三位使用计算机代数系统数学家的不幸 —— 我们能信任这些系统吗？

Antonio J. Durán　Mario Pérez　Juan L. Varona

1. 引言

如今，数学家经常使用计算机代数系统辅助他们的数学研究；他们做思考，然后把冗长的计算留给计算机．每个人都 “知道” 计算机比人们能更好地执行这些工作．但是，当然，我们必须相信这些强大的计算机代数系统的计算结果．首先，让我们澄清这篇论文不是以任何方式去比较不同的计算机代数系统，而是作为一个例子，看看当数学家使用这类软件时，他们所期望水平的当前状态．尽管我们的例子是一个特定的软件系统，我们确信类似的情况可能在其它的程序也会发生．

我们现在使用 Mathematica 来发现我们导出的一些数学结果中的实例和反例，目的是找到正确的假设并最终构建数学的证明．我们的目标是改进 Karlin (卡林) 和 Szegő (塞格) 的一些结果 [4], 其涉及实线上的正交多项式．具体细节不重要；这只是一个典型数学家使用计算机代数系统的例子，让我们简单地来说明它．没有必要完全了解其中的数学，只要意识到它是一个利用计算机代数系统作为工具的典型数学研究．

我们的出发点是实线上一个离散的正测度 $\mu = \Sigma_{n\geq 0} M_n \delta_{a_n}$ (这里 δ_a 表示在 a 点的 Dirac (狄拉克) delta 函数，并且 $a_n < a_{n+1}$), 它有一个正交多项式序列 $\{P_n\}_{n\geq 0}$ (这里 P_n 是 n 次并有正的首系数)．1961 年，Karlin 和 Szegő [4] 对于 $k, n \geq 0$ 考虑 $l \times l$ Casorati (卡索拉蒂) 行列式

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

他们证明了，在 l 是偶的条件下，这些行列式对所有非负整数 n, k 都是正的．注意到多项式 P_n 的指标集合 $\{n, n+1, \ldots, n+l-1\}$ 由相继的非负整数组成．我们要扩展这个精彩

译自：Notices of the AMS, Vol. 61 (2014), No. 10, p. 1249–1252, The Misfortunes of a Trio of Mathematicians Using Computer Algebra Systems. Can We Trust in Them? Antonio J. Durán, Mario Pérez, and Juan L. Varona, figure number 1.
Antonio J. Durán 是西班牙 Universidad de Sevilla 的数学教授．他的邮箱地址是 duran@us.es.
Mario Pérez 是西班牙 Universidad de Zaragoza 的数学教授．他的邮箱地址是 mperez@unizar.es.
Juan L. Varona 是西班牙 Universidad de La Rioja 的数学和计算科学教授．他的邮箱地址是 jvarona @unirioja.es.

的结果到比连续非负整数更一般的指标集合 F. 我们有一些想要证明或者否证的猜想.

我们还没有能够证明我们的猜想，而且，正如我们至目前所了解的，这项任务似乎相当困难. 另一方面，假使我们的猜想是错误的，我们也已经尝试在我们的计算机代数系统的帮助下发现反例. 最终，我们希望这些实验也能启发解决这个问题.

我们继续构建对于离散正测度的正交多项式 (仅涉及有限多个 Dirac delta 函数，实际上，此非为对于我们猜想的限制条件). 对于多项式 P_n, 固定一个指标集合 $F = \{f_1, \ldots, f_l\}$, $f_i < f_{i+1}$, 对于大范围的 k, 为了寻找行列式的负值，我们已经计算了行列式

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

为了避免浮点算法 (舍入，截断，不稳定性) 的常见问题，我们用整数来构造所有的示例. 通过取整数作为 a_n 的值和质点 M_n 的测度值，并且使用正交多项式 P_n 的适当正规化，我们发现在 (2) 中仅涉及整数. 因此，计算应该对于计算机代数系统来说是常规的，并且应该能够完全信任结果. 我们还引入了随机参数 (当然也是整数), 以便轻松地执行许多实验.

在 Mathematica 的帮助下，我们中的一人发现了一些有关我们猜想的反例. 幸运的是，我们中的另一个人使用 Maple, 检查那些假设的反例时，发现他们根本不是反例. 从头修改我们的算法后，我们得出结论：或者是使用 Mathematica 执行的计算，或者是使用 Maple 执行的计算必定有错误. 事情开始变得清楚，使用 Mathematica 的同事也发现了一些关于上述 Karlin 和 Szegő 的关于情况 (1) 的结果的 “反例”, 甚至更显著的是，他的算法在给定相同的输入情况下产生了不同的输出结果. 我们的结论是 Mathematica 的计算不正确. 然而，我们的数学问题 (和我们的算法) 太复杂，以至于不能说服任何人相信当 Mathematica 在使用整数计算时会犯错误.

2. 隔离错误

为试图隔离计算问题, 我们最终认识到, 在某些情况下, Mathematica (当时版本 9.0.1) 在计算大整数项的行列式时出现一些奇怪的错误. 错误不总是发生, 只在某些情况下. 更糟糕的是, 给定相同的矩阵, 行列式函数可以给出不同的值 ! 这类似于 Thomas Nicely 在 1994 年发现的众所周知的奔腾分区错误，只有一些数据会产生影响. 但是似乎 Mathematica 是一个黑盒子，甚至比微处理器的内部更黑，所以很难设法了解什么样的数字是由我们描述的 Mathematica 错误所影响.

相反，我们设计了一种方法，可轻易地生成具有大整数项的矩阵，其行列式用 Mathematica 计算，则明显错了，这个方法可以不参考引出它的数学问题就可以描述. 由于错误并不总是出现，我们开发了一个随机生成这些矩阵的过程. 首先，我们生成一个随机的 14×14 矩阵，矩阵各项的取值是 -99 到 99 之间的整数：

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

basicMatrix =

-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

,

smallMatrix =

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

图 1　basicMatrix 和 smallMatrix 的矩阵例子

为了得到更大的整数，我们把每列乘以 10 的某个次方. 这等价于乘以一个对角矩阵；举个例子，我们取

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}].

为避免只是得到一些以大量 0 结尾的整数，我们加上一个小的随机矩阵 smallMatrix = Table[Table[RandomInteger [{-999, 999}], {i, 1, 14}], {j, 1, 14}]. 那么我们取 bigMatrix = basicMatrix.powersMatrix + smallMatrix (在 Mathematica 的记号里，点 "." 被用来表示矩阵乘法). 现在我们计算行列式两次：

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

奇怪的是，我们经常发现 a 和 b 不相等！很容易发现：当检验是否 a==b 时，经常给出否的答案，或者直接观察比较它们的近似数据 N[a] 和 N[b].

让我们看看一个实际操作这些过程的例子：利用出现在图 1 中的矩阵，我们得到 N[a]=$-3.263388173990166\cdot 10^{9768}$ 和 N[b]=$-8.158470434975415\cdot 10^{9768}$, 并且多次执行这些同样的过程，得到与此不同的其它值. 这些值没有一个是正确的，因为这个 bigMatrix 的行列式大概是 $1.95124219131987\cdot 10^{9762}$.

我们发现的这些错误现象发生在 Mathematica 版本 8 (2010 年 11 月 15 日发布) 到版本 9.0.1 (这是上面提到实验完成时的最新版本，也是本文提交时的第一版本), 在 Mac 和 Windows 操作系统下都是如此. 对版本 6 和 7 很可能没有这个问题，至少对于上述范围的数.

在 2013 年 10 月 7 日, 我们报告了这个缺陷 (编号：303438), 得到了 Wolfram Research Inc. 的一个客气答复：

"你们提到的行列式运算确实存在严重的错误，我们已经把你们提供的信息提交给我们的开发人员.

我们一直致力提高 Mathematica, 我们感谢你们提供的问题. 如果你们遇到任何其他操作问题，或其它要求，请联系我们，不要犹豫. "

到 2014 年 6 月，什么都没有改变. 当我们其中一人又报告了其它的缺陷 (例如，但不限于，在 [2] 中说明的一些问题), 我们又收到了以前类似的回复，在下一个版本发布时，没有一个问题得到解决. 所以，我们所能做的就是等待.

2014 年 6 月 29 日, Mathematica 版本 10 发布，我们立刻尝试检验问题有没有修复. 在网页 `http://www.wolfram.com/mathematica/newin-10/`, 没有任何关于更正错误的布告，我们也没有收到关于我们所提缺陷的其它的反馈.

在新发布的版本里，缺陷仍然存在. 实际上，前节基于随机矩阵的简短描述的缺陷不再出现，但是它仍然有 (2) 中整数矩阵实验的后果. 我们已经找到整系数多项式矩阵的例子，如果整数取值，它的行列式以 Mathematica 版本 10 计算，仍然会出现错误. 再次，当同样行列式被估值两次，仍然经常获得不同的答案. 为了简洁起见，我们不给出这些例子，如果读者有兴趣，有些给出 Mathematica 10 错误的笔记，可以在 `http://www.unirioja.es/cu/jvarona/downloads/notebooksDetM10M7.zip` 下载，对 Mathematica 7 似乎可避免这些缺陷.

3. 其它计算错误的例子

当然，有更多的使用计算机代数软件包计算错误的例子. 其中许多可以在网络论坛或分发列表中找到.

一个典型的 Mathematica 例子是实积分计算得到复数结果，这显然不可能. 比如，用 Mathematica 记号 (`//N` 表示用符号方式计算积分后的数值), 我们有

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

是 $0.881679 + 1.17073i$, 而 $(2t)^2 + (4 - 3t^2)^2 > 0, 0 \leq t \leq 2$.

另一个积分计算错误的例子:`Integrate[Exp[-p*t]*(Sinh[t])^3, {t,0,Infinity}]`. 在这个例子中，Mathematica 给出的结果是 $6/(9 - 10p^2 + p^4)$, 这里条件是 $0 < \mathrm{Re}\,(p) < 1, \mathrm{Im}\,(p) = 0$. 这是明显错误的，因为对实数 p, 这个积分只有在 $p > 3$ 时才收敛. 让我们考虑下面的积分 (我们要感谢这个例子的一位评论人):

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

Mathematica 和 Maple 都给出了零作为计算结果. 但是这不可能正确，因为被积分函数显然在积分区域是正的且不为零.

最后，我们再看一个不是积分的例子，它是关于 Wigner3-j 记号，这是出现在量子力学中的. Mathematica 断言：

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

是 0, 但是它计算

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

得到 $\sqrt{2/15}$, 这是互相矛盾的.

现在我们不能避免这种问题，我们必须意识到这种问题. 任何报告计算结果的数学研究应该致力一些努力来解释为什么作者对结果有信心. 例如，用两种不同的方法进行计算 (两个不同的系统，数值方式和符号方式，...) 结果是一致的.

同时，数学家必须注意计算机代数系统潜在的问题，开发人员应该与数学家合作来避免它们，但这不是现在真正的状况，许多研究人员在处理这些问题上经历了相当的挫折．通常没有明确的方式沟通这种困难，如果有人坚持接触供应商，经常收不到反馈或后续响应．这显然应该改进．

另外，如果一个具有编程专业知识的研究员试图理解发生了什么，另一个问题出现了：不是所有的数学软件包是开放的，人们可以 “开盖查看”，这使人们在错误计算发生时，要弄清到底发生了什么的努力复杂化了．

4．结论

我们一直在我们的数学研究中使用 Mathematica 作为工具．我们所有利用 Mathematica 的计算已经是符号性的，只涉及整数 (大整数，约 10 000 位长) 和多项式 (最多为 60 次), 因此在计算中不会出现数值舍入或不稳定性，我们完全信任由 Mathematica 生成的结果．但我们已经得到了完全错误的结果，或许有人可能认为这是一个难以理解的错误，没有真实的意义，因为大整数不出现在现实生活中．事实并非如此，因为大整数常被使用，例如，在密码学，其每件事情均应在没有严重错误下操作．我们也简短地指出了一些其他错误的计算，这对任何数学家都是明显的．我们怎么能相信计算机代数系统？

我们知道，在复杂的程序中避免错误是非常难的，必须以相当大的努力来检验它们．商业计算机代数系统是黑盒子，他们的算法对用户是不透明的 (当然，源代码也是), 这当然是无助于避免错误．这使使用现代软件验证技术来处理这些类型的系统是困难的 (对于一个在开放源的代数系统上的验证例子，见 [5])．此外，计算机代数系统的已知错误的列表应向用户提供；这对免费软件是标准，但对商业版却是不行的．

尽管做了很多批评了，我们要说明，软件系统已被证明对研究型数学家非常有用．一些著名的实例有，Kenneth Appel (阿佩尔) 和 Wolfgang Haken (黑肯) 证明的四色问题 [1], Thomas Hales 给出的 Kepler (开普勒) 猜想证明 [3]; 不太知名的有，最近数学软件 Kenzo 对已发表的数学定理检测错误的成功 [6]．软件的缺陷不应该阻止我们继续在未来从软件的使用中受益．但是，当处理一个问题，它的答案没有计算机就难以验证时，那么暂且最好能用至少两个计算机代数系统来完成计算．

参考文献

[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.

(张 伟 郭磊磊 译　　王世坤 铁梦玲 校)