Teaching Mathematics and Statistics in Sciences - IPA HU-SRB/0901/221/088
Interesting Mathematical Problems in Sciences and Everyday Life - 2011

Understanding convergence and stability of the Newton-Raphson method

Zoltán Kovács
Department of Analysis, University of Szeged
H-6720 Szeged, Aradi vértanúk tere 1.
kovzol@matek.hu
http://www.math.u-szeged.hu/~kovzol

Approximated solution of one and multivariable equations is an important part of numerical mathematics. The easiest case of the Newton-Raphson method leads to the $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ formula which is both easy to prove and memorize, and it is also very effective in real life problems. However, choosing of the starting x_0 point is very important, because convergence may no longer stand for even the easiest equations.

Computer aided visualization can give a good picture of "good" and "bad" x_0 points, and we are also able to study the end point of the convergence. The relationship between the cubic polynomial equations and the Newton fractal is very obvious, and the latter is a marvellous case of self similarity in fractal geometry. To show such behavior we use the XaoS software which is able to demonstrate the common basins of convergence with different colors in real-time visualization, including zooming in or out.

The multivariate Newton-Raphson method also raises the above questions. Visual analysis of these problems are done by the Sage computer algebra system. Sage has a large set of modern tools, including groupware and web availability. While Sage is a free software, it is affordable to many people, including the teacher and the student as well.

1. Introduction

Finding roots of univariate equations is a very important task of applied mathematics. It is a basic question how one can solve the $f(x) = 0$ equation by utilizing a computer. To answer this question, many methods exist for the solution of this problem. Assuming that f is differentiable and its derivatives are non-zero, one can define the

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

sequence, where x_0 is a "well chosen" point in the neighborhood of one of the expected roots. This calculation process is often called *Newton–Raphson method* in numerical mathematics.

Mathematics is usually precise enough, but the above definition is not. What is a *neighborhood*? What does *well chosen* mean? Of course, there are many investigations and general answers for that, but to learn how difficult this question is, it is easy to construct convenient examples.

This paper tries to show this difficulty on a real life example: the solution of the cubic equation, that is a common problem in many applications.

2. The cubic polynomial equation

Grammar school methods of solving first and second order polynomial equations are well known. The equation

$$f(x) = ax + b = 0 \tag{2}$$

has the solution $x = -\frac{b}{a}$ if $a \neq 0$ and the equation

$$f(x) = ax^2 + bx + c = 0 \tag{3}$$

has the solutions $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ if $a \neq 0$. Solutions for cubic polynomial equations has a much more difficult formula, and it has also a technical problem: to apply it, we must use complex arithmetics and (unless we use tricky methods, and probably a computer algebra system as well) approximated calculations. It is not surprising that the Newton-Raphson method gives a faster way for finding a good approximation of the roots, i.e. the following process will usually provide one of the roots very soon:

$$f(x) = ax^3 + bx^2 + cx + d = 0, f'(x) = 3ax^2 + 2bx + c, x_{n+1} = x_n - \frac{ax_n^3 + bx_n^2 + cx_n + d}{3ax_n^2 + 2bx_n + c}. \tag{4}$$

There are many ways to illustrate the speed of this convergence. Probably one of the smartest is to use a spreadsheet software (e.g. LibreOffice Calc) and create the following table:

	A	**B**	**C**	**D**	**X**
1	1	2	−3	4	−3
2					$=X1-(A\$1*X1\hat{}3+B\$1*X1\hat{}2+C\$1*X1+D\$1)/(3*A\$1*X1\hat{}2+2*B\$1*X1+C\$1)$

To emphasize the coefficients of f we used the **A1**, **B1**, **C1** and **D1** cells for a, b, c and d respectively, and we hided the columns between **E** and **W** to show focus on column **X** (which is for the x_n sequence). As there is no 0th row, we put the starting point to x_1 (instead of x_0) which will be shown as the **X1** column. **X2** contains the general calculation formula for x_n with relative and absolute references as well. To summarize the above example, our table shows the definitions to calculate one of the roots of $f(x) = x^3 + 2x^2 - 3x + 4$ in the neighborhood of -3.

After evaluating **X2** and copying its contents dynamically into the cells below it, setting the precision to 15 decimals (and widening the column size), we get the following table:

	A	**B**	**C**	**D**	**X**
1	1	2	−3	4	−3.000000000000000
2					−3.333333333333330
3					−3.285403050108930
4					−3.284278150031770
5					−3.284277537307130
6					−3.284277537306950
7					−3.284277537306950
8					−3.284277537306950
9					−3.284277537306950

The Reader can verify that the last four elements of sequence in the table are equal, at least in numerical sense.

Theoretical considerations can prove that the speed of convergence becomes quadratic for this process, if the approximation is close enough to the limit. Quadratic convergence means that the number of the correct decimals in the above sequence is approximately doubling in each step. I.e., after the integer part in x_1 and x_2 has 0 correct digits, but x_3 has 2, x_4 has 5, and x_5 has 11.

Unfortunately, x_1 has to be selected carefully. Surprisingly enough, putting e.g. 3 into it, we get a slightly different process:

	A	B	C	D	X
1	1	2	−3	4	3.000000000000000
2					−1.888888888888890
3					1.088816972312120
4					1.193955741621468
5					1.852001854926000
6					1.058953573619320
7					0.134294516934989
8					0.880535307620495
9					−0.380554160315144

and so on, the sequence got mad, seems to become chaotic. Indeed, $x_1 = 3$ is far from the only root of f, but how do we know what is far enough here?

There is nothing special in this example above, this is the normal case. There are *many* initial points for x_1 which generate strange behavior. Two points $\left(x_1 = \frac{-2\pm\sqrt{13}}{3}\right)$ will surely stop the process by generating a "division by zero" error, due to $f'(x) = 0$, but there are also infinitely many points which result the same problem. By non-polynomial examples for f we can also obtain divergence to infinity for x_n as well.

3. All work and no play makes Jack a dull boy

Calculations are important in every day work, but to find interesting relationship between initial points we need to play. It is not easy to find out when a given x_1 is capable of finding a root if we play only the real axis. To learn about the background of this process we need the complex plane. Practically this will mean that we select x_1 not only from the real line, but from anywhere from $\mathbb{C}$.

So let us do the following. Paint the complex plane using the following method:

- If a selected x_1 generates a well behaving process (i.e. convergence to a root) then let us paint x_1 to a light color.
- If x_1 generates a fast convergence, it should be lighter than usual.
- If x_1 generates something different (divergence or "division by zero"), it should be black.

Of course, for an exact painting we need to give precise definitions for the coloring algorithm. But this is just a small detail, the structure of such a picture will be very similar to Figure 1.

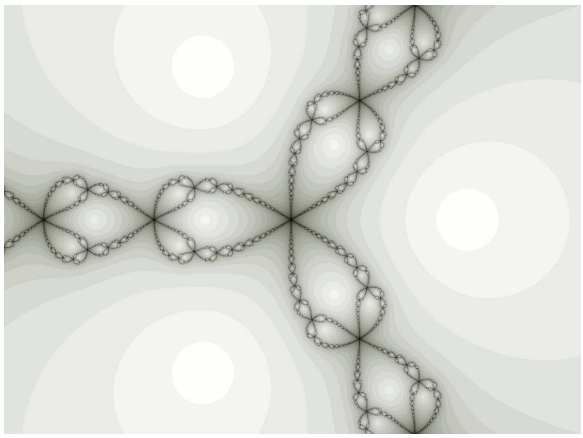

Figure 1: The cubic Newton fractal

The Reader can see that the complex plane is divided into many parts, in which there is a different speed of convergence for the different points, and the parts are bounded by thread-like creatures which are connected by black knots. In short: this kind of calculation has a difficult geometry. But it is interesting enough for investigation and to learn why this happens during the Newton-Raphson process.

To tell the truth, the above figure is not for the previous $f(x)$, but a different one. It is for the $f(x) = x^3 - 1 = 0$ equation. The three white circles show the neighborhood of the third complex unit roots, and the center of the figure is for the origin of the co-ordinate system. However this seems to be the simplest case for the cubic polynomials, its geometry is complicated enough to learn the general behavior of the Newton-Raphson method, or, in other words, to explore the cubic Newton fractal.

3.1. The $x^3 - 1$ case

Having a closer look on the Newton fractal, one can easily consider that there are infinitely many knots in the picture. It is not difficult to find them by calculating the 'bad' x_1 numbers, for which in a given step n, $f'(x_n) = 0$, and in the next step, the process will stop due to division by 0.

Obviously, if $x_1 = 0$, $f'(x_1) = 3x_1^2 = 0$, so x_2 cannot be calculated. Thus, 0 is a knot. No other 'bad' points can be found which cause x_2 not able to be calculated, because the equation $f'(x_1) = 3x_1^2 = 0$ has the only solution $x_1 = 0$.

Now let us search for those x_1 points which cause x_3 not able to be calculated. For that, clearly $x_2 = 0$ is a necessary and sufficient condition:

$$0 = x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = x_1 - \frac{x_1^3 - 1}{3x_1^2} \tag{5}$$

which implies $3x_1^3 = x_1^3 - 1$, and so $x_1 = \sqrt[3]{-\frac{1}{2}}$. Due to working on the complex plane, this will give three different knots for x_1:

$$\begin{aligned}
(x_1)_1 &= \sqrt[3]{\frac{1}{2}}\left(\cos\frac{\pi}{3} + i\sin\frac{\pi}{3}\right), \\
(x_1)_2 &= -\sqrt[3]{\frac{1}{2}}, \\
(x_1)_3 &= \sqrt[3]{\frac{1}{2}}\left(\cos\frac{\pi}{3} - i\sin\frac{\pi}{3}\right).
\end{aligned} \tag{6}$$

These points can also be observed as vertices of an equilateral triangle (which has its center in the origin). These three points, plus the origin give us four different knots until now.

But there are other knots as well. To find those x_1 points which cause x_4 unable to be calculated, we must check $x_3 = 0$ which means

$$0 = x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = x_2 - \frac{x_2^3 - 1}{3x_2^2}, \tag{7}$$

so we obtain

$$\begin{aligned}
(x_2)_1 &= \sqrt[3]{\frac{1}{2}}\left(\cos\frac{\pi}{3} + i\sin\frac{\pi}{3}\right), \\
(x_2)_2 &= -\sqrt[3]{\frac{1}{2}}, \\
(x_2)_3 &= \sqrt[3]{\frac{1}{2}}\left(\cos\frac{\pi}{3} - i\sin\frac{\pi}{3}\right).
\end{aligned} \tag{8}$$

One can easily see that x_1 and x_2 has a cubic polynomial relationship, which is exactly $x_2 = x_1 - \frac{x_1^3-1}{3x_1^2}$, that is $2x_1^3 - 3x_2x_1^2 + 1 = 0$. This gives at most three different solutions for x_1 for each fixed x_2. Thus, at most 9 different x_1 points exist for which x_4 cannot be calculated.

Continuing this reasoning, one can see that in general there will be at most 3^{m-1} different x_1 points for which x_{m+1} cannot be calculated. Further investigations show that for each m positive integer number there are exactly 3^{m-1} different x_1 points for which the Newton-Raphson method fails due to division by zero at x_{m+1}. This means, of course, that there are infinitely many knots for the $f(x) = x^3 - 1$ case.

3.2. Exploring fractals

Fortunately, there are several software tools for exploring this new world generated by such simple formulas. One of the most easy-to-use is XaoS [4], the free fractal engine which makes it possible for the user to view the world of fractals in real-time zooming.

XaoS allows to visualize the common basins of convergence by using the coloring method described above. Two functions can be observed, $f(x) = x^3 - 1$ and $f(x) = x^4 - 1$. By using the left mouse button, the user can zoom into any part of the fractal. By zooming, the user can verify by her experience that the knots are dense at the connection of each thread-like creatures, moreover these creatures are the infinite network of the knots. Self-similarity can also be well studied by utilizing this software, by gathering experience (see Figure 2).

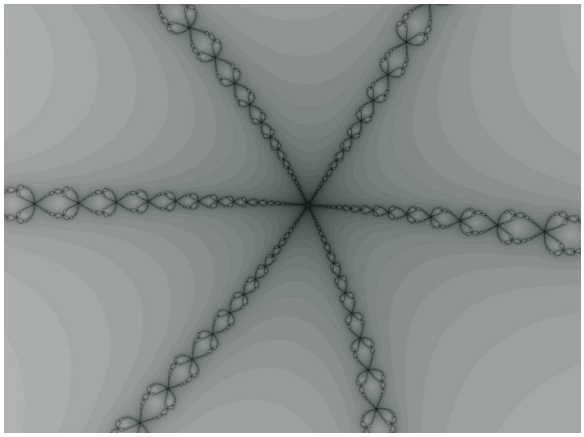

Figure 2: Close-up of the cubic Newton fractal at $-0.223 + 0.124i$

Another approach for visualization is Sage [5], the free mathematics software system. The following Sage code will generate the Newton fractal for an arbitrary $f(x)$ function.

```
f(x)=x^3-1

def iteration(x1,epsilon,maxiter):
    xi=x1
    for i in range(maxiter):
        dfi=N(diff(f,x).substitute(x=xi))
        fi=N(f(xi))
        if dfi<>0:
            xn=xi-fi/dfi
            if abs(xn-xi)<epsilon:
                return i
        else:
            return maxiter+1
        xi=xn
    return maxiter+2

xmin=-2; xmax=2; ymin=-2; ymax=2; epsilon=0.1; size=50; maxiter=10
```

```
18
19 m=matrix(size,size)
20 for i1 in range(size):
21     for i2 in range(size):
22         x1=N((xmin+i1*(xmax-xmin)/size+0.0)+(ymin+i2*(ymax-ymin)/size+0.0)*i)
23         m[i2,i1]=maxiter+2-iteration(x1,epsilon,maxiter)
24 matrix_plot(m)
```

This program calculates the iteration process for a set of initial points by calling the `iteration` subroutine with three parameters: `x1` is the starting x_1 to be checked, `epsilon` is the convergence threshold and `maxiter` is the number of maximal iterations. The subroutine returns the needed number of iterations for x_1 to approximate the speed of convergence. If it is not possible within `maxiter` steps, `maxiter`$+1$ is returned. If a 'division by zero' occurs, the subroutine returns `maxiter`$+2$.

The calculation process is much slower than in XaoS, and some tweaks were also needed in lines 6, 7 and 22 for forcing numerical calculations instead of symbolic ones. For testing convergence the Cauchy criteria was used (line 10). Figure 3 shows the Sage output for $f(x) = x^3 + 2x^2 - 3x + 4$, assuming `xmin=-4; xmax=4; ymin=-4; ymax=4; epsilon=0.1; size=400; maxiter=30` in line 17.

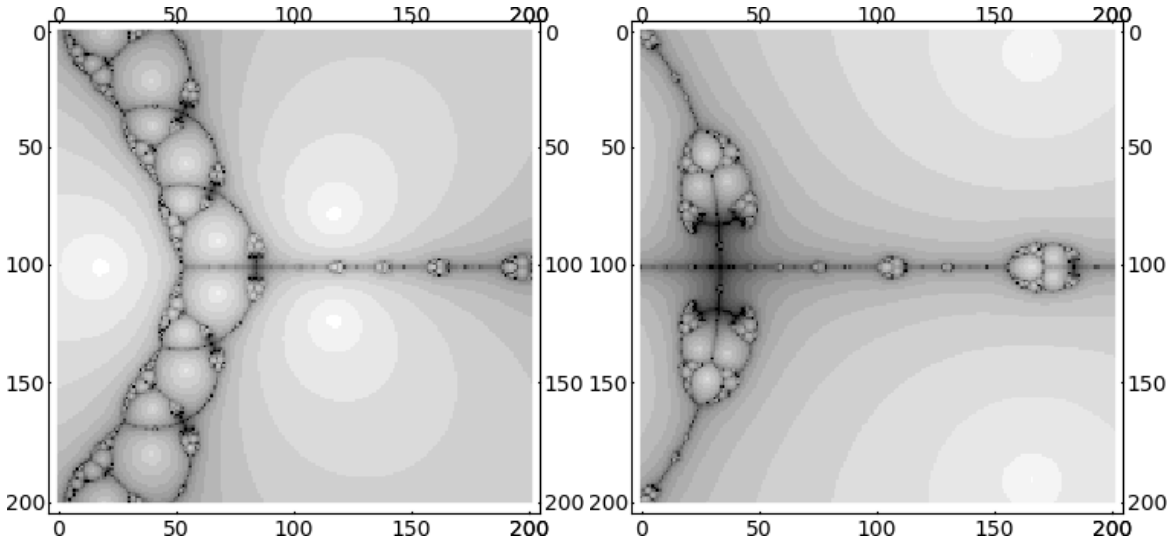

Figure 3: Sage output for $f(x) = x^3+2x^2-3x+4$ in the square determined by vertices $4+4i$, $-4+4i$, $-4-4i$, $4-4i$ and $1+i$, $-1+i$, $-1-i$, $1-i$

The Reader may try her own functions and generate the appropriate fractals. An Internet search should also offer many beautiful images created by various researchers. One of the best galleries can be seen at [6].

4. The multivariate Newton-Raphson method

The above program listing can be developed to show a generalized result for the Newton-Raphson process for multivariate equation solving.

To describe our problem by more variables, we can write

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0, \\ f_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0. \end{aligned} \tag{9}$$

Here $x_1, x_2, \ldots, x_n$ are to be searched. (Usually more solutions may exist.) We will use the short notations

$$\begin{aligned} \mathbf{x} &= \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \\ \mathbf{0} &= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \\ \mathbf{f}(\mathbf{x}) &= \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{pmatrix}, \end{aligned} \tag{10}$$

where all vectors consist of n lines of components. Now our problem in (9) can be shortly written as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{11}$$

For solving (11) by the multivariate Newton-Raphson method it is convenient to define

$$J(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{pmatrix}, \tag{12}$$

usually called Jacobian matrix of $\mathbf{f}(\mathbf{x})$. Now the multivariate Newton-Raphson method is defined by a "well chosen"

$$\mathbf{x}_0 = \begin{pmatrix} x_{0,1} \\ x_{0,2} \\ \vdots \\ x_{0,n} \end{pmatrix} \tag{13}$$

initial point and the generated sequence of

$$\mathbf{x}_1 = \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,n} \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} x_{2,1} \\ x_{2,2} \\ \vdots \\ x_{2,n} \end{pmatrix}, \quad \vdots \tag{14}$$

based on the $\mathbf{x}_0$ n-dimensional point. The calculation process is similar to (1) that can be described by

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (J(\mathbf{x}_n))^{-1} \cdot \mathbf{f}(\mathbf{x}_n). \tag{15}$$

(Note that (1) is a special case of (15) for $n = 1$.) Now we expect that the limit of this sequence (if exists),

$$\mathbf{x}_\infty = \begin{pmatrix} x_{\infty,1} \\ x_{\infty,2} \\ \vdots \\ x_{\infty,n} \end{pmatrix}, \quad \vdots \tag{16}$$

is a solution of (11), or in other words (see (9)),

$$\begin{aligned} f_1(x_{\infty,1}, x_{\infty,2}, \ldots, x_{\infty,n}) &= 0, \\ f_2(x_{\infty,1}, x_{\infty,2}, \ldots, x_{\infty,n}) &= 0, \\ &\vdots \\ f_n(x_{\infty,1}, x_{\infty,2}, \ldots, x_{\infty,n}) &= 0 \end{aligned} \tag{17}$$

is true.

This process is shown by a Sage application, similar to the univariate case.

```
var('x,y')
f=matrix(2,[sin(x)-y,y^2+x^2-1])
J=matrix(SR,2,2)
for i in range(2):
    J[i:i+1,0:]=jacobian(f[i:i+1,0:1],(x,y))

def iteration(x1,y1,epsilon,maxiter):
    xi=matrix(RDF,2,[x1,y1])
    for i in range(maxiter):
        xi1=xi[0,0]
        xi2=xi[1,0]
        Ji=N(J.substitute(x=xi1,y=xi2))
        fi=N(f.substitute(x=xi1,y=xi2))
```

```
14            if Ji.det()<>0:
15                xn=xi-Ji.inverse()*fi
16                if (xn-xi).norm()<epsilon:
17                    return i
18            else:
19                return maxiter+1
20            xi=xn
21        return maxiter+2
22
23 xmin=-5; xmax=5; ymin=-5; ymax=5; epsilon=0.1; size=200; maxiter=10
24
25 m=matrix(size,size)
26 for i1 in range(size):
27     for i2 in range(size):
28         x1=N(xmin+i1*(xmax-xmin)/size+0.0)
29         y1=N(ymin+i2*(ymax-ymin)/size+0.0)
30         m[i2,i1]=maxiter+2-iteration(x1,y1,epsilon,maxiter)
31 matrix_plot(m)
```

The code tries to follow our notations described above, but some technical tricks here are also needed. We fill in the J matrix with derivatives of **f** in line 5. Lines 28–29 work with normal coordinates for $\mathbb{R}^2$.

Our first test cases are $f_1(x,y) = \sin x - y$, $f_2(x,y) = x^2 + y^2 - 1$. Figure 4 shows the output for the first view and suggests that there are two solutions for this equation system.

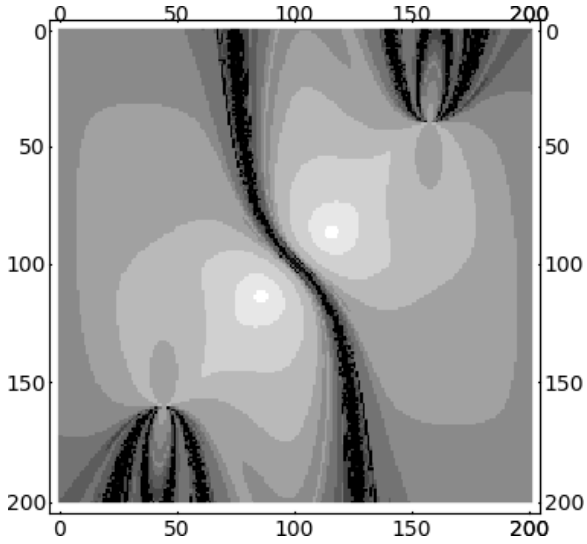

Figure 4: Test case #2 (see the benchmarking table for details) for the multivariate Newton-Raphson method with corners $(-5,-5), (5,-5), (5,5)$ and $(-5,5)$

If we create a plot of f_1 and f_2 (Fig. 5) it is obvious that these two solutions are the intersections of the sine function and the unit circle, $(x,y) = (0.88954, 1.79128)$ and $(x,y) = (-0.88954, -1.79128)$.

Fig. 6 shows a detailed view of an interesting kind of behavior near $(3,3)$.

Another investigation was to find intersection points of the Tschirnhausen cubic [7] and a parabola. Two of its convergence graphs can be seen in Fig. 7.

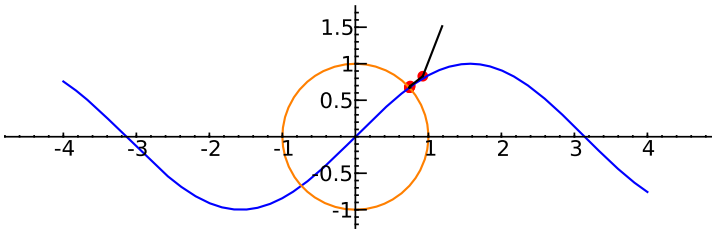

Figure 5: Intersections of the sine function and the unit circle. The black polyline and the red points show some steps of the iteration

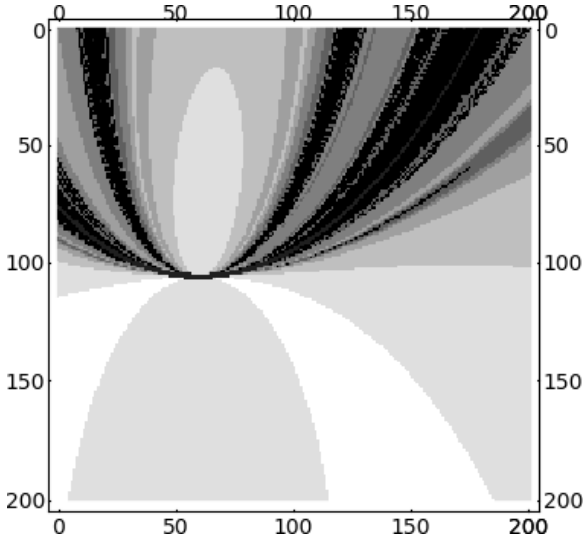

Figure 6: Test case #3 with corners $(2.5, 2.5), (3.5, 2.5), (3.5, 3.5)$ and $(2.5, 3.5)$

4.1. Speed issues

Test	$f_1(x,y)$	$f_2(x,y)$	$x_{\min}$	$x_{\max}$	$y_{\min}$	$y_{\max}$	ε	`size`[1]	`maxiter`	**Time**
#1	$\sin x - y$	x^2+y^2-1	-5	5	-5	5	0.01	21	20	13
#2	$\sin x - y$	x^2+y^2-1	-5	5	-5	5	0.01	201	20	1163
#3	$\sin x - y$	x^2+y^2-1	2.5	3.5	2.5	3.5	0.01	201	20	1261
#4	$y^2-x^3-3x^2$	x^2-y-4	0	1	0	1	0.01	11	10	4
#5	$y^2-x^3-3x^2$	x^2-y-4	0	1	0	1	0.01	101	10	325
#6	$y^2-x^3-3x^2$	x^2-y-4	0	1	1	2	0.01	201	15	2021

This table shows that the obtained figures require a remarkable amount of time to be generated by Sage. (The used architecture was an Intel Pentium 4 2×2.8 GHz with 2 GB of RAM, Debian 5.0 Linux and Sage 4.1.1 installed.) Time is measured in seconds, the benchmarking output has been provided by the `time` command in Sage.

There may be additional speedups to be done (e.g. changing Line 14 to check if $|\det J| < \varepsilon$) but it seems that for getting as fast output as XaoS gives we require a different technology in the background. Of course, to get first impressions about the geometry of the multivariable Newton-Raphson process, Sage gives an acceptable output. The Reader may read more on comparing the speed of Sage to other computer algebra systems (Mathematica 7, Magma and Matlab 2009a) at [8].

[1] Due to a bug in `matrix_plot` in Sage PNG output, such sizes must be used to get nicely labeled axes.

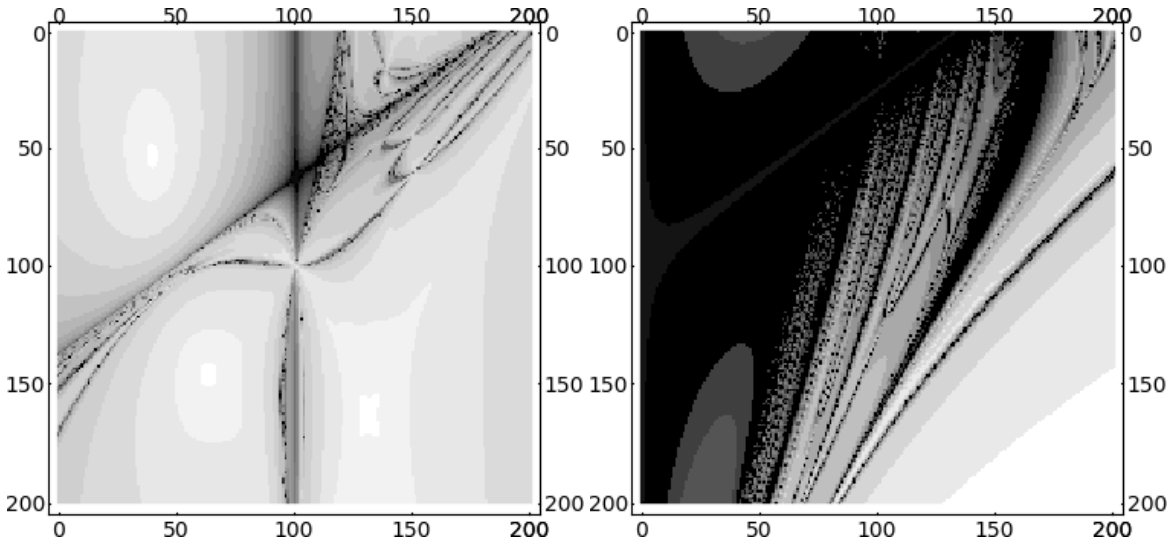

Figure 7: Basins of convergence and divergence of the Newton-Raphson method for finding intersection points of the Tschirnhausen cubic and a parabola with corners $(-4,-4),(4,-4),(4,4),(-4,4)$ and $(0,1),(1,1),(1,2),(0,2)$ (test cases #5 and #6)

5. Stability and some notes on theoretical background

The above calculation shows that it is difficult to predict how the approximation sequence behaves, even for quite simple problems. However, it is obvious that a large set of well behaving initial points can be given for each question. Such a question we can also consider as a *dynamical system* which is defined by the sequence generator formula and the initial point. In other words, to find the cube root of a given complex number, we can take a dynamical system which generates different sequences for different initial points.

The set of initial points that generate a convergent sequence are called *stable* points of the dynamical system. Stable points are important for practical reasons, but other points can also be interesting for mathematical artists. Indeed, non-stable points have unusual geometrical properties, and are exciting to explore.

Theoretical considerations can lead to remarkable results in finding general estimations for the speed of convergence. For example, if the derivative of the f function is always non-zero in a large enough neighborhood of the initial point, and the second order derivative of f is bounded, theory can ensure that the initial point will move on a well predictable orbit. This means that if we choose the initial point close enough to a root of f, we will get the root fast enough for practical use. The Reader can find more in [1] and [2] on this. Some highlighted details of the proofs and further output figures can be viewed on [3].

6. Conclusion

Fractal geometry plays an important role in studying mathematical background of every day life processes. The Newton-Raphson method is an often used technology in calculating roots of equation systems. Its convergence graph, showing the “good” starting points and the “bad” ones, is both an interesting visualization for the sciencists, and also a warning for the end users of the applied numerical methods to use the underlying technology carefully.

Fractal software developers may investigate how the multivariate Newton-Raphson can be implemented to be reasonably faster than usual CAS methods to obtain the geometrical views in a shorter time.

7. Acknowledgement

The author thanks Róbert Vajda for his suggestion on starting research on visualizing the multivariate Newton-Raphson process, and for his suggestions to make the text of this article better.

References

[1] F. Móricz, An Introduction to Numerical Mathematics (In Hungarian, *Bevezetés a numerikus matematikába*). Polygon, Szeged, 2008.

[2] F. Móricz, Numerical Methods in Algebra and Analysis, (In Hungarian, *Numerikus módszerek az algebrában és analízisben*). Polygon, Szeged, 1997.

[3] Z. Kovács, Numerical and Computer Algebra Methods in Analysis (In Hungarian, *Numerikus és számítógép-algebrai módszerek az analízisben*), web lecture notes, 2003. Available at: `http://www.math.u-szeged.hu/~kovzol/n2003t/numerikus-2003.php`. Accessed April 29, 2011.

[4] J. Hubička, T. Marsh, Z. Kovács, J.B. Langston et al., XaoS, computer software, available at `http://xaos.sf.net`

[5] Wikipedia contributors, Sage (mathematics software). *Wikipedia, The Free Encyclopedia. February 11, 2011, 17:03 UTC.* Available at: `http://en.wikipedia.org/wiki/Sage_(mathematics_software)`. Accessed February 15, 2011.

[6] P. Bourke, Gallery Of Fractals Created Using The Newton Raphson Method, 1989, available at `http://local.wasp.uwa.edu.au/ pbourke/fractals/newtonraphson/`

[7] Wikipedia contributors, Tschirnhausen cubic. *Wikipedia, The Free Encyclopedia. November 29, 2010, 03:50 UTC.* Available at: `http://en.wikipedia.org/wiki/Tschirnhausen_cubic`. Accessed February 27, 2011.

[8] Sage, Tour—Benchmarks. Available at: `http://www.sagemath.org/tour-benchmarks.html`. Accessed February 27, 2011.