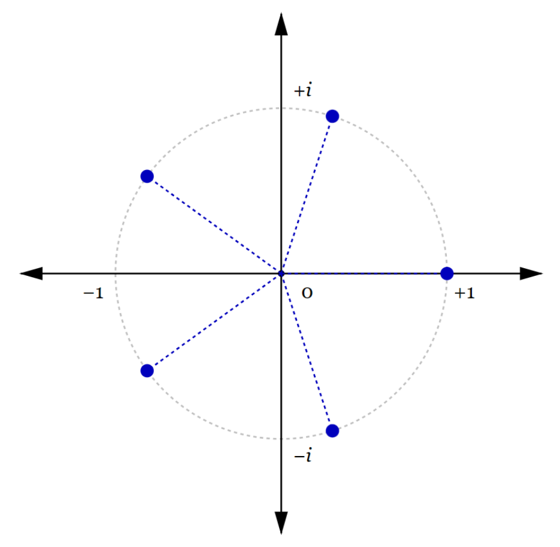


Figure 3. 5th Roots of Unity Represented in the Complex Plane

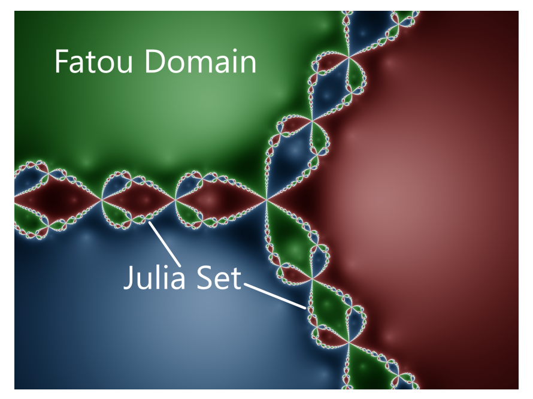
[6]

If we define ω to be the root with the smallest positive argument, we can write the nth roots of unity as $1, \omega, \omega^2, \omega^3, ...\omega^{n-1}$ because they are able to form a geometric progression using De Moivre's Theorem again. Putting these into $S_n = a\frac{\omega^n - 1}{\omega - 1}$, where $\omega^n = 1$, shows that the roots of unity of a function sum to 0. When graphed, roots of unity form vertices of an n sided regular polygon, inscribed onto the circumference of the unit circle. [5]

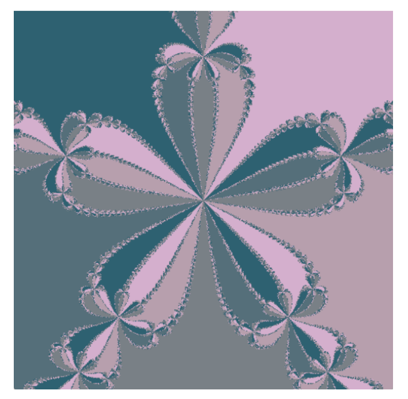
Equations with roots of unity are often chosen when generating fractals because the roots are uniformly distributed about the unit circle and therefore create interesting patterns with a natural symmetry. Furthermore, the ease at which roots of unity can be calculated makes experimentation and implementation more convenient for users.

Fractals

Newtonian Fractals are the Fatou set of the Newton's Method function applied to points in the complex plane. The Fatou set is the union between the Fatou domains of the fractal. A Fatou domain is defined as a set or region of "stability", where the associated points converge in a regular, well behaved manner. Complimenting the Fatou set is the Julia set, which can be thought of as the boundary between the Fatou domains, where the points neither converge nor diverge under iteration. Points on the Julia set behave unpredictably or chaotically, contributing to intricate and dynamic fractal patterns.



(a) Visualisation of Fatou and Julia sets[2]

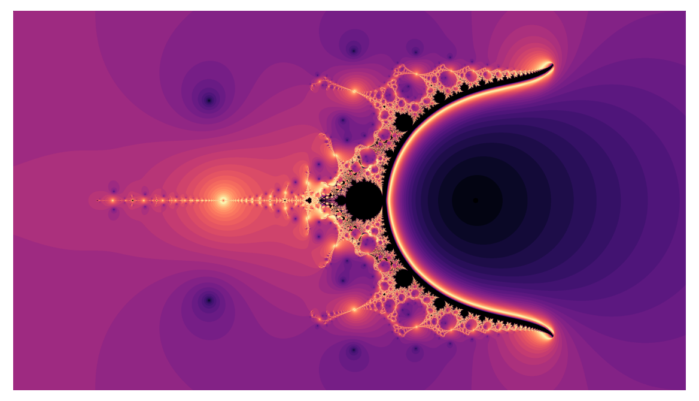


(b) Newtonian Fractal generated on Python

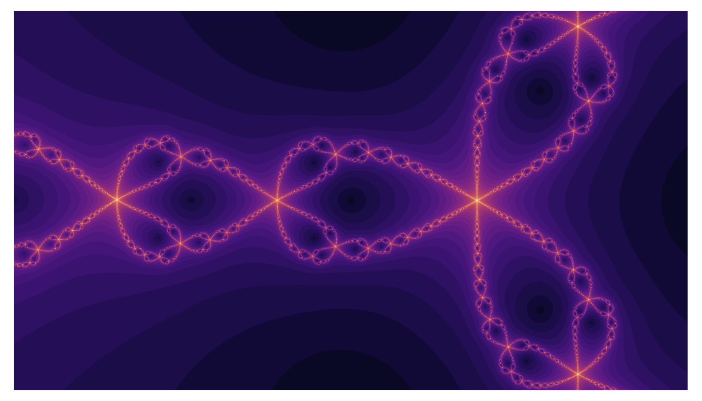
To create the pattern, each of the n roots is assigned a different colour and then, Newton's Method is applied to a high resolution of points in the complex plane. Each pixel is coloured according to the root that it its corresponding point converges to. This creates a self-similar and infinitely repeating fractal where the coloured regions represent the Fatou sets, similar to the basins of attraction. For an extra level of complexity, different points can be shaded depending on the rate of convergence. Lighter shades indicate that points in the region converged quickly to a root, whereas darker shades highlight divergent or slowly converging regions.

Below are some variations of Newton Fractals: [1]

1. **Relaxed Newtonian Fractal** This variation is made by introducing a relaxation factor, R, into the method. This was originally done to increase the rate of convergence. Changing the relaxation factor can stop the fractal from being convergent in places. This creates interesting patterns, some of which aren't mathematically meaningful. $x_{n+1} = x_n - R\frac{f(x_n)}{f'(x_n)} + c$
2. **Nova Fractals** This variation has an added value of c in each iteration. There are both Julia and Mandlebrot versions of the Nova fractal. Mandlebrot versions are created by setting c to the pixel position instead of the initial guess. $x_{n+1} = x_n - R\frac{f(x_n)}{f'(x_n)} + c$



(a) Mandlebrot Nova Fractal [4]



(b) Julia Nova Fractal [4]

Acknowledgements and Bibliography

Professor John Thuburn - Thank you for setting such an interesting project

[1] HDPZ. *Convergent Fractals*. URL: `http://www.hpdz.net/TechInfo/Convergent.htm#Nova`.

[2] Georg-Johann Lay. *Visualising Julia Sets*.

[3] Nucalc. *Complex Functions*. URL: `https://www.nucalc.com/ComplexFunctions.html`.

[4] Lars Rotgers. *Nova Fractals*. URL: `https://rotgers.io/posts/nova-fractals/`.

[5] TLMaths. *A-Level Further Maths B10-02 Complex Numbers: Formalising the nth Roots of Unity*. URL: `https://www.youtube.com/watch?v=xMtN-8oSIJQ&list=PLg2tfDG3Ww4sIZSKWUv6Yol2VMNCDxFQ9&index=2`.

[6] Wikipedia. *Roots of Unity*. URL: `https://en.wikipedia.org/wiki/Root_of_unity`.

Contacts

Tomass Penny: tomasspenny@exe-coll.ac.uk
Nirius McDade: niriusmcdade@exe-coll.ac.uk
Ed Jillians: edjillians@exe-coll.ac.uk
Oliver Grey: olivergrey@exe-coll.ac.uk
Joshua Scates: joshscates@exe-coll.ac.uk