The misfortunes of a trio of mathematicians using Computer Algebra Systems. Can we trust in them?[*,†]

Antonio J. Durán[1], Mario Pérez[2] and Juan L. Varona[3]

[1]Departamento de Análisis Matemático, Universidad de Sevilla, 41080 Sevilla, Spain
email: `duran@us.es`

[2]Departamento de Matemáticas, Universidad de Zaragoza, 50009 Zaragoza, Spain
email: `mperez@unizar.es`

[3]Departamento de Matemáticas y Computación, Universidad de La Rioja, 26004 Logroño, Spain
email: `jvarona@unirioja.es`

Abstract

Computer algebra systems are a great help for mathematical research, but sometimes unexpected errors in the software can also badly affect it. As an example, we show how we have detected an error in Mathematica when computing determinants of matrices with integer entries; not only does it compute the determinants incorrectly, but it also produces different results if one evaluates the same determinant twice.

Mathematics Subject Classification (2010): 68W30.

Introduction

Nowadays, mathematicians often use a computer algebra system as an aid in their mathematical research; they do the thinking and leave the tedious calculations to the computer. Everybody "knows" that computers perform this work better than people. But, of course, we must trust in the results derived via these powerful computer algebra systems. First of all, let us clarify that this paper is not, in any way, a comparison between different computer algebra systems, but a sample of the current state of art of what mathematicians can expect when they use this kind of software. Although our example deals with a concrete system, we are sure that similar situations may occur with other programs.

[*]Partially supported by grants MTM2012-36732-C03-02, MTM2012-36732-C03-03 (Ministerio de Economía y Competitividad), FQM-262, FQM-4643, FQM-7276 (Junta de Andalucía) and Feder Funds (European Union).

[†]This paper has been published in *Notices Amer. Math. Soc.* **61** (2014), 1249–1252.

We are currently using Mathematica to find examples and counterexamples of some mathematical results that we are working out, with the aim of finding the correct hypotheses and eventually constructing a mathematical proof. Our goal was to improve some results by Karlin and Szegő [4] related to orthogonal polynomials on the real line. The details are not important; this is just an example of the use of a computer algebra system by a typical research mathematician, but let us explain it briefly. It is not necessary to completely understand the mathematics, just to realize that it is typical mathematical research using computer algebra as a tool.

Our starting point is a discrete positive measure on the real line, $\mu = \sum_{n\geq 0} M_n \delta_{a_n}$ (where δ_a denotes the Dirac delta at a, and $a_n < a_{n+1}$), having a sequence of orthogonal polynomials $\{P_n\}_{n\geq 0}$ (where P_n has degree n and positive leading coefficient). Karlin and Szegő considered in 1961 (see [4]) the $l \times l$ Casorati determinants

$$\det \begin{pmatrix} P_n(a_k) & P_n(a_{k+1}) & \dots & P_n(a_{k+l-1}) \\ P_{n+1}(a_k) & P_{n+1}(a_{k+1}) & \dots & P_{n+1}(a_{k+l-1}) \\ \vdots & \vdots & \vdots & \vdots \\ P_{n+l-1}(a_k) & P_{n+l-1}(a_{k+1}) & \dots & P_{n+l-1}(a_{k+l-1}) \end{pmatrix}, \quad n, k \geq 0. \tag{1}$$

They proved that, under the assumption that l is even, these determinants are positive for all nonnegative integers n, k. Notice that the set of indices $\{n, n+1, \dots, n+l-1\}$ for the polynomials P_n consists of consecutive nonnegative integers. We are working out an extension of this remarkable result for more general sets of indices F than those formed by consecutive nonnegative integers. We have some conjectures which we want to prove or disprove.

We have not been able to prove our conjectures yet and, as far as we can see, this task seems to be rather difficult. On the other hand, just in case our conjectures are wrong, we have been trying to find counterexamples with the help of our computer algebra system. Eventually we hope these experiments can shed some light on the problem, as well.

We have then proceeded to construct orthogonal polynomials with respect to discrete positive measures (involving only a finite number of Dirac deltas, which is actually not a restriction for our conjectures) by means of their moments. Fixing a set of indices $F = \{f_1, \dots, f_l\}$, $f_i < f_{i+1}$, for the polynomials P_n, we have evaluated the determinants

$$\det \begin{pmatrix} P_{f_1}(a_k) & P_{f_1}(a_{k+1}) & \dots & P_{f_1}(a_{k+l}) \\ P_{f_2}(a_k) & P_{f_2}(a_{k+1}) & \dots & P_{f_2}(a_{k+l}) \\ \vdots & \vdots & \vdots & \vdots \\ P_{f_l}(a_k) & P_{f_l}(a_{k+1}) & \dots & P_{f_l}(a_{k+l}) \end{pmatrix} \tag{2}$$

for a large range of k, looking for some negative value.

To avoid the usual problems with floating point arithmetic (rounding, truncating, instability), we construct all our examples with integers. By taking integers as the values of a_n and the mass points M_n of the measure, and using a suitable normalization of the orthogonal polynomials P_n, only integers are involved in (2). Thus the computations should be routine for a computer algebra system, and one should be able to completely trust in the results. We have also introduced random parameters (also integers, of course) to easily perform many experiments.

With the help of Mathematica, one of us found some counterexamples to our conjectures. Fortunately, another one of us was using Maple and, when checking those supposed counterexamples, he found that they were not counterexamples at all. After revising our algorithms from scratch, we concluded that either the computations performed with Mathematica or the computations performed with Maple had to be wrong. Things started to become clear when the colleague using Mathematica also found some "counterexamples" to the above mentioned result of Karlin and Szegő for the case in (1) and, even more dramatically, his algorithm yielded different outputs given the same inputs. Our conclusion was that Mathematica was computing incorrectly. However, our mathematical problem (and our algorithm) was too complicated to convince anybody that Mathematica was making mistakes when calculating with integers.

Isolating the error

In attempting to isolate the computational problem, we finally realized that, in some circumstances, Mathematica (version 9.0.1 at that time) makes some strange mistakes when computing determinants whose entries are large integers. Errors do not always occur, only in some cases. Even worse, given the *same* matrix, the determinant function can give different values! This resembles the well-known Pentium division bug discovered by Thomas Nicely in 1994, which only affected certain kinds of numbers. But it seems Mathematica is a black box even darker that the internals of a microprocessor, so it is difficult to try to understand what kinds of numbers are affected by the Mathematica bug that we are describing.

Instead, we have devised a method to easily generate matrices with large integer entries whose determinants are clearly erroneously evaluated by Mathematica. This method can be described without referring to the mathematical problem which motivates it. As the error does not always arise, we develop a procedure to randomly generate these matrices. First, we generate a random 14×14 matrix whose entries are integers between -99 and 99,

```
basicMatrix = Table[Table[RandomInteger[{-99, 99}], {i, 1, 14}], {j, 1, 14}]
```

To obtain larger integers, we multiply every column by some power of 10. This is equivalent to multiplying by a diagonal matrix; for instance, we take

```
powersMatrix = DiagonalMatrix[{10^123, 10^152, 10^185, 10^220, 10^397,
  10^449, 10^503, 10^563, 10^979, 10^1059, 10^1143, 10^1229, 10^1319, 10^1412}]
```

To avoid getting only integers ending in many zeroes, we add a small random matrix given by

```
smallMatrix = Table[Table[RandomInteger[{-999, 999}], {i, 1, 14}], {j, 1, 14}]
```

Then, we take

```
bigMatrix = basicMatrix.powersMatrix+smallMatrix
```

(in Mathematica notation, the dot `.` is used to denote the product of matrices). Now, we compute the determinant twice:

```
a = Det[bigMatrix];
b = Det[bigMatrix];
```

Surprisingly, we quite often find that `a` and `b` contain different values! This is easily observed by checking whether `a==b`, which quite often returns `False`, or by visually comparing their numerical approximations `N[a]` and `N[b]`.

Let us see an instance of a real execution of these procedures: with

$$\texttt{basicMatrix} = \begin{pmatrix} -32 & 69 & 89 & -60 & -83 & -22 & -14 & -58 & 85 & 56 & -65 & -30 & -86 & -9 \\ 6 & 99 & 11 & 57 & 47 & -42 & -48 & -65 & 25 & 50 & -70 & -3 & -90 & 31 \\ 78 & 38 & 12 & 64 & -67 & -4 & -52 & -65 & 19 & 71 & 38 & -17 & 51 & -3 \\ -93 & 30 & 89 & 22 & 13 & 48 & -73 & 93 & 11 & -97 & -49 & 61 & -25 & -4 \\ 54 & -22 & 54 & -53 & -52 & 64 & 19 & 1 & 81 & -72 & -11 & 50 & 0 & -81 \\ 65 & -58 & 3 & 57 & 19 & 77 & 76 & -57 & -80 & 22 & 93 & -85 & 67 & 58 \\ 29 & -58 & 47 & 87 & 3 & -6 & -81 & 5 & 98 & 86 & -98 & 51 & -62 & -66 \\ 93 & -77 & 16 & -64 & 48 & 84 & 97 & 75 & 89 & 63 & 34 & -98 & -94 & 19 \\ 45 & -99 & 3 & -57 & 32 & 60 & 74 & 4 & 69 & 98 & -40 & -69 & -28 & -26 \\ -13 & 51 & -99 & -2 & 48 & 71 & -81 & -32 & 78 & 27 & -28 & -22 & 22 & 94 \\ 11 & 72 & -74 & 86 & 79 & -58 & -89 & 80 & 70 & 55 & -49 & 51 & -42 & 66 \\ -72 & 53 & 49 & -46 & 17 & -22 & -48 & -40 & -28 & -85 & 88 & -30 & 74 & 32 \\ -92 & -22 & -90 & 67 & -25 & -28 & -91 & -8 & 32 & -41 & 10 & 6 & 85 & 21 \\ 47 & -73 & -30 & -60 & 99 & 9 & -86 & -70 & 84 & 55 & 19 & 69 & 11 & -84 \end{pmatrix}$$

and

$$\texttt{smallMatrix} = \begin{pmatrix} 528 & 853 & -547 & -323 & 393 & -916 & -11 & -976 & 279 & -665 & 906 & -277 & 103 & -485 \\ 878 & 910 & -306 & -260 & 575 & -765 & -32 & 94 & 254 & 276 & -156 & 625 & -8 & -566 \\ -357 & 451 & -475 & 327 & -84 & 237 & 647 & 505 & -137 & 363 & -808 & 332 & 222 & -998 \\ -76 & 26 & -778 & 505 & 942 & -561 & -350 & 698 & -532 & -507 & -78 & -758 & 346 & -545 \\ -358 & 18 & -229 & -880 & -955 & -346 & 550 & -958 & 867 & -541 & -962 & 646 & 932 & 168 \\ 192 & 233 & 620 & 955 & -877 & 281 & 357 & -226 & -820 & 513 & -882 & 536 & -237 & 877 \\ -234 & -71 & -831 & 880 & -135 & -249 & -427 & 737 & 664 & 298 & -552 & -1 & -712 & -691 \\ 80 & 748 & 684 & 332 & 730 & -111 & -643 & 102 & -242 & -82 & -28 & 585 & 207 & -986 \\ 967 & 1 & -494 & 633 & 891 & -907 & -586 & 129 & 688 & 150 & -501 & -298 & 704 & -68 \\ 406 & -944 & -533 & -827 & 615 & 907 & -443 & -350 & 700 & -878 & 706 & 1 & 800 & 120 \\ 33 & -328 & -543 & 583 & -443 & -635 & 904 & -745 & -398 & -110 & 751 & 660 & 474 & 255 \\ -537 & -311 & 829 & 28 & 175 & 182 & -930 & 258 & -808 & -399 & -43 & -68 & -553 & 421 \\ -373 & -447 & -252 & -619 & -418 & 764 & 994 & -543 & -37 & -845 & 30 & -704 & 147 & -534 \\ 638 & -33 & 932 & -335 & -75 & -676 & -934 & 239 & 210 & 665 & 414 & -803 & 564 & -805 \end{pmatrix}$$

we got `N[a]` $= -3.263388173990166 \cdot 10^{9768}$ and `N[b]` $= -8.158470434975415 \cdot 10^{9768}$ and, executing the same program repeatedly, other values different from these. None of these values is the correct one, because the determinant of `bigMatrix` is, approximately, $1.95124219131987 \cdot 10^{9762}$.

We have found this erroneous behavior in Mathematica version 8 (released on November 15, 2010) up to version 9.0.1 (the latest version when the above mentioned experiments were done and the first version when this manuscript was submitted), both under Mac and Windows. It seems that it does not affect versions 6 and 7, at least in the same range of numbers.

We reported the bug on October 7, 2013 (reference CASE:303438), receiving a kind answer from Wolfram Research Inc.:

> It does appear there is a serious mistake on the determinant operation you mentioned. I have forwarded an incident report to our developers with the information you provided.
>
> We are always interested in improving Mathematica, and I want to thank you for bringing this issue to our attention. If you run into any other behavior problems, or have any additional questions, please don't hesitate to contact us.

By June 2014, nothing had changed. We had received similar replies in the past, when one of us reported other bugs (for instance, but not limited to, some of those explained in [2]), none of which were fixed in the next release. So, all we could do was wait.

In June 29, 2014, Mathematica version 10 was released, and we[1] quickly tried to check if the problem had been fixed. In the web page `http://www.wolfram.com/mathematica/new-in-10/` nothing is mentioned regarding the correction of errors, and we have received no additional feedback on our bug report.

The bug is still present in this new release. Actually, the short description of the previous section based on random matrices no longer shows the bug, but it still has consequences on our experiments with integer matrices as in (2). We have found examples of matrices of polynomials with integer coefficients evaluated at integers whose determinants are wrongly computed by Mathematica version 10. Again, when the same determinant is evaluated twice, different answers are quite often obtained. For the sake of brevity, we do not include these examples here, but if the reader[2] is interested, some notebooks that clearly show the bug in Mathematica 10, to which Mathematica 7 seems to be immune, can be downloaded at `http://www.unirioja.es/cu/jvarona/downloads/notebooksDetM10M7.zip`

Other examples of wrong computations

Of course, there are many more examples of wrong computations done by a computer algebra package. Many of them can be found in internet forums or distribution lists.

One typical example with Mathematica is the computation of a real integral that generates a complex result, which is clearly impossible. For instance, in Mathematica notation (and where `// N` serves to show the numerical value after computing the integral in a symbolic way), we get that

```
Integrate[Sqrt[(2t)^2 + (4 - 3t^2)^2], {t, 0, 2}] // N
```

is $0.881679 + 1.17073i$, although $(2t)^2 + (4 - 3t^2)^2 > 0$ for $0 \leq t \leq 2$.

Another example of a wrong computation of an integral is

```
Integrate[Exp[-p*t]*(Sinh[t])^3, {t, 0, Infinity}]
```

In this case, Mathematica provides the answer $6/(9 - 10p^2 + p^4)$ conditioned to $0 < \mathrm{Re}(p) < 1$ and $\mathrm{Im}(p) = 0$. This is obviously wrong because, for real p, the integral is convergent only when $p > 3$. Let us also consider the following integral (we thank one of the reviewers for this example):

```
Integrate[Integrate[Abs[Exp[2*Pi*I*x] + Exp[2*Pi*I*y]], {x, 0, 1}], {y, 0, 1}]
```

Both Mathematica and Maple return zero as the answer to this calculation. Yet this cannot be correct, because the integrand is clearly positive and nonzero in the indicated region.

Finally, let us see an example which is not an integral, but rather involves the Wigner 3-j symbols which appear in quantum mechanics. Mathematica asserts that

```
ThreeJSymbol[{r, 0}, {s+1, 0}, {s, 0}]
```

is 0, but it computes

```
ThreeJSymbol[{1, 0}, {2, 0}, {1, 0}]
```

[1]And the reviewers of the first version of this paper.

[2]Or the vendors of Mathematica.

as $\sqrt{2/15}$, which is a contradiction.

Nowadays, we cannot avoid this kind of problem, and we must be aware of them. Any mathematical study that reports computational results should dedicate some effort to explaining why the authors have faith in the results. For instance, verifying that the computation was performed in two different ways (with two different systems, both numerically and symbolically,...) and the results agreed.

At the same time, while mathematicians must be aware of the potential problems with computer algebra systems, developers should collaborate to avoid them, and this is far from being the actual situation. Many researchers experience considerable frustration in dealing with such problems. Often there is no clear way to communicate such difficulties, and if one does persist in contacting the vendor, one often receives no feedback or follow-up response. This clearly should be improved.

In addition, if a researcher with programming expertise tries to understand what is happening, another problem arises: not all mathematical software packages are open so that one can "look under the hood", and this complicates our efforts to figure out what is going on when a wrong computation appears.

Conclusions

We have been using Mathematica as a tool in our mathematical research. All our computations with Mathematica were symbolic, involving only integers (large integers, about 10 thousand digits long) and polynomials (with degree 60 at most), so no numerical rounding or instability can arise in them, and we completely trusted the results generated by Mathematica. However, we have obtained completely erroneous results. Perhaps someone may think that this was an esoteric error, without real relevance, because large integers do not appear in real life. This is not the case, because large integers are commonly used, for instance, in cryptography, where everything should work without serious errors. We have also briefly pointed out some other wrong computations that are clear to any mathematician. How then can we trust in computer algebra systems?

We know that it is very difficult to avoid errors in non-trivial programs, and a considerable effort is necessary to check them. Commercial computer algebra systems are black boxes and their algorithms are opaque to the users (and of course, also the source code), which certainly does not contribute to avoiding errors. This makes it difficult to apply modern techniques of software verification to these kinds of systems (as an example of verification in the context of an open source computer algebra system, see [5]). Moreover, lists of known bugs of computer algebra systems should be made available to the users; this is standard in free software, but an anathema for commercial packages.

Having made this criticism, let us stress that software systems have proved very useful to research mathematicians. Some well-known instances are the proof of the four-color problem by Kenneth Appel and Wolfgang Haken [1] and the Kepler conjecture by Thomas Hales [3]; less well-known is the recent success of the mathematical software Kenzo in detecting an error in a published mathematical theorem (see [6]). Software bugs should not prevent us from continuing this mutually beneficial relationship in the future. However, for the time being, when dealing with a problem whose answer cannot be easily verified without a computer, it is highly advisable to perform the computations with at least two computer algebra systems.

References

[1] K. Appel and W. Haken, The solution of the four-color-map problem, *Sci. Amer.* **237** (1977), 108–121.

[2] Ó. Ciaurri and J. L. Varona, How reliable are computer calculations? (Spanish), *Gac. R. Soc. Mat. Esp.* **9** (2006), 483–514.

[3] T. C. Hales, A proof of the Kepler conjecture, *Ann. of Math. (2)* **162** (2005), 1065–1185.

[4] S. Karlin and G. Szegő, On certain determinants whose elements are orthogonal polynomials, *J. Analyse Math.* **8** (1960/1961), 1–157.

[5] L. Lambán, J. Rubio, F. J. Martín-Mateos and J. L. Ruiz-Reina, Verifying the bridge between Simplicial Topology and Algebra: the Eilenberg-Zilber algorithm, *Log. J. IGPL* **22** (2014), 39–65.

[6] A. Romero and J. Rubio, Homotopy groups of suspended classifying spaces: an experimental approach, *Math. Comp.* **82** (2013), 2237–2244.