SIGSAM Bulletin (ACM Press), December 2000, **134**, 16-32

Numerical Monsters

Christopher Essex, Matt Davison, and Christian Schulzky

Department of Applied Mathematics, University of Western Ontario, Canada N6A 5B7

Abstract: When the results of certain computer calculations are shown to be not simply incorrect but *dramatically* incorrect, we have a powerful reason to be cautious about *all* computer-based calculations. In this paper we present a "Rogue's Gallery" of simple calculations whose correct solutions are obvious to humans but whose numerical solutions are incorrect in pathological ways. We call these calculations, which can be guaranteed to wreak numerical mayhem across both software packages and hardware platforms, "Numerical Monsters". Our monsters can be used to provide deep insights into how computer calculations fail, and we use them to engender appreciation for the subject of numerical analysis in our students. Although these monsters are based on well-understood numerical pathologies, even experienced numerical analysts will find surprises in their behaviour and can use the lessons they bring to become even better masters of their tools.

1 Introduction

In a 1950's science fiction short story, "The Feeling of Power", Isaac Asimov [1] introduced a world where people had become so totally dependent on computers and calculators that they were completely amazed by a person who had rediscovered how to do arithmetic without one. In fact, as the story unfolded, this forgotten human capability became a deep military secret, because there were advantages for humans to think after all.

In the 1980s, a crafty experiment [2] was performed on school children and adults which demonstrated that many people have developed a blind dependence on computing devices. The unwitting subjects were provided with rigged calculators that were prepared to give systematically wrong answers. Incorrect results from these rigged calculators were contrasted with those from correct hand calculations also done by the same subjects. Only a minority chose to believe their own correct hand calculations over the wrong ones produced with the calculators.

Of course, as any numerical analyst knows, one doesn't need to rig a calculator to get wrong answers. Wrong answers are just part of the nature of finite computers running afoul of the infinities of mathematics. Even the inhabitants of Asimov's fictional world were spared this fundamental reality of computing devices.

A blind acceptance of what computers provide, is no surprise to anyone who teaches an introductory course on numerical analysis. Initially, it is often a near thing just to convince beginning students, steeped in the myth of perfect computation, that there is any value to the subject at all. Advanced students can backslide into fancying that perfect computation can be found in computer algebra environments. Those who teach computation must take students from these science fiction mindsets into the realm of reality. Computers are so mysterious, fast, and consistent, it is very hard to imagine the truth: mathematics is too big to fit into computers.

This paper introduces a proactive strategy for introducing important numerical analysis issues. Rather than discussing how to do things to avoid numerical problems, it seeks numerical trouble. This has something of the character of work by Kahan, e.g. [3], in the conscious search for numerical difficulties. The goals are however somewhat different. There is no aim to seek things that need fixing, but instead it is to help people understand that computers are not absolute, and that they cannot be by their nature.

This approach exploits the modern graphical environments, even in those originating in computer algebra packages, putting different platforms and packages into a common context. In a computer algebra environment, it means exploring pathologies at a given precision. Observing that precision can be set to a higher level misses the point that finite computers cannot be set to calculate with infinite infinite precision. Moreover the graphical environment is simply fun for experienced and inexperienced people to begin with the computational analog to a demolition derby. It leads into spectacular computational monsters, which despite their brazen nature raise subtle issues that may even prove challenging to research numerical analysts.

We make no claim to a comprehensive treatment of these "numerical monsters", either in terms of what monsters are possible or in terms of all aspects of their nature, which turn out to be wonderfully complex and subtle. The goal here is merely to encourage the search for different and interesting types of these creatures. In doing so we suggest some features that ought to be looked out for when encountering them, and how they might be usefully incorporated into curricula.

We begin with a treatment of simple round-off beasts which lead to spurious stripes and false zeros. The pivot monster shows how these can be connected to basic numerical analysis issues such as why one needs to pivot. This is taken to a higher level in the wide class of bow-tie monsters, which open up many subtle forms, revealing not only uniformity across platforms, but also fascinating differences, reflecting underlying differences in how numerics are treated. This opens the door for more advanced discussion of programing and numerical issues.

2 Tiger Stripes and Machine Epsilon

The function

$$T(x) = \mathrm{e}^{x} \ln(1 + \mathrm{e}^{-x}), \tag{1}$$

is mathematically very well behaved. It is positive, with a strictly positive but decreasing slope. It is easy to show that $T(x) = \mathrm{e}^{x}\left(\mathrm{e}^{-x} - \mathcal{O}(\mathrm{e}^{-2x})\right)$ for large x, clearly going to 1 in the limit $x \to \infty$. This means that the function rises steadily to 1 for all x.

However, numerically, this function is very badly behaved, in that the function that is actually plotted by the computer, T_c, is dramatically different than $T(x)$. T_c does not rise steadily for increasing x for all x, as can be seen in figure 1, which arises from the IEEE arithmetic of MAPLE 6. There, for $x \approx 30$, the upward trend falters and T_c drops spuriously to zero in a flurry of apparent noise, departing from its proper value of nearly 1. However in figure 2, which is a close up of this spurious behaviour, it is apparent that it is not noise at all but a systematic structure in the form of regular "tiger" stripes, immediately proceeding the spurious collapse to zero of the computed function.

This simple function gives us not only an error of order 1, but in addition the stripes are explicable, and the point where the function drops to zero is a direct consequence of machine epsilon, and completely predictable. Machine epsilon is the smallest value ϵ for which $1 + \epsilon \neq 1$. It gauges how small a number will not quite disappear due to round off. In this sense $\epsilon/2$, one shift operation smaller, will be the largest number that behaves as zero when added to 1. We conclude that $\mathrm{e}^{-x} = \epsilon/2$ where the function drops spuriously to zero, or $x = -\ln(\epsilon/2)$.

This value is easily confirmed by taking the first zero position straight off the plot and comparing it to the value $x = -\ln(\epsilon/2)$, where $\epsilon = 2.2204 \times 10^{-16}$ is the machine epsilon provided in MATLAB 5.

This is a powerful exercise for students to see that the seeming chaos of the tiger stripes comes to a conclusion at an entirely predictable point.

The stripes themselves are closely related. They also arise from round off in that rounding causes $1 + \mathrm{e}^{-x}$ to behave as a constant over progressively longer intervals as x approaches $-\ln(\epsilon/2)$. That is, the changes in e^{-x} are too small to show up in comparison to 1, until certain critical values of x are reached. Each of these values leads to downward jumps in the sum, inducing a stepping behaviour. These steps when multiplied by e^{x} lead to a sequence of exponentials, which appear as stripes. Note that these stripes can even exceed 1, contrary to the nature of the function $T(x)$ that the computer was supposed to plot.

This is an example of pathological behaviour that is far from noise, and which has a form that we can understand and predict. This particular numerical monster is quite robust, and will show up in very similar form in virtually any numerical environment. Note at this point that figure 1 shows the tiger stripes in MAPLE 6 and the closeup in figure 2 was produced with MATLAB 5, but these plots would be virtually identical on comparable

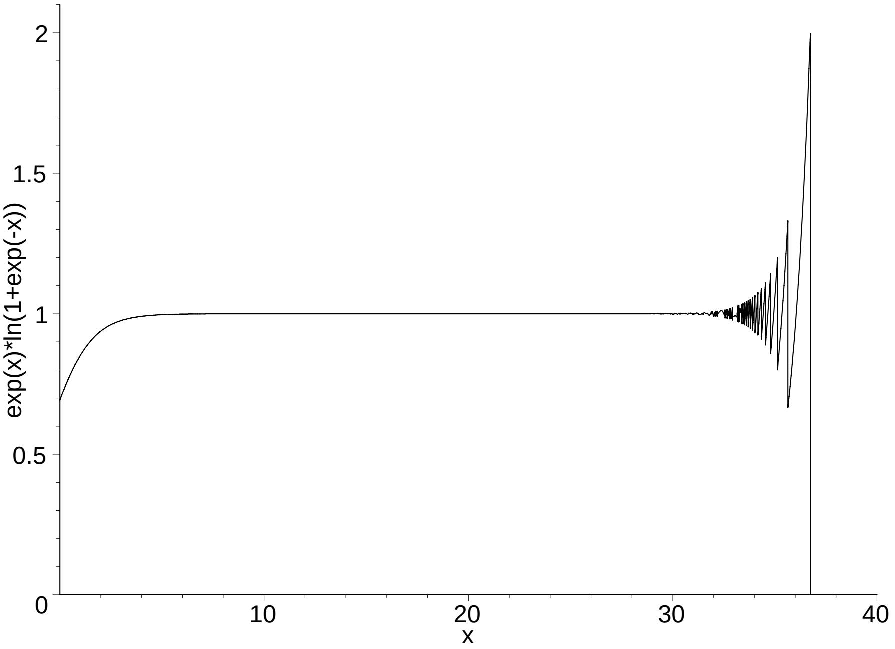

Figure 1: The plot of the function $T(x) = \mathrm{e}^{x}\ln(1 + \mathrm{e}^{-x})$ with the computer algebra package MAPLE 6. For $x > 30$ a dramatic deviation from the theoretical value 1 is found. For $x > 36.7$ the plotted value drops spuriously to zero.

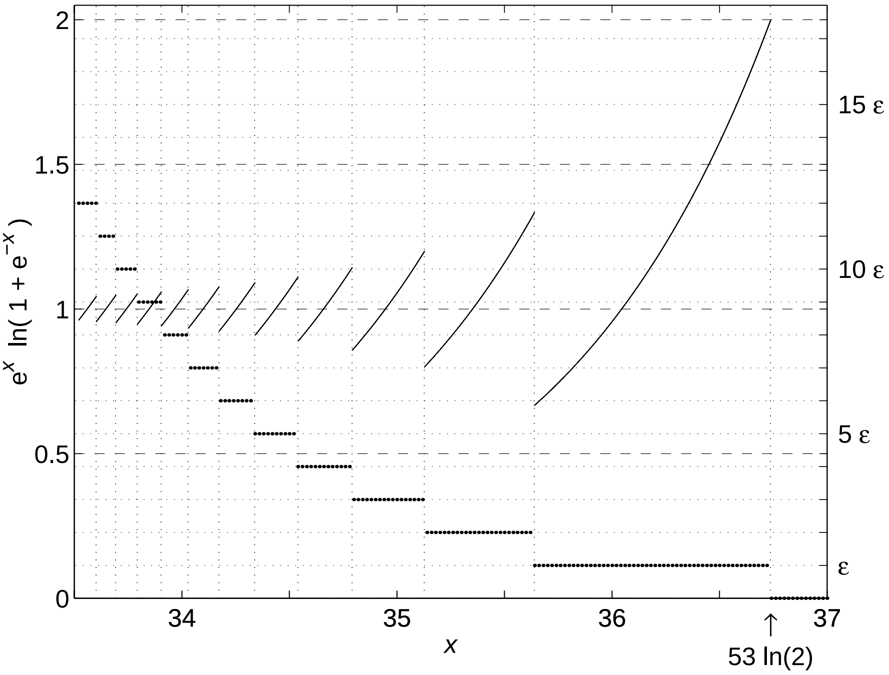

Figure 2: A close up of the apparent noise in figure 1. Although the plot deviates from the theoretical result $T(x) \approx 1$ in this regime, it shows a regular behaviour which we name tiger stripes. The spurious collapse to zero is denoted by the arrow. The function $\ln(1 + e^{-x})$ is also plotted (dots), exhibiting false plateaus of increasing length. This explains the tiger stripes, which are therefore just exponential functions. Note that this plot was created with MATLAB 5.

ranges. That is T_c is nearly identical for both MATLAB 5 and MAPLE 6, even though these functions both differ from $T(x)$. In subsequent examples we will see that monster invariance will not hold across software packages, even if broad structures will be common.

A simple refinement of this monster becomes a useful and quick way to read off effective precision. That is if instead of (1) we plot

$$T(x) = 2^x \ln(1 + 2^{-x}). \tag{2}$$

In the case of MATHEMATICA 4.0 figure 3 shows a drop to zero at $x = 53$, implying 52 bits of precision. Points were intentionally connected to emphasize the drop to zero. The resulting value is confirmed by computing 2^{-52} and comparing to Mathematica's value ϵ.

In addition to expression (2) we plot

$$T(x) = 2^x \ln(1 - 2^{-x}). \tag{3}$$

The graph of equation (3) appears as a negative version of (2), except that (3) collapses to zero one shift operation later than (2). That means that in some sense an approach to zero by subtraction is more precise than the approach through addition.

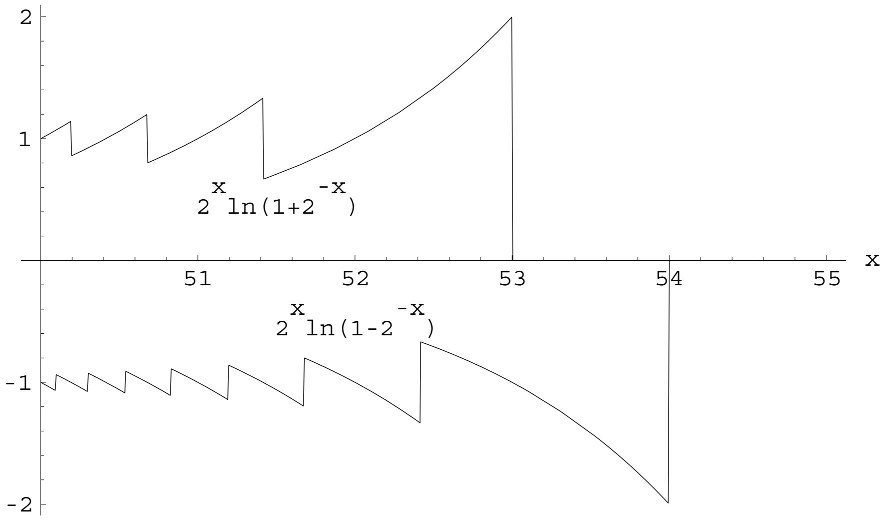

Figure 3: The functions $2^x \ln(1 \pm 2^{-x})$ are shown. The drop to zero for $x = 53$ implies 52 bits of precision. The collapse to zero at $x = 54$ indicates a higher effective precision for subtraction.

This is remarkable in that a common presumption is that subtraction is numerically more problematic through the traditional worry about the difference of large numbers. The present case seems mysteriously contrary to that. However the explanation is rather disappointingly obvious in that $1+\epsilon$ will involve only the lowest order digits of 1. To tell the difference from 1 then depends on the highest precision, while in contrast $1-\epsilon$ will involve all digits in 1 because of carrying operations. Subtraction in this case clearly manifests itself across all digits, so while adding expands the demand for precision subtraction, contrary to the cliche, reduces it. In particular it provides one more shift operation of precision.

This simple example provides a strong introduction to many details of round off error. It also shows that numerical noise isn't really noise, but is instead a well defined, reproducible, and generic property of floating point computation across many platforms.

3 The Pivot Monster

Pivoting during LU decomposition is accepted as a crucial practice, but it is not often demonstrated why it is essential to extract a correct answer from a computer. A simple explanation is rarely forthcoming in many presentations, which typically offer instead examples of how, rather than why, to do it. We provide here a numerical monster that comes directly from an LU decomposition that has not been pivoted. Based on the results of the previous section, all of the properties of this monster can be easily understood. In many respects it is a simpler monster than the tiger, but it is essentially the same.

We begin with the following matrix system

$$\begin{pmatrix} \delta & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1+\delta \\ 2 \end{pmatrix} . \tag{4}$$

LU decomposition of the coefficient matrix without pivoting is possible, leading to

$$\begin{pmatrix} \delta & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{1}{\delta} & 1 \end{pmatrix} \begin{pmatrix} \delta & 1 \\ 0 & 1-\frac{1}{\delta} \end{pmatrix} . \tag{5}$$

This form leads to the new system

$$\begin{pmatrix} \delta & 1 \\ 0 & 1-\frac{1}{\delta} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1+\delta \\ 1-\frac{1}{\delta} \end{pmatrix} . \tag{6}$$

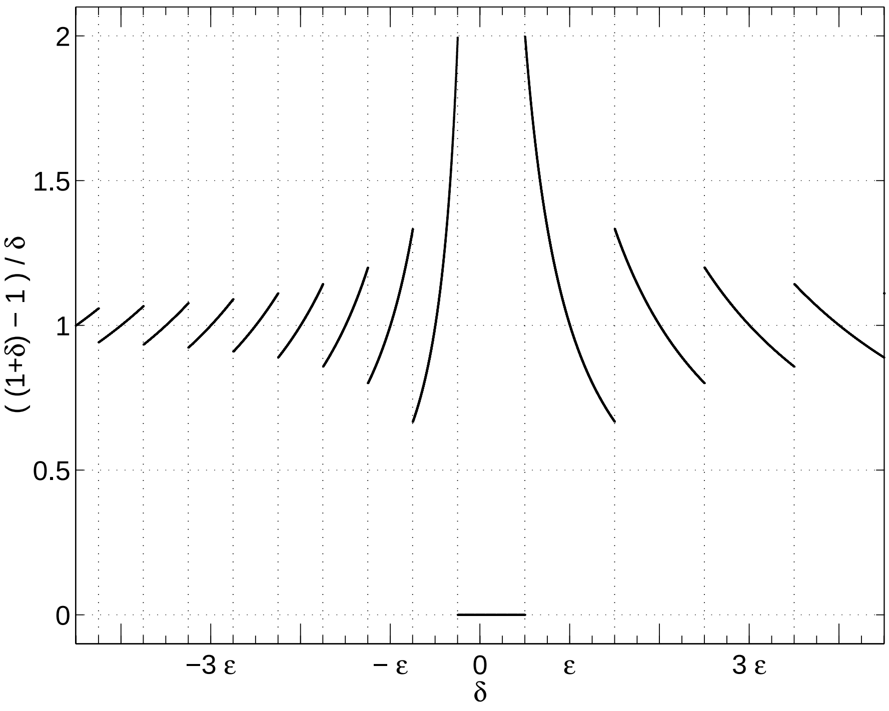

Figure 4: The expression for x_1 in equation (7) is plotted for δ near 0. Instead of having the constant value 1 the function shows a tiger-stripe like structure. It drops spuriously to zero for $\delta \approx 0$. Note that the distance between the ticks on the δ-axis is $\epsilon/4$, with $\epsilon = 2.2204 \times 10^{-16}$ for MATLAB 5.

This equation provides the solution

$$\begin{aligned} x_2 &= 1 \\ x_1 &= \tfrac{(1+\delta)-1}{\delta} \ . \end{aligned} \tag{7}$$

The expression for x_1 is clearly equal to 1, independent of the value of δ. However a floating point process will compute the expression as presented, as it represents here a sequence of numerical operations on specific numbers instead of a general algebraic expression to be transparently simplified. The algebraic form here simply makes clear what would not otherwise be clear in a floating point LU decomposition in the absence of pivoting. Pivoting makes an almost trivial change on the corresponding expressions, which are of course algebraically equivalent. However they are numerically very different.

One simple pivot leads to the trivially equivalent matrix system

$$\begin{pmatrix} 1 & 1 \\ \delta & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1+\delta \end{pmatrix}, \tag{8}$$

which produces the algebraically equivalent solutions

$$\begin{array}{l} x_2 = \frac{1-\delta}{1-\delta} \\ x_1 = 2 - \frac{1-\delta}{1-\delta}. \end{array} \tag{9}$$

However these are very different in that they can be plotted without difficulty, while x_1 in expression (7) is a simple but close relative to the numerical tiger. A simple plot (figure 4) near $\delta = 0$ shows that it has stripes of its own. Equation (9) will not have these, as the δ-terms have the value of 1, even for floating point calculations. x_1 for (7) however is not 1 in floating point arithmetic as can be seen from figure 4.

The stripes here differ from that of the tiger stripes in that they are not exponentials but hyperbolas instead. That is because the numerator in (7) takes on constant values while δ changes in digits that are lost in the sum $1 + \delta$. As before when the change in δ grows large enough, the result will finally show up in $1 + \delta$ and restart a new hyperbola.

We see the same asymmetry from figure 3 appearing here: the spurious collapse of the expression to zero occurs exactly one shift operation later for negative than for positive values. This is another occurrence of a case where subtraction introduces less error than addition.

All of the properties of the original tiger are here, and they are clearly the source of error introduced in the absence of proper pivoting. With this example it is transparent why pivoting is crucial to accurate computation.

Closely related to the tiger and the pivot is another monster where an algebraic simplification is not that obvious. Consider the function

$$f(x) = \frac{\sqrt{x+9} - 3}{x} . \tag{10}$$

A Taylor expansion about $x = 0$ results in

$$f(x) = \frac{1}{6} - \frac{x}{208} + \cdots , \tag{11}$$

which yields the constant value 1/6 near $x = 0$. Figure 5 shows a plot of this function, which we call the mustache, between -2×10^{-14} and 2×10^{-14}. Again we find a variation of the tiger strips discussed in the previous section instead of the constant value 1/6.

These simple round off monsters might seem inconsequential, outside of introductory examples, because of the small scale of the errors arising. While in subsequent classes of monsters, this need not be so, it is worth pointing out that even in the last of the discussed cases, computer division by $f(0)$, for example, can lead to division by zero, instead of a multiplication by 6 — a nontrivial error!

4 Bow Tie Monsters

In the previous section we demonstrated that the order in which calculations are performed can become very important numerically. The bow tie numerical monsters depend on this. Consider the trivial binomial formula,

$$(x-1)^2 = x^2 - 2x + 1 \; . \tag{12}$$

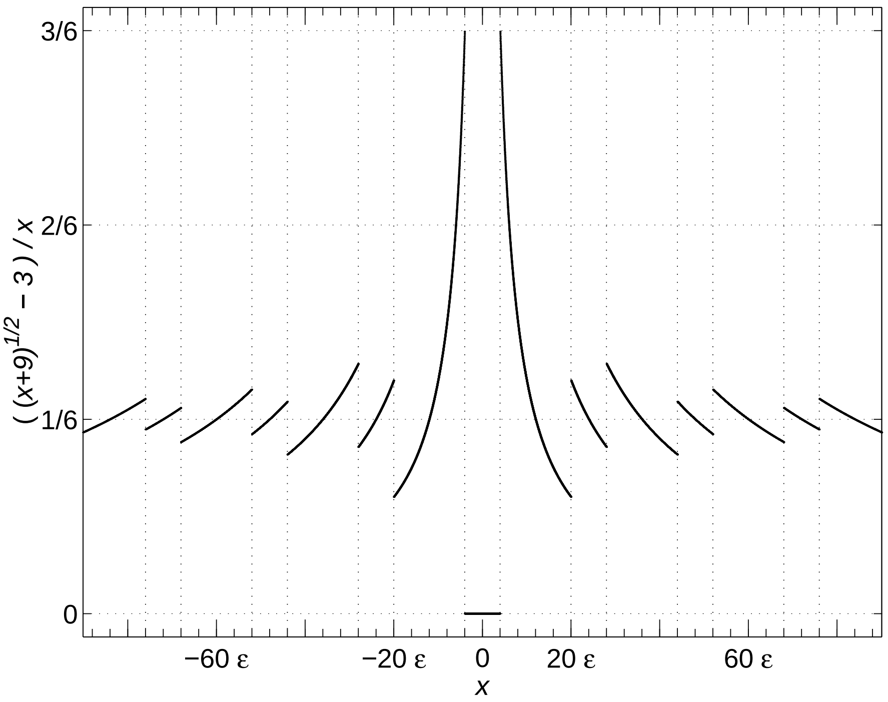

Figure 5: The function $f(x) = (\sqrt{x+9} - 3)/x$ is plotted for x near 0. Instead of the mathematical function which is close to 1/6 given by the Taylor expansion (11), we get one that has the character of a mustache of tiger stripes. Here the distance between the ticks on the x-axis is 4ϵ.

What happens if we investigate the difference between these two expressions on a computer.

Figure 6, which reminds us of a bow tie, shows the difference

$$(x^2 - 2x + 1) - (x - 1)^2$$

plotted in terms of x—1. Though the value is indeed zero most of the time, we also encounter substantial deviations from zero in a very systematic manner. These can be made arbitrarily large depending only on the distance from $x = 1$. In certain intervals of $x - 1$ the function alternates between zero and two values of the same magnitude but different signs.

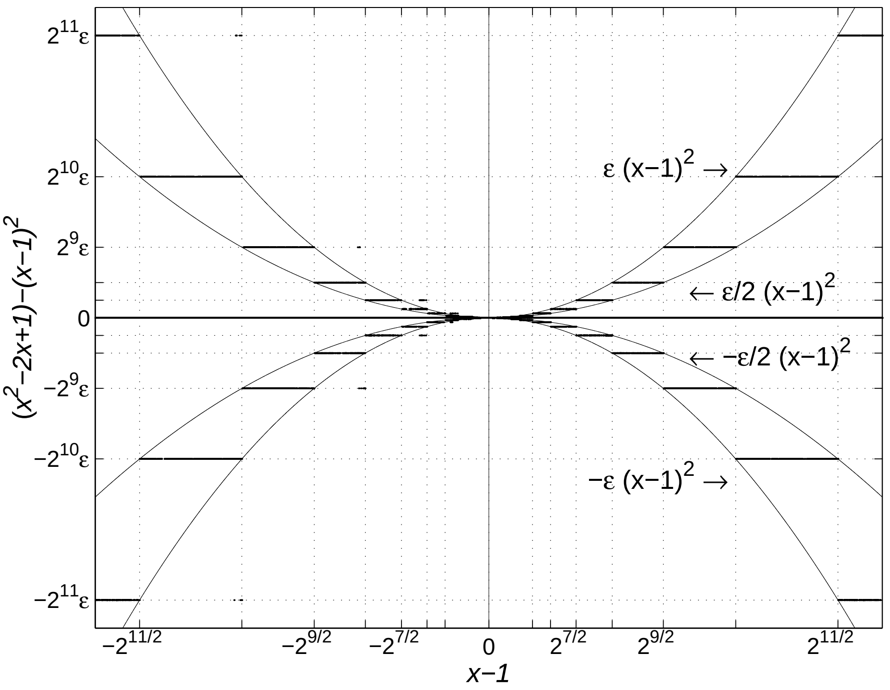

Figure 6: The difference $(x^2 - 2x + 1) - (x - 1)^2$ plotted using MATLAB 5. Instead of zero, a ladder-like structure is found in the bow tie shape. For the width of a rung, the values of the plotted difference function jump between three values — two with the same magnitude, but with opposite signs, and zero. Note that there are also a few "escape" points which pathologically escape the simple but otherwise rigorous envelopes. Note also the extraordinary pervasiveness of ϵ in all features of the graph.

These values change by a factor of two in going from one interval to the next. This leads to a ladder like structure. Amazingly the steps of the ladder are bounded by the functions $\pm\epsilon(x-1)^2$ and $\pm\frac{\epsilon}{2}(x-1)^2$, which give the bow-tie-like appearance. Note that the actual deviations can be expressed in multiples of machine epsilon ϵ. As ϵ drops to zero clearly all of these features will vanish.

To investigate the ladder in more detail, note that the length of successive intervals increases by $2^{1/2}$. This suggests a log-log plot of the bow tie monster. For the sake of simplicity we consider a blow up of the first quadrant (see figure 7).

For larger x we find the perfect linear ladder structure we expected. But for decreasing x-values, more and more deviations, like the escape points mentioned in figure 6, can be seen. This leads to a bifurcation-like behaviour that ends up in a noisy and chaotic structure for $x - 1 < 2^{-3}$. Thus a transition between two different regimes showing this monster to be more

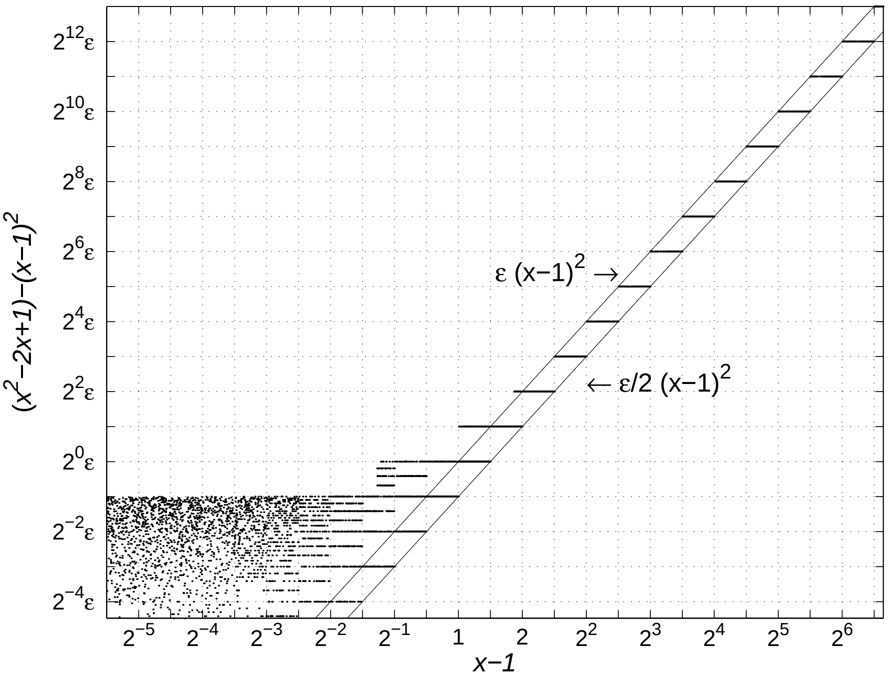

Figure 7: A close up of the first quadrant of figure 6 in a log-log plot. This shows existence of two different regimes: a ladder like, and a fairly chaotic one.

monstrously subtle than a first glance might suggest.

Now let us zoom into the chaotic regime and have a closer look at the vicinity of $x = 1$. Figure 8 reveals lots of structure even in this noisy, but structured, domain.

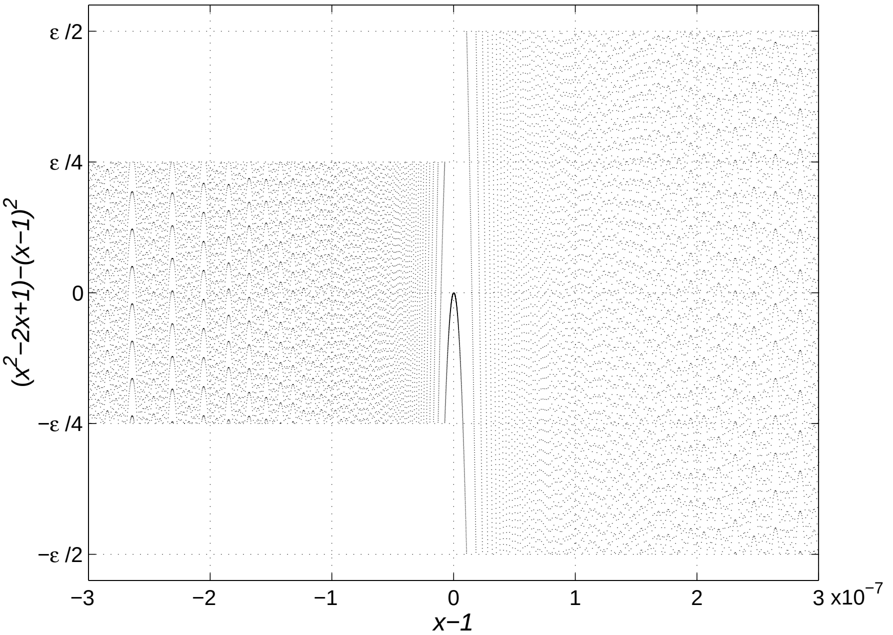

Figure 8: A closer look into the noisy regime of the bow tie. Note the asymmetry about $x - 1 = 0$.

Besides the obvious asymmetry between the left and right sides of the plot, which can be traced back to the shift operations discussed in connection with the tiger stripes in section 2, the plot has an amazingly self-similar character. An explanation for this and a connection to chaos is beyond the scope of this article, and will be left to future work. Instead we focus on the parabola near $x = 1$.

In figure 9 the two different parts of equation 12 are plotted separately. The expanded form shows steps, as in the case of the logarithm discussed in figure 2. Thus for intervals with a length of order ϵ the function $x^2 - 2x + 1$ is constant whereas the part that is written as a perfect square does not show such step behaviour. Therefore the sum of both parts leads to a tiger-stripe-like structure.

Instead of deepening the discussion of these curious properties in such

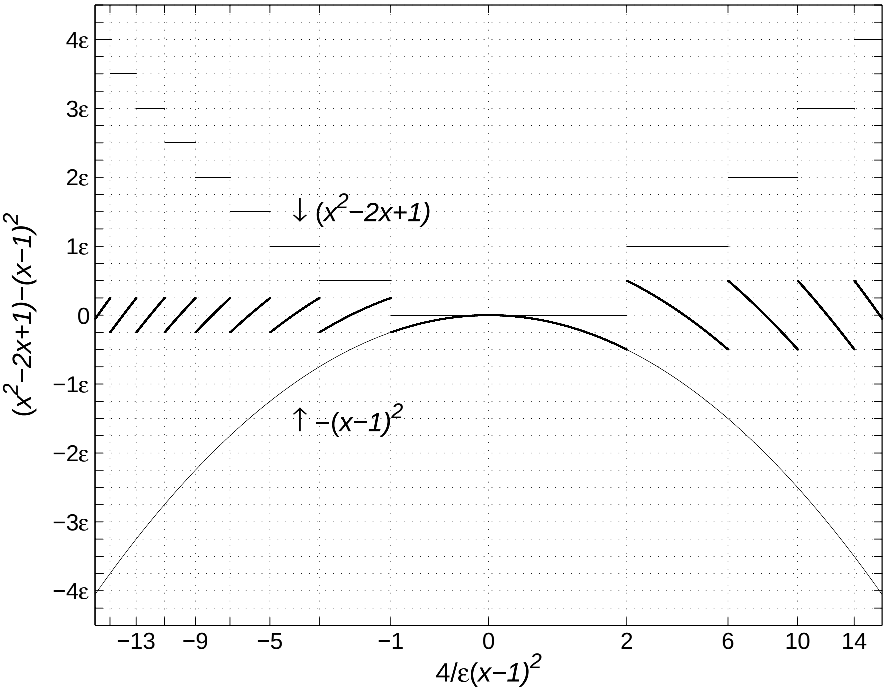

Figure 9: The bow tie monster around $x = 1$. A discussion of the separate parts of the difference reveals similar tiger stripes to those of figure 2. Note that the previously discussed asymmetry is observed here again.

a simple plot, we note that the bow tie monster can also be observed using other software packages. But interestingly, in contrast to the tiger class of monsters, fascinating differences emerge.

For example, investigating the same difference, $(x^2 - 2x + 1) - (x - 1)^2$, with MAPLE 6 leads to a very similar result to that above using MATLAB 5 (see left hand side of figure 10). However changing the evaluation order of the two terms, i.e. $(x - 1)^2 - (x^2 - 2x + 1)$, leads to a qualitatively different appearance of the monster. This degree of non-associativity (this reversal is not commutation although one may be tempted to use that term) was not observed with MATLAB 5 for this monster. Note that even with the different appearance, the envelope and ladder structure is preserved.

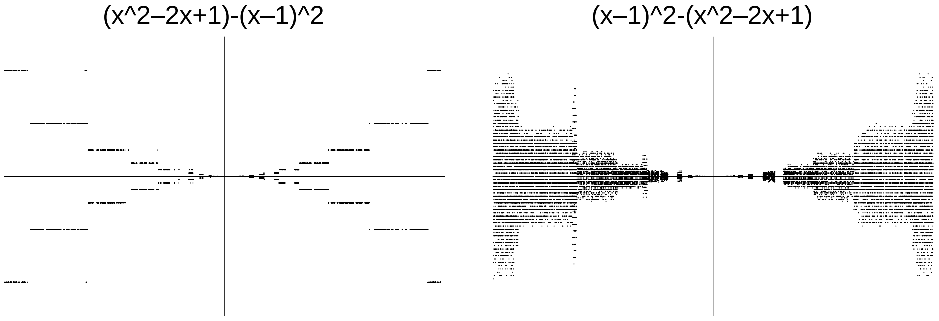

Figure 10: The bow tie monster plotted with MAPLE 6. Changing the order of calculation of the two terms leads to a surprisingly different appearance for the monster. For the sake of clarity the axes labels are omitted. All magnitudes correspond to those given in figure 6.

5 Trig Tie Terror

The techniques found in the last section carry over to other functions too. This leads to a wide class of new and different monsters. Take, for example, the Taylor expansion of the sine

$$\sin(x) = x - \frac{1}{6}x^3 + \mathcal{O}(x^5) \ . \tag{13}$$

Plot the difference between $\sin(x)$ and the first two terms of the expansion. Figure 11 shows a plot of $\sin(x) - (x - x^3/6)$ about $x = 0$ created with MAPLE 6. The graph, although distinctive in its own right, clearly has much in common with the quadratic bow tie. The partially filled in domains within the bounding envelope have some particular similarity to the right hand side of figure 10, which was also plotted with MAPLE 6. However, the detailed structure has a very different character, appearing more stochastic, than the right side of figure 10.

In any case in figure 11 we see again the error regions building up a step structure, where the steps are bounded by the envelopes with the form of the originating function multiplied by ϵ and a power of 2: $\pm\frac{\epsilon}{2}\sin(x)$ and $\pm\frac{\epsilon}{4}\sin(x)$. The same difference plotted in MATHEMATICA 4.0 gives just the steps, but without the filled regions, and the bounding functions are $\pm 2\epsilon \sin(x)$ and $\pm\epsilon/2 \sin(x)$ instead. This suggests different internal routines, even though these routines are clearly all subject to the same fundamental constraints. A curious observation is that for both of the software packages, the monster vanishes if $(x - x^3/6) - \sin(x)$ is plotted instead! That is the re-

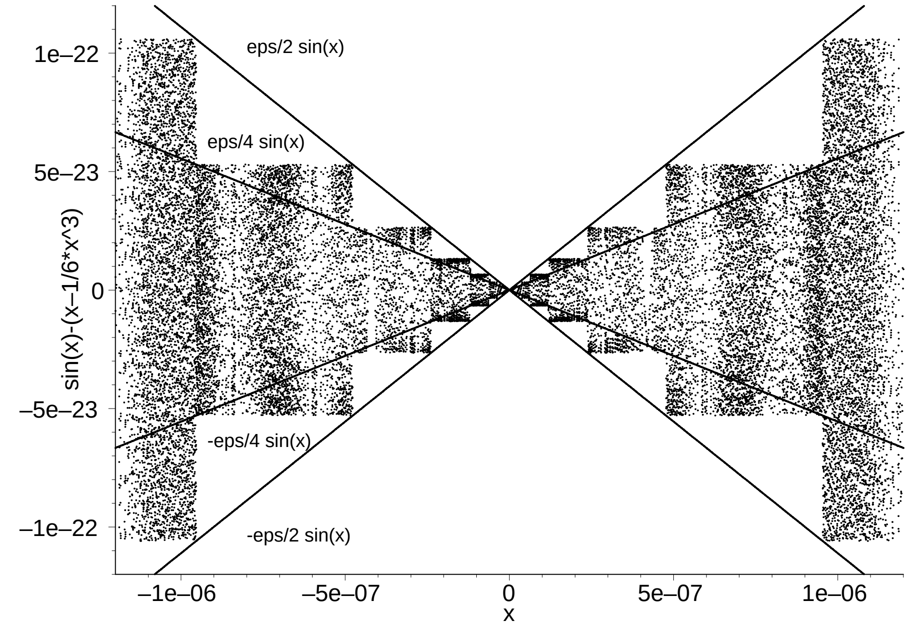

Figure 11: The difference between the sine and its Taylor expansion. Note the similarity to the right hand side of figure 10 and the steps are bounded by the same type of envelope as the polynomial bow tie.

sulting plot shows simply the zero-line. Moreover in MATLAB 5 this monster does not appear at all, independent of the order of the terms!

We see that these distinct bow tie versions of $\sin(x)$ with different software packages appear to have a common standard form, even if they differ in details. The standard form is instructive for beginners, while the differences should be a matter of interest for more advanced investigations as well. In the latter case not only can one use these differences to explore the distinctions between packages and platforms, we can create many different bow-tie-like structures for many different functions, each with their own particular features.

One might think, that just considering the first two terms in the Taylor expansion is not sufficient. With computer algebra packages, shouldn't we always be going to precise high order representations, free from round off? This sounds fine until one tries to plot it, then the numbers have to be rounded off anyway to jam them into the few pixels that one has to represent data. Figure 12 therefore is an effective way to show a beginner that computer algebra does not end the subject of numerical analysis. It is a

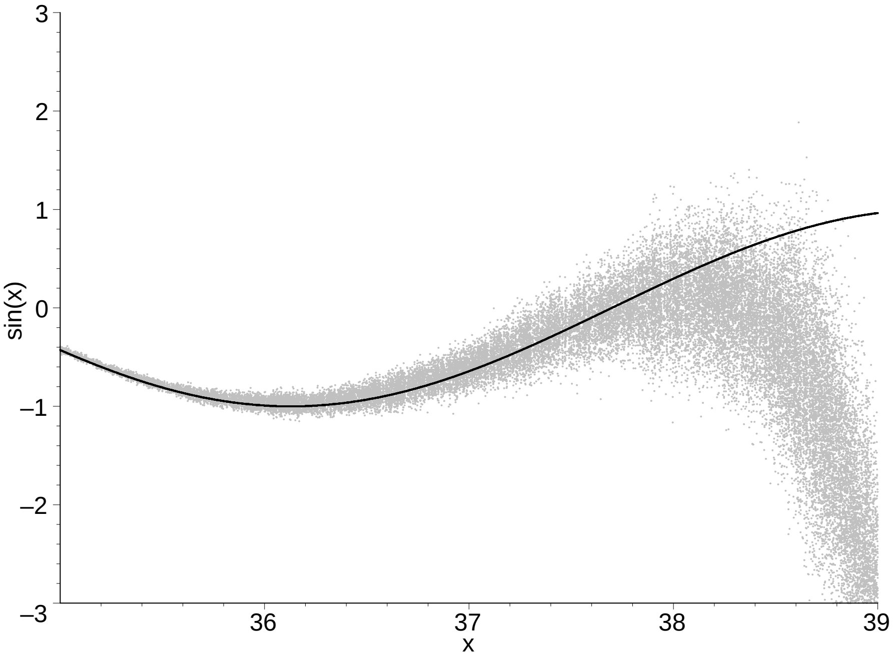

Figure 12: The Taylor expansion of the $\sin(x)$ up to order 100. In the depicted range the round off errors sum up to order one inducing the tornado-like structure.

plot of the $\sin(x)$ expansion up to order 100. Even in this case, where the algebraic potential of MAPLE 6 is exploited, we cannot use the increased accuracy of the expansion in plotting. This particular monster we call the tornado.

6 The Wall

Another area where predictable questions from students about numerical analysis arise is in connection with numerical derivatives: Can't we just take a smaller step size? Can't we just take more terms? These questions are easily addressed with the monster we call the Wall.

We consider the derivative of a function $f(x)$ numerically by setting up its Taylor expansion:

$$f(x_0 + kh) = \sum_{\nu=0}^{\infty} \frac{(kh)^\nu}{\nu!} f^{(\nu)}(x_0) \ , \tag{14}$$

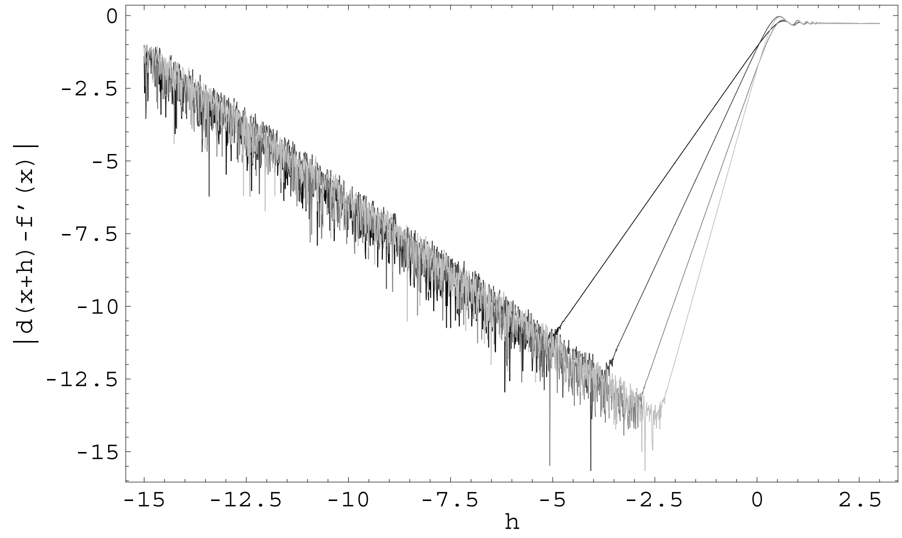

Figure 13: Numerical derivative error using different order formulas using Mathematica 4.0. The error is plotted (log-log) for the case $f(x) = \sin(x)$ at the expansion point $x_0 = 1$. The four curves are, from left to right, the cases where $n = 1$ ($k = 1$, eq.(15)), $n = 2$ ($k = -1, 1$, eq.(16)), $n = 3$ ($k = -1, 1, 2$), and $n = 4$ ($k = -2, -1, 1, 2$).

with the nonzero integer k. Consider n of these expansions, each with different values of k. Together they yield an $n + 1$-point formula for the first derivative. For example in the trivial case of $n = 1$ and $k = 1$ we find the well known prelimit form for the first derivative:

$$f'(x_0) \approx d_1(x_0, h) = \frac{f(x_0 + h) - f(x_0)}{h} \; . \tag{15}$$

For $n = 2$, where we select the two values for k of -1 and 1 we find the nearly as well known formula for the midpoint rule:

$$f'(x_0) \approx d_2(x_0, h) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} \; . \tag{16}$$

In figure 13 the error $|d_n(x_0, h) - f'(x_0)|$ for three cases is plotted (log-log) with respect to the step size h for $f(x) = \sin(x)$ and the expansion point $x_0 = 1$. For large h (i.e. > 1) we see oscillating behaviour. Decreasing the step size reduces the error for the first derivative as expected. As more points n are employed in the numerical approximation this rate of decrease is more rapid. In the upper right region of figure 13 the four curves depicted therefore correspond to higher order estimates from left to right.

The complication is, as is well known to numerical analysts, that the error goes back up again with decreasing step size. It makes a nice exercise for students to estimate the minimum error on the graph. But it is not so well known is that the noisy left side of the graph, which has the appearance of melting wax, is nearly the same line for all order schemes. While the higher order methods do have smaller minimum errors, they reach them for larger h, and they proceed upward along the same path.

In fact it *seems* that the error for the described method of numerical derivation can not be arbitrarily decreased, but has a lower limit independent of the order of the method. Figure 13 shows that all of the methods considered run into the same round off wall. That wall makes the lower left of the domain inaccessible, and it makes for interesting discussion in a classroom.

7 City Streets

As a last example of a monster for this introductory presentation, we consider the computation of the residual, $R(A)$, for a randomly generated 2x2 matrix for the problem $Ax = b$ with $b = (1, 0)^T$. Using MATLAB 5 the residual can be calculated in two different ways:

$$R_1(A) = A * (\mathrm{inv}(A) * b) - b \qquad \text{and} \qquad R_2(A) = A * (A \backslash b) - b \; , \tag{17}$$

where the second method uses the left division operation implemented in MATLAB 5. We use this alternative method because it produces an interesting result. In figure 14 the residuals $R_1(A)$ is denoted by "x" and $R_2(A)$ by "o". Instead of having the theoretical value of $(0, 0)^T$, these residuals take on values other than zero, as is to be expected in the presence of round off error.

If one examines the resulting plot on large enough scales, the points appear like a disorganized cloud, as one might expect with numerical noise. However we did not use such a scale for the figure. Instead the figure depicts these residual vectors on a small scale. On that scale these seemingly random deviations are all seen to fall on a rigorous rectangular grid, which leaves the impression of city streets seen from the air. The two types of residuals, even though they are numerically different, all fall on the exactly the same streets. These streets have a distance which is a multiple of $2^{-n}\epsilon$, where n

increases if one goes towards $(0,0)^T$. These errors vanish with ϵ. They are not random but are instead highly structured.

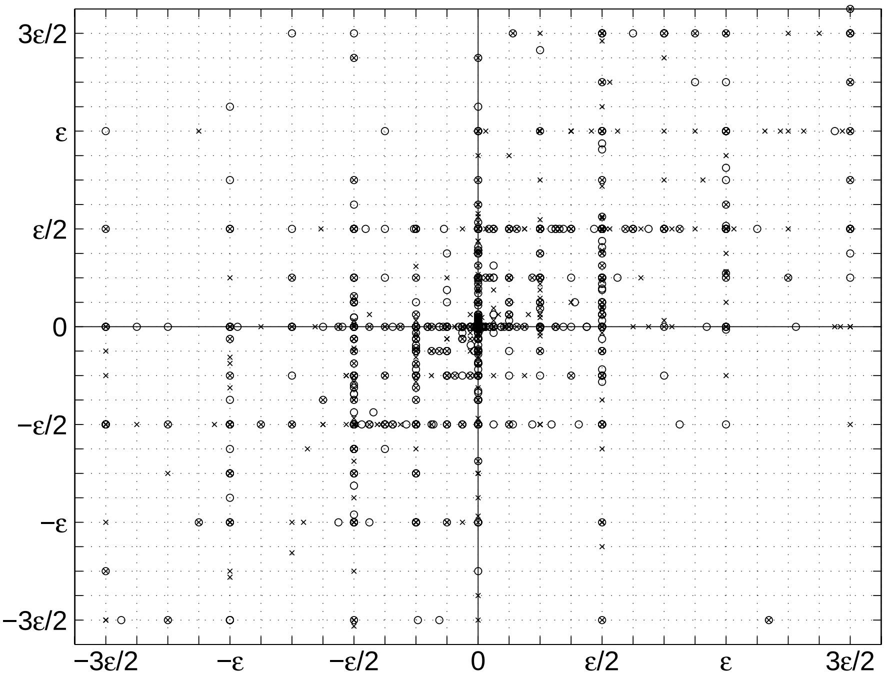

Figure 14: The residuals (17) of the matrix equation $Ax = b$ calculated with MATLAB 5. The two components of the residuals, $R_1(A)$ (denoted by "x") and $R_2(A)$ (denoted by "o"), are used as the axes. Note that all residuals lie on "streets" which are positioned as powers of 2 times ϵ.

8 Conclusion

Round off error is often described as noise. Indeed it is typically called numerical noise. However it is difficult to justify this usage on a computer system which is ostensibly deterministic. Nonetheless, as we have seen, calling it noise is not so unreasonable either. There is so much, systematic though it may be, that is quite mysterious without very determined efforts to root out its mechanisms, often some arcane combination of machine and software arithmetics.

This character makes it particularly problematic in explaining numerical error and there by motivating numerical analysis to beginners: yes, there is error in a computation on a computer; yes, it is deterministic in a human designed machine prepared to be that way; no, we don't know what the precise error is, so we only estimate the error or find bounds at best. We are forced to tell them that we must approximate something that they would rather not believe exists at all. And if it is finally accepted that computers have error, it is only because of a belief in an explicit cause. That is the moment when we usually announce that the error can now be treated as stochastic. The result is that beginners can be simultaneously mystified by and dismissive of this important subject.

This is further complicated by the designer's desire to create code for different arithmetics, or more adaptive levels of precision, each to more closely represent true mathematical behaviour. This leads to many often non-standard layers of complication between the hardware and the user, leading to inscrutable behaviours exhibited above. The modern user necessarily must work in many different computational environments. It is simply unrealistic to expect that the peculiarities of each will be more than superficially understood. Indeed many of these peculiarities must necessarily fall into the domain of current research.

The objective of this paper was to introduce a collection of simple calculations which are certain to create numerical mayhem in nearly any contemporary system in its plotting environment. The plotting environment was chosen for several reasons. The visual environment is becoming more and more important in practical applications for computational programs. It is also the great leveler between computer algebra environments and floating point ones, as all numbers, no matter how precise, must still be jammed into a not so large number of pixels. How these pixels are filled leads to the most idiosyncratic software arithmetics, as they are designed to meet the harsh demands of authentic mathematics in artificial graphical environments. Finally, a picture leads to instant understanding of these issues, for students and more experienced users alike.

The goal was two fold: first, to visually elucidate traditional numerical analysis issues such as machine epsilon, and the need for pivoting in matrix computation; second, to provide simple means to quickly assess idiosyncracies of any particular piece of computational software. The reader is cautioned in this latter regard not to seek "fixes" from fancy settings for a particular piece of software. The goal is to *understand* and *recognize* the numerical idiosyncracies. They will not be so easily recognized in real applications as they are in these test cases, even in a case where they may be just as significant. Moreover they can never be fully eliminated from finite computers.

For students and even more experienced users these numerical monsters may demonstrate in a knowable and simple manner just how wrong computations can be. There is much more to say about these monsters in future publications. It is hoped that more monsters will be collected, not only because of their practical value but also because they are fun.

References

[1] Isaac Asimov, "The Feeling of Power" in *Nine Tomorrows* Doubleday, New York 1959.

[2] Lois Timnik, "Electronic Bullies" in *Psychology Today* **16** 10-15, 1982.

[3] William Kahan, "Mathematics Written in Sand" *Proc. Joint Statistical Mtg. of the American Statistical Association*, 12-26, 1983.