Un ejemplo de demostración con ayuda del ordenador

Presentación

Esta práctica es un regalo que nos hace el Profesor **Armando G.M. Neves**, de la Universidad Federal de Minas Gerais en Brasil. El Profesor Neves es físico y matemático, y experto en programación con *Mathematica*. No dejes de visitar su página *http://www.mat.ufmg.br/~aneves* donde, entre otras cosas, encontrarás, en formato de *notebooks*, un curso bastante completo de introducción a *Mathematica*. Aunque el contenido de la práctica no está directamente relacionado con el programa propio de la asignatura de Cálculo, considero que esta excursión está justificada por su belleza y su interés. Confío en que cuando llegues al final estarás de acuerdo conmigo.
Ni que decir tiene que yo tan sólo me he limitado a traducir casi literalmente la práctica de su versión portuguesa, evitando, eso sí, algunos tecnicismos. Naturalmente, aprovecho la ocasión para explicar algunas funciones útiles de *Mathematica* que posiblemente aún no conoces.

Introducción

El ordenador puede usarse de distintas formas para obtener resultados matemáticos. Posiblemente, la forma más usual consiste en utilizarlo para realizar largos cálculos numéricos, a partir de cuyos resultados se pueden formular conjeturas que posteriormente habrá que demostrar. Otra posibilidad consiste en probar matemáticamente, a la manera tradicional, que un problema que implica un número infinito de casos puede reducirse a un número finito, *n*, de casos. Cuando *n* es pequeño, se acostumbra a usar las técnicas tradicionales para comprobar el resultado en cada uno de los casos. Cuando *n* es un número grande, se puede usar el ordenador para hacer esta tarea, **suponiendo**, claro está, que *el resultado de los cálculos hechos por el ordenador es exacto*. Cuando el problema implica solamente operaciones con números enteros, se acepta que la hipótesis de la última frase es verdadera y que el teorema está demostrado si el ordenador lo verifica en cada uno de los *n* casos.
Vamos a ver en esta práctica un ejemplo de un problema con infinitos casos que puede ser reducido a un número finito, *n*, de casos (donde *n* será 243), cada uno de ellos podrá ser verificado por el ordenador mediante operaciones con números enteros solamente.

El problema

Este problema (y la idea para su solución) está sacado del libro The Beginner's Guide to *Mathematica* Versión 3, de Jerry Glynn y Theodore Gray, y, según los autores, fue propuesto y difundido por un cierto profesor Nichols. El problema es el siguiente.
Tomemos un número natural x_0 y definamos x_1 como la suma de los cuadrados de los dígitos decimales de x_0, x_2 como la suma de los cuadrados de los dígitos decimales de x_1, etc. Definamos una función **sumcuad** que realice la operación de sumar los cuadrados de los dígitos decimales de un número natural. *Mathematica* dispone de una función `IntegerDigits[n]` que proporciona una *lista* con los dígitos decimales del número `n`. Bastará, por tanto, sumar los números de la lista `IntegerDigits[n]^2`. Para sumar los elementos de una lista cuya *longitud* desconocemos, lo mejor es usar la función **Plus** aplicada a los elementos de dicha lista. Escribe `Plus[a,b,c,d]` y `FullForm[{a,b,c,d}]`. Como puedes ver, para *Mathematica*, una *lista es una función*, **list**, que se aplica a los elementos que la forman. Por ello, para aplicar `Plus` a la *lista* `IntegerDigits[n]^2` todo lo que hay que hacer es sustituir en la *expresión* `IntegerDigits[n]^2` su *cabecera*, `List`, por `Plus`. Esto se consigue con el comando `Apply[f,expr]` que sustituye la cabecera de `expr` por `f`. Definimos, por tanto:

```
In[1]:=
 sumcuad[n_]:=Apply[Plus,IntegerDigits[n]^2]
```

Se trata ahora de *iterar* esta función para que a partir de un valor inicial x_0, calcule, por ejemplo, $x_1, x_2, \ldots, x_{15}$. Para iterar una función `f` disponemos del comando `Nest[f,expr,k]` que da como salida `f` aplicada `k` veces a `expr`. Cuando, además, queremos obtener la *lista* de los valores de `Nest[f,expr,k]` desde `k=0` a `k=n`, usamos `NestList[f,expr,n]`. Escribe `Nest[f,x,4]`, `NestList[f,x,4]` y `NestList[sumcuad,27,15]` para entender bien esto. Vamos a obtener ahora una matriz cuyas filas van a ser las listas `NestList[sumcuad,k,15]` desde `k=1` hasta `k=30`.

```
In[2]:=
 MatrixForm[Table[NestList[sumcuad,k,15],{k,30}]]
```

Si observas con detenimiento la matriz obtenida, verás que sólo hay dos posibilidades para la sucesión de valores x_n; o esta sucesión alcanza en algún lugar el valor 1, que después se repetirá indefinidamente, o bien, después de algunas iteraciones iniciales, la sucesión acaba por caer en el ciclo de valores $145, 42, 20, 4, 16, 37, 58, 89$, el cual se repetirá indefinidamente en este orden. En el primer caso diremos que la sucesión tiene un *punto fijo* (el 1), en el segundo caso diremos que la sucesión tiene un *ciclo de período* 8.

Sorprendido con este resultado, el profesor Nichols decidió verificar si esto seguía siendo así para valores mayores que `k=30`. Probemos ahora con `k=100`, pero en vez de mostrar todos los resultados, definamos una función que compruebe si los comportamientos posibles son solamente los dos que han aparecido hasta ahora. Para ello, basta comprobar si entre los números $\{x_0, x_1, \ldots, x_k\}$ están el 1 o el 145 (o cualquier otro número del ciclo de período 8 ya identificado). Definimos, por tanto, una función cuya variable es una *lista* y que da como salida un texto apropiado según que en dicha lista esté el 1 o el 145 o ninguno de ellos. El comando `MemberQ[list,expr]` devuelve `True` si un elemento de la lista, `list`, coincide con `expr` y `False` en otro caso.

```
In[3]:=
 test[x_List]:= "punto fijo/; MemberQ[x,1];
 test[x_List]:= "período 8/; MemberQ[x,145];
 test[x_List]:= "ni lo uno ni lo otro"
```

Queremos aplicar `test` a todas las listas de la lista `Table[NestList[sumcuad,k,15],{k,100}]`. Disponemos para ello del comando `Map[f,expr]` (también escrito `f /@ expr`) que aplica `f` a cada elemento de *primer nivel* en `expr`. Escribe para entenderlo `Map[f,{a,b,{c,d},u+v}]`. Fíjate cómo funciona.

```
In[4]:=
 datos30=Map[test,Table[NestList[sumcuad,k,15],{k,30}]]
```

Si pedimos `datos100` la lista sería muy larga. El comando `Cases[list,expr]` proporciona una lista (que puede ser vacía - `{}`) con los elementos de `list` que coinciden con `expr`. Por su parte, el comando `Count[list,expr]` da el número de elementos en la lista `list` que coinciden con `expr`.

```
In[5]:=
 datos100=Map[test,Table[NestList[sumcuad,k,15],{k,100}]];


In[6]:=
 Cases[datos100,"ni lo uno ni lo otro"]


In[7]:=
 Count[datos100,"punto fijo"]
```

Está claro, ¿verdad? Parece que un 20% de los casos acaban en punto fijo y los demás en el ciclo de período 8. Hagamos una prueba aún más exigente. Generaremos una lista de 100 números aleatorios de mil dígitos cada uno y comprobaremos si el comportamiento es el mismo. ¡No olvides escribir ";" si no quieres que la pantalla se llene con números enormes!

```
In[8]:=
 aleat=Table[Random[Integer,{10^999,10^1000}],{100}];


In[9]:=
 datosaleat= Map[test,Table[NestList[sumcuad,aleat[[k]],15],
 {k,100}]];


In[10]:=
 Cases[datosaleat,"ni lo uno ni lo otro"]


In[11]:=
 Count[datosaleat,"punto fijo"]
```

Reflexiona: hemos generado 100 números aleatorios de 1000 dígitos cada uno; hemos aplicado a cada uno de ellos la función **sumcuad** 15 veces y, de nuevo, solamente aparecen los dos comportamientos ya conocidos. Es llamativo, además, que incluso partiendo de valores iniciales enormes, basten tan sólo 15 iteraciones para detectar uno de los dos comportamientos. Podemos enunciar ahora un teorema que trataremos de demostrar.

Teorema

Sea $\texttt{sumcuad} : \mathbb{N} \to \mathbb{N}$ la función antes definida y, para cada $x_0 \in \mathbb{N}$ sea $\{x_n\}$ la sucesión dada por $x_1 = \texttt{sumcuad}[x_0]$, $x_{n+1} = \texttt{sumcuad}[x_n]$. Entonces para todo $x_0 \in \mathbb{N}$ existe $n_0 \in \mathbb{N}$ tal que para todo $n > n_0$ se verifica una, y sólo una, de las siguientes alternativas a) y b):

a) $x_n = 1$,

b) $x_n \in \{145, 42, 20, 4, 16, 37, 58, 89\}$.

La parte tradicional de la demostración

Es claro que de entre todos los números de 3 dígitos el valor máximo se **sumcuad** se alcanzará en 999. Igualmente, de entre todos los números de 4 dígitos el valor máximo se alcanzará en 9999, etc. Echemos un vistazo a estos valores máximos.

```
In[12]:=
 sumcuad[999]


In[13]:=
 sumcuad[9999]
```

Por tanto, si queremos verificar la validez del teorema para valores de $x_0 \leqq 999$, será suficiente verificarlo para todos los casos hasta $x_0 = 243$. Ahora, si $x_0 \leqq 9999$, entonces $x_1 \leqq 324$ y, por tanto, $x_2 \leqq 243$, por lo que después de dos iteraciones empezando con cualquier número de 4 dígitos obtenemos un número comprendido entre 1 y 243. Dejemos de lado por el momento la verificación de si hasta 243 los únicos casos son los dos ya conocidos. El hecho es que queremos verificar el teorema no hasta 9999, sino para todo $n \in \mathbb{N}$. Podemos razonar como sigue. Definamos

```
In[14]:=
 f[n_]:=Log[10, (10^n -1)/(81*n)]
```

Es decir, `f` es el logaritmo decimal de la razón entre el mayor número con `n` dígitos decimales, $10^n - 1$, y el valor de **sumcuad** en dicho número, `sumcuad[`10^n-1`]`$=81n$. Por tanto, $f(n) > 1$ implica que $\dfrac{10^n - 1}{\texttt{sumcuad}[10^n - 1]} > 10$, esto es, $\texttt{sumcuad}[10^n - 1] < 10^{n-1} - \dfrac{1}{10}$, de donde se sigue que el número natural $\texttt{sumcuad}[10^n - 1]$ tiene como máximo $n-1$ dígitos y, como para $k \leqslant 10^n - 1$ se tiene que `sumcuad[`k`]`$\leqq$`sumcuad[`10^n-1`]`, lo mismo será cierto para `sumcuad[k]` con $k \leqslant 10^n - 1$.

Para calcular algunos valores de `f` escribe `Table[N[f[n]],{n,10}]`. El primer valor de n para el que `f[n]`>1 es $n = 4$, lo que nos dice que `sumcuad[9999]` tiene como máximo 3 dígitos. Observa que `f` parece ser una función creciente en $[1, +\infty[$. Puedes hacerte una idea de cómo es `f` dibujando su gráfica en el intervalo $[1,5]$, `Plot[{1,f[x]},{x,1,5}]`. Podemos probar fácilmente que `f` es creciente estudiando el signo de su derivada lo que puede hacerse fácilmente "a mano", pero usaremos el ordenador por comodidad.

```
In[15]:=
 f'[x]


In[16]:=
 FullSimplify[%]
```

De donde se sigue que `f'[x]` es positiva para $x > 0$. En consecuencia `f[n]`>1 para todo $n \geqslant 4$. Por tanto, **sumcuad** aplicado a cualquier número con $n \geqslant 4$ dígitos, será un número que como máximo tendrá $n-1$

dígitos. Deducimos que si iteramos **sumcuad** tomando como valor inicial cualquier número natural, después de un número finito de iteraciones habremos obtenido como resultado un número comprendido entre 1 y 9999. Pero entonces sabemos que con una o dos iteraciones más habremos obtenido un número comprendido entre 1 y 243. Si comprobamos ahora que partiendo como dato inicial x_0 de un número entre 1 y 243 las iteraciones de **sumcuad** recaen en un punto fijo o en un ciclo de período 8, habremos probado que lo mismo ocurre para cualquier $x_0 \in \mathbb{N}$.

La parte de la demostración con ayuda del ordenador

Es claro que un matemático de principios del siglo XX que ni siquiera tuviera a su disposición una calculadora podría llevar a cabo el resto de la demostración. A fin de cuentas, sólo tendría que comprobar 243 casos lo que, ciertamente, le llevaría mucho menos tiempo que ciertos cálculos astronómicos que fueron hechos con éxito hace ya varios siglos. Lo más llamativo de este asunto es que muy probablemente un matemático de principios del siglo XX o anterior, no habría llegado a hacer la conjetura que estamos demostrando.
Bien, todavía hay trabajo que hacer. Tenemos que verificar 243 casos.

```
In[17]:=
 Cases[Map[test,Table[NestList[sumcuad,k,15],
 {k,243}]],"ni lo uno ni lo otro"]
```

Como queríamos demostrar.

Problemas

1 Hemos iterado siempre 15 veces la función **sumcuad**. Se eligió este número de iteraciones porque, de una parte, parecía suficiente para que aparecieran los dos comportamientos conocidos y, de otra, no era demasiado grande como para dificultar los cálculos ni su visualización en pantalla. Calcula lo grande que debe ser la condición inicial para que después de 15 iteraciones no se obtenga como resultado un número comprendido entre 1 y 9999.

2 Todo lo anterior es válido también para las sumas de las potencias $q-$ésimas de los dígitos. Prueba que para $q = 3$ la sucesión de las iteraciones tiene un número finito de posibles ciclos o un punto fijo. ¿Cuántos y cuales son estos ciclos? Repite esto con $q = 4$ (esto te costará bastante más trabajo).
Sugerencia: El número de comportamiento periódicos posibles va en aumento. Para no gastar tiempo mirando las listas de resultados y tratando de descubrir los comportamientos que allí aparecen, define una función **periodo** que descubra el período de un ciclo en el caso de que este ocurra. Otra posibilidad es usar el comando `Union[lista]` para eliminar de `lista` todos los elementos repetidos.