متى لا يجب الوثوق في الحواسيب؟

عبد العزيز شوتري
قسم الرياضيات، المدرسة العليا للأساتذة، القبة
abdelaziz.choutri@g.ens-kouba.dz

1. مقدمة

لقد أصبحت الحواسيب جزءا من حياتنا اليومية، وأصبح حتى تلاميذ الابتدائي لا يقومون بأي عملية حسابية، ولو كانت بسيطة، دون الاستعانة بالآلات الحاسبة أو بحاسبات الهواتف ويثقون في نتائجها ثقة عمياء. ولم يعد يقتصر الأمر على تلاميذ الابتدائي بل تعدّاه إلى طلبة الثانوي والجامعي، والكثير من الباحثين في شتى الميادين العلمية.

نودّ في هذا المقام التنبيه إلى أن الحواسيب ككل الآلات الأخرى، يمكن أن تنتج عن استعمالها الخاطئ أخطاء حسابية لا نتفطن إليها، مع أن لها تأثيرا كبيرا على النتائج المرجوّة.

سنسرد في هذا المقال عدة أنواع من الأخطاء الناتجة عن البرمجة التي لا تنبهنا إليها الحواسيب. وقد قسّمنا هذه الأخطاء إلى عدة مجموعات، ورتبناها من الأخطاء البسيطة إلى الأقل بساطة، إلى الأكثر تعقيدا. وهذه الأخيرة متعلقة بالنمذجة الرياضية للمسائل الفيزيائية والمسائل الطبيعية عموما التي تتدخل الحواسيب في نتائجها. وكثيرا ما نجد نتائج على شكل جداول إحصائية ورسومات بيانية مزهوٌ بها أصحابها، وأحيانا يتفاخرون بها ويقدّمونها بشكل يسرُّ الناظرين إليها. مع أنها، للأسف، خاطئة وبعيدة كل البعد عن النتائج الفعلية.

يعتقد بعض المبرمجين وخاصة المبتدئين منهم، نظرا لانبهارهم بالسرعة الفائقة للحواسيب، أنهم استطاعوا استعمال الحواسيب لحل مسائل رياضية أو في علوم الطبيعة عموما. ونتيجة ثقتهم العمياء في الحواسيب يرتكبون أخطاء في البرمجة لا تنبّه لها الحواسيب، ويحصلون على نتائج يبنون عليها تصوراتهم للحلول، ويطلقون الأحكام المستقبلية على مسائل مشابهة لم يقوموا بحلها بحجة التعميم. وبذلك يكونون قد ارتكبوا أخطاء عواقبها أحيانا وخيمة بدون علم، وبالتالي تكون تنبؤاتهم بعيدة عن الحقيقة لدرجة التضاد.

ومع الأسف، تعدّت هذه الأخطاء إلى المحترفين فأصبح البعض منهم يتعمّدون بعض الأخطاء، وأحيانا يعدّلون في النتائج قصد تقديمها بالشكل الذي يتمنّون أن تكون وذلك بحثا عن السبق العلمي والشهرة، وأحيانا بحثا عن الثراء.

نكتفي هنا مثال عن هذا النوع من الأخطاء المتعمّدة، والتي استُعمل فيها الحاسوب، وربما سنخصص لها مقالا في الأعداد القادمة من المجلة.

النص التالي مأخوذ مترجما من صحيفة المركز الوطني للبحث العلمي الفرنسي (CNRS)، وقد جاء تحت عنوان: "سبع قضايا شهيرة لعلماء متهمين بالاحتيال" في الموقع

https://lejournal.cnrs.fr/articles/sept-cas-celebres-de-scientifiques-accuses-de-fraude

النص:

في عام 1998، نشر الجراح البريطاني أندرو ويكفيلد في دراسة على اثني عشر طفلاً، زعم فيها أن بعضهم قد طوّر شكلاً من أشكال التوحّد بعد إعطاء لقاح الحصبة والحصبة الألمانية (MMR). أدّى هذا المنشور إلى انخفاض حاد في تغطية التطعيم في المملكة المتحدة، وزيادة ملحوظة في حالات الحصبة. ومع ذلك، لم يتمكن أي فريق آخر من

تكرار هذه النتيجة. والأسوأ من ذلك أن صحيفة صنداي تايمز كشفت في عام 2004 أن الأطفال الذين يُزعم أنهم أصيبوا بالتوحّد كانوا مصابين بالتوحّد قبل التطعيم.

قبل كل شيء، كشفت الصحيفة أن ويكفيلد تمّت رشوته من قِبل محام أراد اتخاذ إجراء قانوني ضد المختبر الذي ينتج اللقاح. كانت نتائج الدراسة في الواقع ملفّقة بالكامل لبلوغ هذا هذا الغرض. وقد سلّطت العديد من المقالات في المجلة الطبية البريطانية الضوء في وقت لاحق، موضحة أن ويكفيلد خطط لبدء عمل تجاري مدعوم بحملة دعائية ضد اللقاحات.

وفي يناير 2010، قضت محكمة المجلس الطبي العام البريطاني بأن الجراح مذنب وأنه قام بتلفيق البيانات، وجرّدته من حقّه في ممارسة الطب في المملكة المتحدة مدى الحياة. ومع ذلك، فإن هذا الرأي حول لقاح الحصبة والنكاف والحصبة الألمانية، الذي لا أساس له من الصحة، لم يتبدّد تمامًا لدى الجمهور. ويكفيلد، الذي أنكر دائمًا أي احتيال، ظل يمارس عمله الآن في الولايات المتحدة ويتدخل بانتظام لصالح جماعات الضغط المناهضة للتلقيح، وبذلك فهو يواصل مسيرته المهنية كـ"تاجر للشك". انتهى النص المقتبس.

نعود الآن إلى الأخطاء غير المتعمّدة.

مثال1

إذا كان a=1/2 فإن قيمة a عند الحاسوب تساوي 0 لأن القسمة هنا تتم في مجموعة الأعداد الصحيحة)، فإذا أردناها في مجموعة الإعداد الحقيقية يجب كتابة a=1./2 أو a=1/2. أو a=1./2.، ففي هذه الحالات تكون النتيجة حقيقية أي a=0.5. مع ملاحظة أننا إذا استعملنا برامج جاهزة للحساب كبرامج Scilab أو MatLab فهاته البرامج هي التي تتكفل بإجراء القسمة الحقيقية (أي في مجموعة الإعداد الحقيقية) إلا إذا تعمّدنا القسمة الصحيحة (أي في مجموعة الأعداد الصحيحة). ففي هذه الحالة يجب إخبار هاته البرامج.

2. الأخطاء الناتجة عن استعمال الأعداد الصحيحة الكبيرة

مثال 2

لنقم بالعملية الحسابية التالية باستعمال آلة حاسبة، أو حاسوب، أو باستعمال حاسبة الهاتف.

11111111x11111111

سنجد عدة نتائج مختلفة منها:

$$10000001x10000001=1.x10^{14}=100000000000000$$

أو

10000001x10000001=100000020000001.

من الواضح أن الفرق كبير جدا!

كما يمكن أن نجري العملية التالية أيضا 111111111x111111111.

سنجد

$$111111111x111111111=1.234567898765x10^{16}=12345678987650000$$

أو

12345678987654321.

السؤال: ما هي النتيجة الصحيحة؟

من الواضح أن النتيجة الأولى في كلتا الحالتين خاطئة لأن ضرب عددين آخر أرقامهما 1 لا يمكن أن ينتهي بـ 0.

ماذا حدث؟ وأين الخلل؟

إن قدرة أي آلة (آلة حاسبة أو حاسوب) على تخزين الأعداد محدودة مهما كان تطور الحاسوب، إذ أنه لا يمكن تخزين الأعداد الأكبر من 2^n حيث n=16 أو n=32 أو n=64.

يقوم الحاسوب بتحويل العدد من النظام العشري إلى النظام الثنائي، ثم يخزنه على الشكل التالي

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

وهذا هو أكبر عدد في النظام الثنائي، ويساوي 2^{16-1} -1 في النظام العشري في حالة n=16.

مثال 3

نجري العملية التالية بواسطة برنامج بلغة C أو Fortran في حاسوب قدرته التخزينية n=32:

12 !x13.

سيعطي النتيجة التالية

12 !x13=1932053504.

من الواضح أن النتيجة خاطئة !!

ما هو تفسير ما قام به الحاسوب؟

لقد حسب ! 12 فوجدها 479001600 =! 12، وهذا صحيح، ثم يقوم بتخزينها على الشكل

11100100011001111110000000000

0	0	0	1	1	1	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

وعند ضرب هذا العدد في 13 يحصل على النتيجة التالية

10111001100101000110011000000000.

ومن ثمّ يخزنها، وذلك بملء الجدول من اليمين إلى اليسار فيحصل على

1	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0

يُهمل الرقم 1 في الأخير، ولا يخزّنه لأنه في الموضع 33 وقدرة الآلة هي 32 فقط. فعندما نطلب من الآلة كتابة النتيجة تخرجها في النظام العشري على الشكل

13 !=1932053504.

وبطبيعة الحال، فهي نتيجة خاطئة.

3. الأخطاء في الأعداد الحقيقية (جزء صحيح وجزء عشري)

أما الأعداد الحقيقية فتُخزّن بقيمة تقريبية، حيث أنه لا يراعي إلا عددا محددا من الأرقام بعد الفاصلة، فيأخذ بعين الاعتبار 6 أرقام إذا طلبنا منه دقة بسيطة أو 10 أرقام إذا كانت الدقة مضاعفة، وتُهمل الأرقام الأخرى. وهذا ناتج عن قدرة كل آلة على تخزين الأعداد، ومهما كان الحاسوب فقدرته محدودة.

لنأخذ مثالا بسيطا، وليكن جمع 0.1 ثلاث مرات.

```c
#include <stdio.h>
 main()
{

   int i,k,j,n;
 for(j=1;j<=3;j++)
 {
  printf("n=");
  scanf("%d",&n);
  k=1;
   for(i=1;i<=n;i++)
   k=k*i;

  printf("%d!=%d\n",n,k);
}
}
```

```
    2 |  main()
      |  ^~~~
n=5
5!=120
n=12
12!=479001600
n=13
13!=1932053504
```

الشكل 1. برنامج يحسب عاملي عدد طبيعي

```c
#include <stdio.h>
 main()
{
   float x;
   int i;
   x=0;
   for(i=1;i<=3;i++)
 {  x=x+0.1;

 } printf("x=%10.8f\n",x);
}
```

```
    2 |  main()
      |  ^~~~
x=0.30000001
```

الشكل 2. برنامج يقوم بجمع 0.1 ثلاث مرات X=0.1+0.1+0.1

نلاحظ أن مجموع 0.1 ثلاث مرات من المفروض أن يعطي 0.3 بالضبط، لكن النتيجة المتحصّل عليها بالحاسوب هي 0.30000001.

فمن أين نتج هذا الخطأ؟

تصوّروا لو أننا في مسألة اقتصادية نريد جمع 0.1 مليون مرة، فالنتيجة ستكون

x=100958.34375000

بدلا عن 100000. وبالتالي فالخطأ كبير جدا وهو 958.34375000، وهو ناتج عن كون العدد 0.1 أصم في النظام الثنائي أي

$(0.1)_{10}=(0.0001100110011.....)_2$.

فعدد الأرقام بعد الفاصلة غير منته. وبالتالي تأخذ الآلة بعين الاعتبار ستة أرقام فقط بعد الفاصلة وتُهمل الباقي.

4. القسمة على عدد حقيقي صغير جدا

إذا أردنا أن نستعمل الحاسوب لإيجاد $\frac{1}{10^{-11}}$ فيكون البرنامج التالي كما هو موضح في الصورة.

```
1  #include <stdio.h>
2  #include <math.h>
3   main()
4  {
5  float x,y;
6  x=pow(10,-11);//x=10^(-10)
7  y=1./x;
8  printf("x=%f\n y=%f\n",x,y);
9  }
```

```
   11 |  main()
      |  ^~~~
x=0.000000
 y=99999997952.000000
```

الشكل 3. حساب مقلوب عدد صغير جدا بدقة عادية

نلاحظ أن قيمة x في السطر 6 من البرنامج هي 10^{-11}، وقيمة y في السطر 7 هي مقلوب x. لكن النتيجة مختلفة تماما عن الواقع حيث من المفروض أن يكون x=0.00000000001 و y=100000000. وهذا ناتج عن كون الحاسوب لا يراعي إلا 10 أرقام بعد الفاصلة ويَعتبر باقي الأرقام غير معنوية بالعملية، أي أن هناك أرقاما أخرى يأخذها الحاسوب ولا نراها، ولا يمكن للمستعمل أن يتحكم فيها. ولذا تخرج النتيجة على الشكل

y=99999997952.000000 .

إذا استبدلنا السطر 5 بالتصريح double x,y نحصل على نتيجة صحيحة. وذلك لأننا طلبنا حجز مكان مضاعف للأعداد الصحيحة فيكون عدد الأرقام المعنوية بعد الفاصلة هو 15 بدلا من 10 كما هو مبين في الصورة.

```
1  #include <stdio.h>
2  #include <math.h>
3   main()
4  {
5  double x,y;
6  x=pow(10,-11);//x=10^(-10)
7  y=1./x;
8  printf("x=%15.13lf\n y=%lf\n",x,y);
9  }
```

```
   3 |  main()
     |  ^~~~
x=0.0000000000100
 y=100000000000.000000
```

الشكل 4. حساب مقلوب عدد صغير جدا بدقة مضاعفة

لكن إذا أخذنا $x<10^{-15}$ فإننا نحصل على نتائج خاطئة.

5. ما العمل إذا كنا في حاجة إلى هذه العمليات؟

توجد طريقة رياضية تسمى الحساب الشكلي Formal calculus. والفكرة الأساسية في هذه الطريقة هي أن نتعامل مع الأرقام المكوّنة للعدد وليس العدد ذاته. فيكتب العدد 2045 مثلا، على الشكل الشعاعي (2,0,4,5)، وبدلا عن تخزين العدد كما هو، نخزّن أرقامه فقط. وفي هذه الطريقة نُعرّف من جديد العمليات الأساسية الأربع. فالجمع مثلا نعرّفه كالتالي:

$$2045+83=(2,0,4,5)+(0,0,8,3)$$
$$=(2,1,2,8)$$
$$=2128\ .$$

وتعرّف عمليات الطرح والضرب والقسمة على الأرقام.

6. أخطاء ناتجة عن النماذج الرياضية

هناك الكثير من المسائل الفيزيائية التي توضع لها نماذج رياضية وتدرس وتحلّل رياضيا، ونثبت أن تلك المسائل تتمتع بحل. وفي كثير من الأحيان يكون الحل وحيدا تحت شروط محددة، وعندما ننتقل إلى حساب الحل عدديا تكون المسائل العددية على شكل جملة معادلات خطية.

نلاحظ هنا أنه إذا لم نكن نعرف خصائص مصفوفة الجملة فقد نحصل على حل ليس له أي علاقة بالحل الفعلي للمسألة.

لنأخذ المثال التالي: لتكن الجملة التالية

$$\begin{bmatrix}7 & 1 & 11 & 10\\ 2 & 6 & 5 & 2\\ 8 & 11 & 3 & 8\\ 6 & 9 & 3 & 6\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix}=\begin{bmatrix}29\\ 15\\ 30\\ 24\end{bmatrix}.$$

إن حل هاته الجملة هو

$$x=\begin{bmatrix}1\\ 1\\ 1\\ 1\end{bmatrix}$$

لنستبدل شعاع الطرف الثاني بالشعاع التالي

$$\begin{bmatrix}29\\ 15\\ 30\\ 23.9\end{bmatrix}$$

(لقد استبدلنا المركبة الأخيرة بـ 23.9 بدلا من 24، وهو تغيير طفيف كثيرا ما يحدث كخطأ تقريب.). عندئذ نحصل على حل بعيد كل البعد عن الحل الأول، وهو

$$x=\begin{bmatrix}4.4555556\\ 0.4333333\\ 1.4333333\\ -1.8388889\end{bmatrix}.$$

وهذا ناتج عن كون المصفوفة غير مكيّفة جيدا، أي أن عددها التكيفي matrix condition number بعيد جدا عن الواحد. حيث أن عدد تكيّف مصفوفة معرفة موجبة تماما هو

$$cond(A) = \frac{|max\ (\lambda_i(A\ ))|}{|min\ (\lambda_i(A\ ))|}$$

حيث تمثّل $\{\lambda_i(A\ )\}$مجموعة القيم الذاتية للمصفوفة A.

فإذا كان هذا العدد بعيدا عن العدد 1 تكون المصفوفة سيئة التكييف poorly conditioned.

ولمعالجة هذا النوع من المسائل نستعمل طريقة تفكيك المصفوفة حسب عناصرها الشاذة singular value .

فإذا كانت لدينا الجملة $Ax = b$ فإن

$$A = VSU^T$$

حيث U و V مصفوفتان متعامدتان و S مصفوفة قطرية يحتوي قطرها على العناصر الشاذة للمصفوفة A.

ثم نُكوّن الأشعة الملحقة y و z كما يلي

$$Uz = b$$
$$Sy = z$$
$$V^T x = y$$

وبما أن المصفوفتين U و V متعامدتان فإن حساب z و x لا يطرح أي مشكلة، حيث

$$z = U^T b$$
$$x = Vb$$

7. خلاصة

إذا أردنا استعمال حاسوب فيجب التعرّف على خصائصه وقدراته على التخزين ومعرفة إن كان نظام تشغيله من نوع 32bits أو 64bits. ويجب كذلك ألّا نضع الثقة المطلقة في الحواسيب، حيث يتعين إيجاد طريقة لمراقبة النتائج كاستعمال البرنامج لحل مسألة نعرف حلها مسبقا. فإذا أعطانا حلا غير الحل المنتظر نستنتج أن طريقة برمجتنا يشوبها خلل يجب إصلاحه. أمّا بالنسبة للنماذج الرياضية، فهناك طرق تحليلية لمعرفة فعالية النماذج ومواكبتها للشروط الفيزيائية للمسائل.