Software matemático libre

por

Miguel Á. Abánades, Francisco Botana, Jesús Escribano y Luis F. Tabera

1. Introducción

El *software libre* (o software "open source" [1]) está cobrando, en diversos ámbitos, una importancia cada vez mayor, de forma que está dejando de ser algo propio de especialistas (o de *freaks* informáticos) y está pasando a ser algo conocido (o al menos utilizado) por un número cada vez mayor de personas. Por ejemplo, el fracaso del sistema operativo Windows Vista (de la compañia Microsoft), a pesar de los 4000 ingenieros participantes en su desarrollo, ha animado a muchos usuarios a instalar distribuciones de Linux como Ubuntu, que ofrecen sistemas de instalación sencillos y seguros. Otro ejemplo de software libre ampliamente utilizado es el navegador web Firefox.

Podemos definir software libre como aquel software para el que tenemos:

i) Libertad para ejecutarlo en cualquier sitio, con cualquier propósito y para siempre.

ii) Libertad para estudiarlo y adaptarlo a nuestras necesidades. (Esto exige el acceso al código fuente).

iii) Libertad de redistribución, de modo que se nos permita colaborar con colegas, alumnos, ...

iv) Libertad para mejorar el programa y publicar mejoras. [8]

¿Qué impacto tiene el software libre en el mundo matemático? Por supuesto, todos sabemos que el estándar *de facto* en la edición matemática es TEX/LATEX, una de las joyas de software libre. Sin embargo, creemos que el impacto es, y va a ser, mucho mayor, tanto en la docencia de las matemáticas como en la investigación matemática.

En cuanto a la docencia, la utilización de programas informáticos es cada vez más común en el aula, a todos los niveles. Programas para realizar diversos cálculos, para representar funciones o configuraciones geométricas son cada vez más utilizados. Sin embargo, los precios de las licencias de estas herramientas, y su dificultad de acceso, pueden limitar a veces su utilización. En cambio, el uso de programas libres puede

[1] Los conceptos de software libre y software de código abierto no son idénticos, y de hecho, hay varias diferencias de matiz. Sin embargo, en cuanto a este artículo se refiere, podemos considerar ambos conceptos equivalentes.

facilitar el acercamiento de estos programas a los alumnos y a los profesores (tanto en el aula como, sobre todo, en la casa), por su inmediato acceso gratuito. En los últimos tiempos han aparecido interesantes aplicaciones en este sentido. En esta nota, entre otras aplicaciones, destacaremos el programa *GeoGebra*, un sistema de geometría dinámica de gran ayuda para la enseñanza de la Geometría.

Un argumento similar se puede utilizar también para el software utilizado en investigación matemática. Por ejemplo, muchos profesores universitarios utilizan programas de cálculo simbólico (CAS) como Mathematica o Maple. Estos programas tienen precios que pudieran ser calificados como excesivos, que son pagados con resignación por las grandes universidades, pero que pueden poner en dificultades a departamentos pequeños o a investigadores con presupuestos limitados. Pero no nos queremos quedar en las limitaciones económicas. Diversos estudios han puesto en duda la fiabilidad de algunos programas de cálculo simbólico [5]. Lo que es peor, las empresas responsables no se han mostrado muy dispuestas a admitir sus errores y a explicar claramente qué algoritmos utilizan para sus cálculos. El código es cerrado y por tanto no nos es posible saber cómo se realiza una determinada operación, y mucho menos tenemos esperanzas de poder aportar una solución útil para la comunidad matemática. Por tanto, para poder realizar una investigación sólidamente fundamentada en software fiable, es necesario disponer del código de las aplicaciones que estamos utilizando. En esta nota tratamos con detalle esta problemática, y proponemos una alternativa desde el software libre a los CAS propietarios: SAGE.

2. Una breve historia del software libre

Citamos de [8]:

> Desde hace más de 30 años nos hemos acostumbrado a que quien me vende un programa me impone las condiciones bajo las que puedo usarlo, prohibiéndome, por ejemplo, que se lo pase a un amigo. A pesar de ser software, no puedo adaptarlo a mis necesidades, ni siquiera corregir errores, debiendo esperar a que el fabricante los arregle. Esto no tiene por qué ser así, y es precisamente el software libre el que me concede las libertades que el software propietario me niega.

Si la historia de la informática es necesariamente breve, podemos decir que el software libre está presente desde el principio, o mejor, que el software nació inicialmente libre. En los inicios de la informática, el software se solía distribuir con el hardware correspondiente y el código era totalmente accesible. Esto va poco a poco cambiando, y a principio de los años 70 aparece más y más software propietario, separado del hardware. Sin embargo, surgen importantes iniciativas de software libre, como *Spice*[2] o TEX.

Al hablar de la historia del software libre, es necesario hablar de Richard Stallman. En 1984, Richard Stallman abandona el Massachusetts Institute of Technology (MIT) y empieza a trabajar en la idea de construir un sistema de software completo,

[2] Simulation Program with Integrated Circuit Emphasis, un simulador de circuitos integrados.

de propósito general, pero completamente libre. El proyecto se llamó GNU[3]. Preocupado por establecer de forma clara las licencias de los usuarios, escribió la licencia GPL (General Public License). También fundó la *Free Software Foundation* (FSF) para conseguir fondos para su trabajo y sentó los fundamentos éticos del software libre [23]. Toda una serie de trabajos se desarrollan a lo largo de los años 80 y 90. Finalmente, el proyecto GNU confluye con las ideas de un joven estudiante finés, Linus Torvalds, y aparece GNU/Linux, el sistema operativo libre más conocido y utilizado. Por supuesto, el trabajo se mantiene, con plena vigencia, en la actualidad (GNOME, KDE, Ubuntu, OpenOffice, Firefox, ...), como veremos a lo largo de este artículo.

Figura 1: Richard Stallman y Linus Torvalds

El término *software libre* fue definido por Stallman mediante las libertades mencionadas en las introducción: Libertad de ejecución, libertad de redistribución y libertad de estudio, modificación y mejora de los programas, junto con la libertad de publicación de las mejoras. Para todo ello, es necesario tener acceso al código fuente.

Muy relacionado, pero distinto, es el concepto de *programas de código abierto*, promovido por Eric Raymond y la Open Source Iniciative. Ambos términos son muy distintos desde el punto de vista filosófico, ya que el segundo hace más énfasis en la disponibilidad del código fuente que en la libertad, pero desde luego tienen muchos puntos en común.

Aquí podemos ver las dos motivaciones principales para el desarrollo del software libre:

- La motivación ética (Free Software Foundation), que argumenta que el software es conocimiento y debe poderse difundir sin trabas, y por tanto, considera su ocultación como un hecho antisocial; y
- La motivación pragmática (Open Source Iniciative), que argumenta ventajas técnicas y económicas.

[3]Acrónimo recursivo, GNU's Not Unix.

Las ventajas en las utilización de software libre están ampliamente documentadas: para el usuario, para el desarrollador, para el integrador, para el mantenimiento de servicios, ... Destacaremos las ventajas para la Administración pública. La Administración pública es un "usuario" muy especial, con obligaciones especiales con el ciudadano: servicios accesibles, neutrales respecto a los fabricantes, en los que se garanticen la integridad, privacidad y seguridad de los datos de los ciudadanos. Esto le obliga a ser respetuosa con los estándares.

Por cierto, destacamos que "software libre" y "software gratuito" no son la misma cosa, a pesar de que en inglés se utilice la misma palabra, "free". El software libre puede ser gratuito o no (por ejemplo, se puede comprar una copia en CD de una distribución de Linux), pero siempre debe ser posible redistribuirlo libremente y siempre se debe tener acceso al código fuente (lo que no necesariamente pasa con el software gratuito)

3. Uso en docencia

La nueva sociedad de hoy, la sociedad de la información y el conocimiento, requiere de nuevos enfoques formativos que nos permitan *aprender a aprender* para seguir formándonos toda la vida. (Aunque a veces se tiene la sensación de que no se busca una mayor formación, sino una mayor versatilidad, una más adecuada capacidad para desempeñar distintas tareas en el sistema productivo a lo largo de la vida, lo que algunos llaman la *flexiseguridad*.) Si el proceso de formación continua ha sido necesario siempre, no cabe duda de que actualmente su papel es crítico. En este aspecto, el aprendizaje de (y con) las nuevas tecnologías desde una fase temprana del desarrollo educativo juega un papel fundamental. Contenidos más dinámicos, mayor flexibilidad de adaptación, interactividad o facilidad en la actualización de contenidos son algunas de las ventajas que ofrece la introducción de las tecnologías de la información y la comunicación (TIC) en las aulas de cualquier nivel educativo.

En el caso particular de la enseñanza de las matemáticas, se resume muy bien en [11]:

> *El uso del ordenador en clase de Matemáticas favorece la adquisición de conceptos, permite el tratamiento de la diversidad y el trabajo en grupo, y es un elemento motivador que valora positivamente el error.*

Si bien el uso de las TIC en el aula parece positivo, hay que tener cuidado con el *cómo*. Incorporar las nuevas tecnologías al aula requiere que se explicite un modelo pedagógico de uso de las mismas. Lamentablemente en muchas aulas las tecnologías son empleadas únicamente como un medio nuevo al servicio de métodos tradicionales de enseñanza (lecciones magistrales, ejercicios repetitivos, control de aprendizaje memorístico, ...). Esto queda reflejado con humor amargo en el vídeo de animación *Tecnologia e Metodologia* [31] del *Grupo de trabalho de Imagem e Conhecimiento* de la *Universidade Presidente Antonio Carlos* de Brasil que muestra cómo se puede usar la más moderna tecnología para no cambiar nada. En la figura 2 se muestran dos fotogramas con el aula repitiendo las mismas tablas de multiplicar antes y después de las incorporación de las TIC.

Figura 2: Nuevas tecnologías, viejas tablas de multiplicar.

Uso de las TIC en las aulas españolas

El exhaustivo estudio acerca del uso de ordenadores e Internet en la educación preuniversitaria en Europa publicado por la Comisión Europea el 29 de septiembre de 2006 [7] muestra que los avances en la integración de las TIC en el aula son firmes en todos los países. En particular, se recoge que en España la media de estudiantes por ordenador es de 10,5, cuando en 2001 era de 14, acercándonos de este modo a la media europea (que es 9). Se destaca también el hecho de que en España la proporción de centros de educación primaria y secundaria con acceso a Internet por banda ancha sea del 80,7 %, muy superior al 67 % de la media europea y al del conjunto de la sociedad española. Sin embargo, esta disponibilidad de medios en los centros educativos españoles, no se traduce en un uso real de los mismos por parte del profesorado.

No sorprende que sean los países con mejores resultados educativos (Reino Unido, Holanda, países escandinavos) los más activos en la integración de las nuevas tecnologías en el currículo de todas las asignaturas. Mientras en el Reino Unido un 95,2 % de las escuelas utilizan ordenadores en el aula y un 94,7 % de los profesores que trabajan en esas escuelas consideran que los ordenadores e Internet están integrados en la mayoría de las asignaturas, en España estos porcentajes son muy inferiores: el 47,6 % y el 79,9 % respectivamente.

Si añadimos que un 93 % del profesorado declara tener suficientes conocimientos de nuevas tecnologías, debido fundamentalmente a un uso personal de las mismas, parece que ni la falta de medios ni la falta de formación son responsables de la baja integración de las TIC en las aulas españolas.

La gran diferencia entre el uso doméstico y el uso docente (sobre todo el uso como elemento curricular) de las TIC por parte del profesorado tiene que ver, entre otras cosas, con el hecho de que en el centro escolar el profesor es un usuario de un sistema parecido pero distinto al propio (con distintas aplicaciones, distintas versiones, etc.), produciéndose un distanciamiento por parte del profesor, no exento de temor, que no propicia el clima de confianza necesario a la hora de introducir cualquier novedad en el aula. En este punto las herramientas de software libre tienen mucho que aportar. La posibilidad para profesores (y alumnos) de tener en casa (¡legalmente!) las mismas

aplicaciones y versiones que en el aula creemos que podría ayudar en gran medida a aumentar los porcentajes de uso de las TIC como parte integrante del proceso de enseñanza aprendizaje.

En el caso particular de la enseñanza de las matemáticas, en cuyo profesorado los conocimientos tecnológicos están mucho más extendidos, la escasa implantación de las TIC en el aula (ver sección 3.1) está si cabe menos justificada.

La relevancia del software libre en entornos educativos queda patente por el hecho de que en la *Conferencia Internacional de Software Libre* organizada por la Junta de Andalucía y la Junta de Extremadura, y celebrada en Málaga en octubre de 2008, hubiese dos sesiones temáticas y dos talleres dedicados a la relación del software libre con la educación [34].

Aspectos técnicos como criterios de uso

En [19] Moreno nos indica que, aparte de la fundamentación educativa, los marcos de referencia fundamentales para establecer criterios de uso de las TIC en la práctica docente son la funcionalidad de los medios, sus posibilidades didácticas y los aspectos técnicos.

Respecto a los aspectos técnicos a tener en cuenta, el mismo autor indica que los criterios más relevantes son

- Facilidad de adquisición y existencia de servicio postventa
- Precio
- Sencillez de manejo y manipulación
- Sencillez de mantenimiento y control
- Movilidad y portabilidad
- Compatibilidad con otras aplicaciones
- Adecuación a las necesidades e instalaciones
- Flexibilidad de utilización
- Facilidad de actualización
- Posibilidad de trabajar en un entorno multiárea y multiusuario

Si bien creemos que las ventajas del software educativo libre no se reducen únicamente a las referentes a sus características técnicas, como se ilustra en la sección 3.1.1, sin duda son estas las más visibles. En particular, incluso los más escépticos estarían de acuerdo en que las herramientas de software libre optimizan todos y cada uno de los diez puntos anteriores.

El uso de formatos abiertos y compatibles es una característica del software libre que sólo redunda en ventajas para el usuario final. Si acaso, un usuario acostumbrado a un programa específico de software propietario (asistente matemático, editor de textos, hoja de cálculo, ...) es probable que se vea cautivo, es decir, llevado a continuar con el mismo fabricante debido a que los sistemas cerrados suelen usar formatos de archivo propietarios que impiden la interoperabilidad con otro software.

Especialmente grave es el uso por parte de las administraciones públicas del uso de formatos propietarios en la comunicación con el ciudadano que obligan a usar un determinado proveedor de software.

Esto, en particular, hace que la enseñanza sea un jugoso objetivo, no sólo por el fabuloso negocio que representa para algunas empresas el dotar al sistema educativo de todo el material requerido, o por las ventas de licencias de software a los centros de enseñanza. Además, los alumnos que se formen en estos programas probablemente sean la base de usuarios de los mismos en su vida adulta. El propio hecho de que en el primer criterio de elección se mencione el servicio *postventa*, dando por sentado un modelo de software propietario, es otro síntoma del mismo problema.

Creemos que el software libre puede cambiar la manera en que se entiende la relación de los usuarios con las aplicaciones informáticas. Y creemos firmemente que esto es todavía más cierto en el caso de las herramientas educativas en general y matemáticas en particular como se muestra en las siguientes secciones.

3.1. Docencia preuniversitaria

Aunque el uso de los ordenadores en las clases de matemáticas de educación infantil y primaria está cada vez más extendido, las herramientas utilizadas son generalmente materiales preparados en forma de juegos (applets, animaciones flash, páginas webs,...) disponibles en repositorios institucionales (por ejemplo CNICE [33], ZonaClic [28]) que no corresponden exactamente al concepto de software matemático considerado en esta nota. Nos centraremos pues en esta sección en el impacto del software libre matemático en el caso de la educación secundaria.

Estando esta etapa educativa totalmente regulada por ley, conviene indicar cómo recoge la LOE [14] el uso de las nuevas tecnologías en la descripción general de sus dos etapas. En su artículo 23, donde se especifican los 12 objetivos generales de la Enseñanza Secundaria Obligatoria, se puede leer (el énfasis es nuestro) que

> *La educación secundaria obligatoria contribuirá a desarrollar en los alumnos y las alumnas las capacidades que les permitan:*
>
> *(e) Desarrollar destrezas básicas en la utilización de las fuentes de información para, con sentido crítico, adquirir nuevos conocimientos. Adquirir una preparación básica en el campo de las* ***tecnologías****, especialmente las de la* ***información*** *y la* ***comunicación****.*

Del mismo modo en el artículo 33, donde se detallan los 14 objetivos del Bachillerato encontramos el siguiente:

> *(g) Utilizar con solvencia y responsabilidad* ***tecnologías*** *de la* ***información*** *y la* ***comunicación****.*

De este modo la incorporación de las TIC en nuestras aulas no es sólo un hecho deseable, sino una obligación por ley.

Sin embargo, del excelente informe *El profesorado de Matemáticas ante las Tecnologías de la Información y la Comunicación* publicado en esta misma revista en

2006 [20] se desprende que la realidad es que aunque el 80 % del profesorado muestra una predisposición personal y profesional positiva hacia la integración de las TIC en las clases de matemáticas, sólo el 34,5 % de los profesores de matemáticas hace que sus alumnos utilicen las TIC para el aprendizaje de las matemáticas. Esto convierte pues la integración de las TIC en el proceso de aprendizaje de las matemáticas en un fenómeno ciertamente minoritario. Si bien este informe describe la situación del uso de las TIC hace más de dos años, a falta de estudios más actualizados, tomaremos los datos que aporta como aproximación razonable a la realidad.

Ya apuntábamos algunas de las causas del bajo índice de integración de las TIC en el proceso de aprendizaje en general. Aunque la principal causa parece ser la falta de una instrumentalización adecuada de las TIC, en el citado estudio de Pérez [20] se pueden analizar datos que nos dan pistas sobre otros problemas encontrados por los profesores de matemáticas en su camino (¡los que lo emprenden!) hacia el uso de las TIC en el aula.

Según este estudio, el 80 % de los profesores achacan las dificultades en el uso de las TIC en el aula a la propia organización escolar (horarios, duración de las clases, distribución de espacios, etc.) y además dos de cada tres manifiestan inseguridad en sus conocimientos técnicos.

Pero el dato más interesante, desde el punto de vista de esta nota, es que más de dos tercios del profesorado opina que el software disponible es insuficiente y señala la escasez de herramientas educativas específicas como una de las causas de la escasa utilización de las TIC.

Dada la proliferación de herramientas de software libre específicas en cualquier área de Matemáticas (Álgebra, Cálculo, Geometría, Estadística,...ver [36]), el hecho de que tal proporción de los profesionales docentes opinen que no hay herramientas adecuadas disponibles es sin duda un signo de la gran desinformación existente respecto del software libre. Una apuesta a nivel institucional por el uso generalizado de herramientas de software libre no hará más accesibles las herramientas (están ahí, sólo hay que *cogerlas*), pero ayudará sin duda a romper con la filosofía imperante de software propietario.

Software libre educativo de Matemáticas

El mismo estudio de Pérez indica además cuáles son los programas más utilizados por los profesores de matemáticas en secundaria. Si no contamos el uso de los applets de Descartes elaborados por el CNICE, que son usados por un 23 % de los profesores que usan las TIC en clase, el porcentaje de uso entre estos profesores de los programas más significativos es el siguiente:

- **Hoja de cálculo**: 58 %
- **Derive**: 50 % (programa comercial para cálculo matemático avanzado: variables, expresiones algebraicas, ecuaciones, funciones, vectores, matrices, trigonometría, etc. Con capacidades de calculadora científica, puede representar funciones gráficas en dos y tres dimensiones).

- **Cabri**: 38 % (software comercial de geometría dinámica. Permite la representación e interacción dinámica con construcciones geométricas bidimensionales).

Teniendo en cuenta que los programas habituales de tipo hoja de cálculo forman parte de los paquetes usuales de ofimática de uso generalista, podemos decir que estadísticamente el uso de programas específicos de matemáticas se reduce a dos: *Derive* y *Cabri*.

Considerando la queja apuntada anteriormente sobre la falta de software específico por parte del profesorado de matemáticas, parece razonable pensar que el uso casi exclusivo de estos dos programas no se debe tanto a la demanda de los profesores, sino a, lo que es muy distinto, la escasa disponibilidad de alternativas en los centros. En este punto, el software libre se presenta de nuevo como una solución. En particular, para cada uno de los tres programas más usados, existen excelentes programas de acceso libre. A continuación se indican los tres más relevantes, no queriendo decir esto que no existan otros igual de apropiados.

- **OpenOffice.org** [39] es una suite ofimática de software libre y código abierto de distribución gratuita que además de otros programas incluye una aplicación para la gestión de hojas de cálculo denominada *Calc* compatible y similar a Microsoft Excel.
- **SAGE** es un sistema algebraico computacional (CAS) que reúne bajo un solo entorno diversos paquetes de cálculo matemático avanzados (teoría de grupos, geometría algebraica,...). Más detalles en la sección 4.1.
- **GeoGebra** es un sistema de geometría dinámica, cuyo motor de cálculo es software libre, que añade capacidades algebraicas, estableciéndose una relación directa entre los objetos de la ventana algebraica y los de la ventana geométrica. Más detalles en la sección 3.1.1.

Si bien la lista anterior recoge aplicaciones informáticas de solidez técnica incuestionable, no son estas características técnicas lo más relevante respecto de las herramientas matemáticas de software libre. Alrededor de cada una de las aplicaciones mencionadas hay miles de usuarios comprometidos con la aplicación. Algunos comparten ficheros en *wikis*, otros comparten conocimientos avanzados en foros y algunos hasta colaboran como desarrolladores de las nuevas versiones.

Sin duda son estas *comunidades* surgidas a su alrededor el mayor valor de las herramientas de software libre, ideadas como proyectos colaborativos altamente participativos como ilustra la sección 3.1.1.

3.1.1. El caso de GeoGebra

Como se ha empezado a apuntar brevemente en la sección anterior, GeoGebra es un software matemático interactivo para educación secundaria con funcionalidades para el estudio de la Geometría, el Álgebra y el Cálculo.

Por un lado, GeoGebra es un sistema de geometría dinámica, es decir permite realizar construcciones geométricas planas que a posteriori pueden modificarse dinámi-

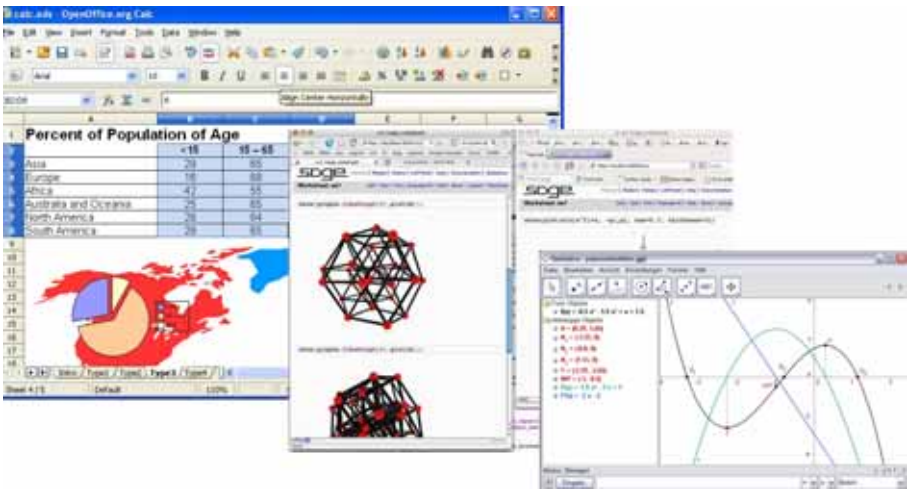

Figura 3: Alternativas libres

camente. Por otra parte, se pueden introducir ecuaciones y coordenadas directamente. Así, GeoGebra tiene la potencia de manejar variables vinculadas a números ofreciendo un repertorio de comandos propios del análisis matemático, aptos para tareas como identificar puntos singulares de una función.

Estas dos perspectivas caracterizan a GeoGebra: una expresión en la ventana algebraica se corresponde con un objeto en la ventana geométrica y viceversa (ver figura 4).

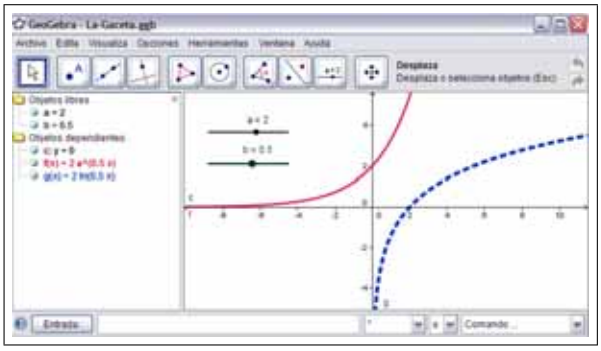

Figura 4: Ventanas algebraica y geométrica de GeoGebra.

Pero no queremos dar aquí muchos más detalles técnicos de un software que fue extensa y profundamente descrito por el experto Rafael Losada en su magnífico artículo *GeoGebra: la eficiencia de la intuición* de 2007 en esta misma revista [16]. Más que en la herramienta GeoGebra, preferimos centrarnos en el *fenómeno* GeoGebra.

GeoGebra surgió en 2001 como el trabajo de fin de máster en Educación Matemática en la Universidad de Salzburgo (Austria) de Markus Hohenwarter, por entonces profesor de instituto. Lo que se suponía que iba a ser una herramienta menor, casi de uso personal, ganó en 2002 el premio de la academia europea de software (EASA) en la categoría de Matemáticas [27] y en 2003 el premio al mejor software académico austriaco. Según el propio Hohenwarter relata [10], se vio entonces *obligado* a continuar con el proyecto que se convirtió en el tema central de su tesis doctoral en la misma universidad.

Desde entonces, la herramienta, que se sirvió del boca a boca e Internet para distribuirse rápidamente por todo el mundo, se ha convertido en un proyecto colaborativo con cifras que hablan por sí solas:

- Usuarios en 190 países (¡La ONU tiene 192 estados miembros!)
- Versiones en 44 idiomas
- Medio millón de visitas mensuales a su página web

GeoGebra se distribuye de manera gratuita a través de internet [29]. El motor que realiza los cálculo es software libre (GNU GPL). Sin embargo, la documentación, paquetes de idioma e instalador, se distribuyen con una licencia (Creative Commons Attribution-NonCommercial-NoDerivs 3.0) que no permite distribuir modificaciones, ni su uso en aplicaciones comerciales. Esto hace que el conjunto no sea software libre según la definición que hemos dado al principio del artículo. [4]

Es importante indicar que entre los 44 idiomas disponibles se encuentran, además del castellano, el vasco, el gallego y el catalán. No hace falta subrayar que la disponibilidad de versiones en el idioma adecuado es un criterio crucial de elección de las herramientas educativas en todas las etapas preuniversitarias.

Un aspecto a destacar es que el enorme apoyo popular ha venido acompañado de un gran interés académico como muestran las tesis de máster sobre GeoGebra leídas en la universidad de Cambridge, Inglaterra [4, 17] o la de doctorado en la universidad de Salzburgo, Austria [21].

En esta misma línea, el interés de su creador de desarrollar todo el potencial educativo de GeoGebra, le ha llevado a impulsar la creación de una red de *Institutos GeoGebra Internacionales* (IGI) [30] que sirven como plataforma desde la cual profesores e investigadores de todo el mundo trabajan juntos para promover la docencia de las matemáticas. Más concretamente, desde los IGI se pretende apoyar y coordinar actividades entre las que destacan el desarrollo de:

- Materiales docentes para talleres (por supuesto de libre disposición)
- Talleres formativos para profesores y futuros formadores de GeoGebra
- Mejoras y nuevas funcionalidades para GeoGebra
- Un sistema online de apoyo a profesores
- Proyectos de investigación sobre GeoGebra
- Presentaciones en congresos nacionales e internacionales

[4] Este sería un ejemplo de software que es open source pero no libre.

La red de IGI, comenzada con la inauguración del IGI de Noruega en septiembre de 2008, cuenta en la actualidad con ocho centros en distintos países (desde Austria a EE.UU., pasando por Turquía) entre los que hay que destacar el primer IGI español en Barcelona [26]. Próximamente hay prevista la inauguración de 10 más (Egipto, Costa Rica, Suecia...), entre los que se encuentra uno en Cantabria, con vocación estatal.

Un ejemplo del impacto que el carácter libre de GeoGebra ha tenido en el desarrollo de su *comunidad* queda ilustrado por la anécdota recientemente relatada por el propio Hohenwarter [10]: en el congreso internacional de educación matemática (ICME) celebrado en México en 2008 [32], un profesor de Filipinas se le acercó para comentarle que en su país había proyectos alrededor de GeoGebra en los que participaban miles de profesores de secundaria.

3.2. Docencia universitaria

La variedad de herramientas informáticas empleadas en la docencia a nivel universitario es amplia. Sin embargo, sólo algunas aplicaciones parecen capaces de cambiar fundamentalmente el proceso de enseñanza y aprendizaje de las matemáticas. Sin lugar a dudas, los CAS, que combinan capacidades de cálculo simbólico, numérico y gráfico, caen en esta categoría, y de hecho ocupan un primer puesto destacado como la herramienta más utilizada en docencia matemática universitaria [1, 13]. Entre los CAS, los más conocidos son Derive [15], Maple [6], Mathematica [3] y MatLab [18].

La popularidad de los CAS se debe no sólo a su gran versatilidad, siendo usados en una gran variedad de áreas matemáticas, sino a su potencial para convertirse en el principal utensilio de la caja de herramientas matemáticas del estudiante y posteriormente del matemático profesional. De hecho, hay estudios que sugieren que la destreza adquirida en estas complejas herramientas durante la etapa de formación influye directamente en el tipo de trabajo y estudios posteriores [2].

Si bien no entraremos aquí a valorar la necesidad de que en la enseñanza universitaria de primer ciclo se escoja un CAS de referencia (como es el caso en Austria con la adopción de Derive como programa nacional para la enseñanza matemática preuniversitaria [5]) sí queremos señalar que hay situaciones de coexistencia de múltiples CAS que claramente son inadecuadas para formar a nuestros estudiantes.

En particular, y por no hablar de terceros, este es el caso en un departamento de Matemáticas de una pequeña universidad de provincias. En esta universidad, básicamente politécnica, se dispone de una licencia global de Matlab, pero de otras fuentes y por vías variadas, se usan en la enseñanza también Mathematica, Maple y Derive. Es más, en alguna asignatura impartida por distintos profesores, es posible encontrar clases de prácticas desarrolladas con distintos CAS.

Aunque la elección del óptimo (¡u óptimos!) se escapa de los objetivos de esta nota, es razonable pensar que esta multiplicidad de programas no propicia precisamente el proceso educativo, especialmente en lo relativo a los CAS, caracterizados

[5] Debemos mencionar, en este caso, el peligro de que la compañia que distribuye un programa desaparezca, o sencillamente deje de mantener el producto, como ha pasado con Derive.

por tener una curva de aprendizaje muy dura. A las consideraciones anteriores hay que añadir que no siempre el estudiante dispone para su uso particular de licencias de los programas correspondientes.

En este punto, los argumentos a favor del uso de herramientas matemáticas de software libre son similares a los apuntados en el caso de la docencia preuniversitaria: libre disponibilidad y portabilidad para profesores y estudiantes. En particular, un candidato excelente para convertirse en un sustituto libre de los CAS mencionados es SAGE, descrito en detalle en la sección 4.1. De hecho, en la web de SAGE [35] se puede leer que su misión es:

> *Crear una alternativa viable de software libre a Magma, Maple, Mathematica y Matlab.*

Pero además de los argumentos de libre disponibilidad y portabilidad, en la etapa universitaria cobran relevancia también los argumentos de transparencia. Las herramientas comerciales mencionadas permiten al profesor elaborar materiales para su uso a través de internet (cuya disponibilidad asumimos universal entre los estudiantes universitarios). Por ejemplo, Mathematica ofrece *webMathematica*, que permite al alumno interactuar remotamente con partes escogidas del núcleo de Mathematica. Aunque esto es interesante por el ahorro al usuario de tediosos cálculos, la experiencia de varios años de sucesivo uso nos hace pensar que este tipo de aplicaciones, concebidas básicamente como *cajas negras*, no propician una reflexión sobre el problema considerado sino que más bien actúan a modo de oráculo que adecuadamente interrogado devuelve la respuesta requerida.

De nuevo estas reflexiones nos llevan a señalar a SAGE, por su naturaleza de herramienta de código abierto, como una herramienta mucho más adecuada. Hablaremos con más detalle de SAGE en la sección 4.1.

4. Uso en investigación

¿Qué puede aportar el software libre a la investigación en Matemáticas? Ya hemos mencionado la posibilidad de acceso libre e inmediato a determinados recursos informáticos. Aparte de esto, en este misma *Gaceta* [5], ya se ha formulado la pregunta de hasta qué punto puede uno fiarse de los cálculos efectuados por un Sistema de Cálculo Simbólico (CAS). En general, estos cálculos serán aceptados como ciertos si y solo si son *demostrables*, esto es, si podemos proporcionar una prueba formal. El problema es que en mucho momentos, los cálculos no pueden comprobados, debido a la naturaleza propietaria y opaca de los CAS más utilizados. Si, por ejemplo, consideramos la integral

$$\int_0^\infty log(z)\frac{1-cos(z)}{z}dz.$$

Mathematica 3.0 y 4.1 dan $1/24(-12\gamma^2+\pi^2)$ como resultado. La versión 5.0 da el mismo valor, pero nos avisa que *no ha sido posible comprobar la convergencia*. Finalmente, la penúltima versión comercializada, 6.x, sí que es capaz de detectar

la no convergencia de la integral, y contesta con la expresión original, pero sólo si una opción, aparentemente irrelevante, es invocada (ver detalles en [24]). Este es el mismo comportamiento de la versión 7. En [5] podemos encontrar otros ejemplos relacionados con Mathematica, e interesantes reflexiones sobre la fiabilidad de los CAS en general.

Si alguien intenta profundizar en el estudio de este comportamiento de Mathematica, se encuentra rápidamente con dificultades: Wolfram Research Inc. considera el conocimiento una propiedad particular, y proteje su código haciéndolo inaccesible:

> For the internals of Mathematica are quite complicated, and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances. [25] [6]

Esta afirmación, que ha sido objeto de un artículo de opinión en las *Notices* de la AMS por Joyner y Stein [12], es confrontada en el mismo artículo con la siguiente afirmación de J. Neubüser, creador de GAP:

> With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. [7]

Por esta razón, creemos que es importante que existan herramientas (por ejemplo, CAS) de código abierto, que nos aporten la fiabilidad que nos ocultan los programas propietarios. Esto nos lleva a hablar de SAGE.

4.1. SAGE

Cuando se habla de software libre en matemáticas se puede decir que hay mucho y bueno. Incluso la mayoría de los programas de código cerrado de matemáticas suelen utilizar librerías que provienen del software libre, por ejemplo, la librería libre GMP de cálculo aritmético de precisión arbitraria se utiliza en Maple [40], Magma [41] o Matlab [42]. También se usa en Maple y Magma las librerías ATLAS o las rutinas de LAPACK en Matlab. Estos tal vez sean de los ejemplos más populares de librerías matemáticas ampliamente utilizadas.

Sin embargo, la situación usual del software libre en matemáticas es el de un programa o librería que resuelve un problema concreto y, generalmente, bastante especializado. También existen programas de uso más general en un dominio específico. Maxima como sistema de álgebra computacional, Octave para cálculo numérico,

[6] La parte interna de Mathematica es bastante complicada, e incluso dada una descripción básica del algoritmo usado para un propósito particular, por lo general es muy difícil llegar a una conclusión fiable acerca de cómo la aplicación detallada de este algoritmo se comportará en realidad en determinadas circunstancias.

[7] Con esta situación, dos de las más elementales normas de conducta en las matemáticas son violadas: En matemáticas la información se transmite de forma gratuita, y todo está abierto para su control.

Polymake en el estudio de politopos, Singular para el estudio de geometría algebraica, GAP en teoría de grupos, ... La lista de software existente se escapa del propósito y espacio de esta nota.

En 2004, William Stein inició un proyecto que pretende desarrollar un sistema de matemáticas computacional de propósito general, SAGE (*Software for Algebra and Geometry Experimentation*). A pesar de su nombre y de que originariamente se usara principalmente en teoría de números, actualmente se usa en muchos otros campos (estadística a través de R o cálculo numérico con Numpy/SciPy). La última versión lanzada es la 3.4. Según las notas de esta versión, su desarrollo incluye a más de 125 desarrolladores y, según William Stein en su blog, es usado por unas 10.000 personas [43].

Podríamos mostrar la potencia de cálculo de SAGE, a través de ejemplos de uso en distintos ámbitos, pero para ello es más conveniente ir a la página del proyecto [35] y ver los ejemplos y tutoriales que serán sin duda más claros y variados. Para el lector que esté más interesado en SAGE, resaltamos que en junio de 2009 se celebrarán en Barcelona los *SAGE days* [44], un workshop en el que se reunen desarrolladores y usuarios de este sistema. En esta nota preferimos sin embargo centrarnos en las características de SAGE que lo diferencian de otros programas de matemática computacional.

Primeramente, SAGE no pretende "reinventar la rueda", entendiendo esto como reimplementar algoritmos que ya existan. Si se quiere añadir una funcionalidad nueva que no esté ya implementada, en vez de implementarla de cero, la forma preferida de actuar es buscar software libre de calidad que ya implemente esta característica e incluirlo en SAGE.

Otra de las características de este sistema es el uso de Python como lenguaje base. A diferencia de otros sistemas de matemáticas computacionales, que proporcionan su propio lenguaje de programación, los desarrolladores de SAGE han elegido un lenguaje ya existente. Así, el usuario que aprenda a utilizar SAGE aprenderá a utilizar un lenguaje de scripting muy popular. Python es un lenguaje interpretado orientado a objetos, aunque también permite programación imperativa, funcional, funciones lambda, ... Es un lenguaje fácil de aprender y utilizar, ampliamente utilizado para interactuar con diversos programas (Blender, Gimp). Por ejemplo es uno de los lenguages más utilizados por la compañía Google para desarrollar sus productos y es esencial en el funcionamiento del portal YouTube. Por otro lado, Python es un lenguaje que está pensado para escribir código limpio que se pueda releer cómodamente, estas características lo hacen bastante atractivo en el ámbito de la docencia.

Python permite un acceso fácil a funciones y algoritmos disponibles en otras librerías o escritos en otros lenguajes, lo que es ideal para el objetivo de SAGE de incluir multitud de software libre matemático de diversas procedencias. Esto permite que, para mejorar la eficacia de los programas, las partes críticas se puedan escribir en otros lenguages compilados, como C/C++/Fortran e incluir el código resultante dentro de un programa que se pueda acceder a él con Python.

Volviendo a SAGE y a la idea de no reinventar la rueda, este trae por defecto una cantidad ingente de software matemático para multitud de tareas. Se incluye software

para álgebra (Maxima, SymPy), aritmética rápida (GMP, MPFR, MPFI, Quad-double, Givaro), geometría algebraica (Singular), teoría de números (PARI, NTL mwrank, ECM, FLINTQS, GMP-ECM) teoría de grupos (GAP), cálculo científico (GSL, SciPy, NumPy, cvxopt), estadística (R), imágenes (Matplotlib, Tachyon3d, Jmol) entre otros. Por si esto no fuera suficiente, SAGE contiene además multitud de interfaces para poder comunicarse con software externo que el usuario tenga instalado en su sistema, estas interfaces son tanto a programas libres (Octave, Polymake) como de código cerrado (Maple, Mathematica). Esto permite, hasta cierto punto, poder intercambiar información entre distintos programas de software para aprovechar lo mejor de cada uno. Veamos el siguiente ejemplo:

```
sage: R.<x,y,t>=QQ['x','y','t'];
sage: R1=singular(R);
sage: f=singular(x^2+y^2-1);
sage: g=maxima(y*x+t*x);
sage: h=maple(x^2-y^2);
sage: expand((f+g)*h)
-y^4-x*y^3-t*x*y^2+y^2+x^3*y+x^4+t*x^3-x^2
sage: maple("with(algcurves,parametrization)");
sage: maple(f).parametrization(x,y,t)
[2*t/(1+t^2), (-1+t^2)/(1+t^2)]
sage: singular(h).factorize()

[1]:
   _[1]=1
   _[2]=x-y
   _[3]=x+y
[2]:
   1,1,1
sage: maxima(h).factor()
-(y-x)*(y+x)
sage: factor(g)
x*(y+t)
```

Este código primeramente define como marco de trabajo el anillo de polinomios racionales con tres variables x, y, t. Define un polinomio f en Singular, un polinomio g en Maxima y un polinomio h en Maple. Después hace un sencillo cálculo para mostrar que se puede trabajar con ellos. Posteriormente, calcula una parametrización con Maple de la curva definida por el polinomio descrito en Singular, factoriza el polinomio definido en Maple con los algoritmos de Singular y Maxima y factoriza el polinomio g con los propios algoritmos de SAGE (que internamente usarán Singular o Maxima). Por supuesto, esta no es la forma usual de trabajar con SAGE pero es muy ilustrativo para demostrar la extrema flexibilidad de este programa a la hora de integrar otro software.

La interfaz de texto que proporciona SAGE por defecto puede ser un tanto incómoda, especialmente si se pretende programar cualquier algoritmo no trivial.

Por ello SAGE proporciona otra interfaz a través del *notebook*. Simplemente tecleando `notebook()` en la consola se tiene disponible una interfaz gráfica en su ordenador accesible desde cualquier navegador de internet moderno. La gran ventaja de esta interfaz gráfica es que se puede acceder a ella trivialmente desde la red. Podemos tener un servidor ejecutando SAGE y el notebook, mientras los ordenadores cliente se pueden conectar a este servidor desde su navegador preferido para poder ejecutar su código. Esta interfaz es muy cómoda para poder mostrar gráficos, poder programar y editar comandos ya ejecutados, tiene salida con calidad TEX/LATEXy permite guardar la sesión y cargar sesiones previamentes guardadas. Si se quiere probar SAGE sin instalarlo, una buena idea es probar alguno de los notebooks que están disponibles en internet, por ejemplo en `http://www.sagenb.org/` o `https://kimba.mat.ucm.es:9000`.

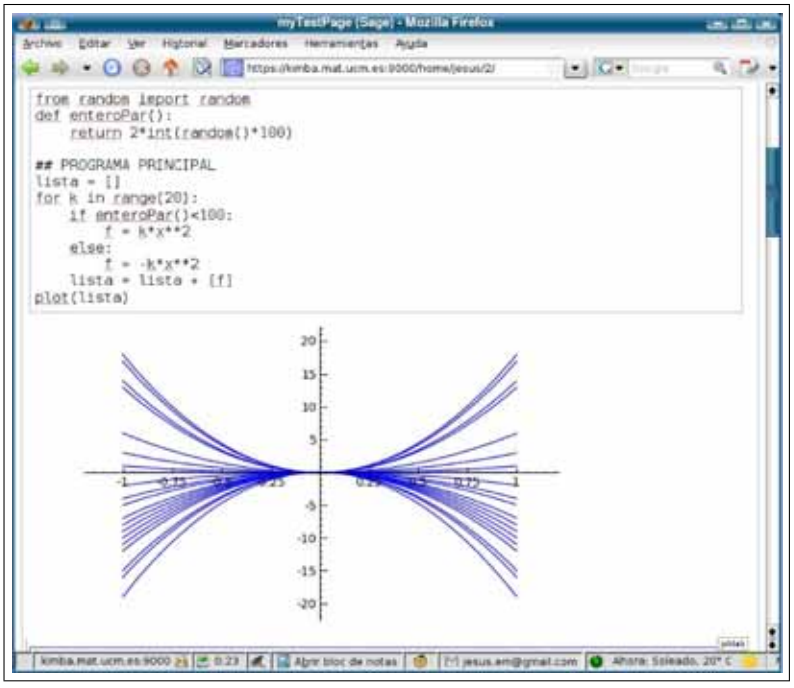

Figura 5: Servidor SAGE en https://kimba.mat.ucm.es

Sin embargo, debemos destacar que no son todo ventajas (en el modelo de desarrollo de software libre es común mostrar en público los problemas que tiene el software para que cualquiera pueda ayudar a solucionarlos). El principal problema de SAGE ahora mismo es que todavía no existe ninguna versión nativa para el sistema operativo Windows y, de momento, las opciones que se ofrecen para los usuarios de este sistema es utilizar virtualización. Este sistema es completamente funcional, pero conlleva una pérdida de rendimiento en la ejecución.

Por otro lado, la interfaz gráfica del notebook permite que el navegador de internet que se conecte al servidor ejecute código Python. Esto puede ser un problema de seguridad grave si no se tiene cuidado, especialmente para los servidores que permiten acceso desde internet a los notebooks. Por último, como todo software, SAGE no está exento de bugs, aunque la estrategia de desarrollo abierto permite que muchas personas colaboren en solucionar los problemas.

Es destacable que SAGE está pensado como una unidad. Las ventajas de este modelo son que facilita una distribución centralizada y también permite que los desarrolladores modifiquen los programas que componen SAGE para mejorar el rendimiento y la integración. El usuario sólo debe bajarse un fichero que contiene un directorio con todo SAGE en su interior, puede mover la carpeta sin que el programa deje de funcionar y para desinstalarlo, sólo debe borrar el directorio instalado. Esta es la misma estrategia que han seguido muchas aplicaciones de instalación en el sistema MacOSX, los paquetes klik en Linux o las portableapps en Windows. La desventaja de este modelo es que rompe con la idea de las librerías compartidas. Por ejemplo SAGE trae su propio intérprete de Python, su interprete de lisp y sus propias copias de Maxima y Singular. Por lo que el usuario que tenga alguno de estos programas ya instalado terminará con dos copias instaladas en el sistema. Esto dificulta que, a pesar de ser software libre, SAGE sea incluido en las distribuciones más populares de Linux para su instalación y mantenimiento automático.

4.2. La importancia del código abierto: El caso de la demostración de la conjetura de Kepler

> *Cubum autem in duos cubos, aut quadrato-quadratum in duos quadrato-quadratos, et generaliter nullam in infinitum ultra quadratum potestatem in duos eiusdem nominis fas est dividere cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exigitas non caperet.*

Estas cuatro líneas marginales dieron el pistoletazo de salida a un fecundo trabajo varias veces centenario cuyo resultado final fue un trabajo de un centenar de paginas publicado en los Annals of Mathematics en 1996 [22]. Como es natural, la afirmación de Fermat de haber demostrado el teorema no bastó para que este fuese considerado verdadero. Hubo que esperar al trabajo de Wiles para su general aceptación.

Otro problema nacido igualmente en el siglo XVII y referido a la mayor densidad de empaquetamiento de esferas en el espacio, la conjetura de Kepler, el 18º problema de Hilbert, fue también resuelto en la última década del pasado siglo. Pero si bien la solución de ambos problemas fue coetánea, no resulto igual de sencilla la general aceptación de las pruebas. Mientras que la de Wiles fue rápidamente aceptada (tras la corrección de un error en la demostración original), la solución de Thomas Hales a la conjetura de Kepler experimentó más dificultades.

A su descripción en un workshop celebrado en el Instituto de Estudios Avanzados de Princeton, en el que no se presentaron objeciones, siguió la invitación a publicarlo en los Annals, tras el habitual proceso de revisión. Pero puesto que la prueba completa comprende ocho artículos (algunos previamente publicados) a lo largo de 250 paginas y alrededor de 3Gb de datos y 40.000 líneas de código en Java,

C, C++, CPLEX y Mathematica [37], bastantes revisores rehusaron la tarea. Finalmente doce bondadosos matemáticos invirtieron varios años en la revisión hasta certificar la demostración.

Con la lectura del párrafo anterior el lector habrá experimentado una sensación de *déjà vu*: una situación similar se produjo cuando, en 1976, Appel y Haken usaron computadoras para desechar los 1936 posibles contraejemplos a la conjetura de los cuatro colores [9]. Se afirmó entonces que una demostración matemática es como un poema, pero la propuesta era una guía telefónica. Una crítica semejante fue hecha en Princeton a la prueba de Hales y aceptada por Sam Ferguson, colaborador de Hales. Como reacción a los límites encontrados en el proceso tradicional de revisión de demostraciones basadas en ordenador, Hales ha iniciado un proyecto [38] en cuya introducción se dice:

> The purpose of the *flyspeck* project is to produce a formal proof of the Kepler Conjecture. The name "flyspeck" comes from matching the pattern /f.*p.*k/ against an English dictionary. FPK in turn is an acronym for "The Formal Proof of Kepler". [8]

Una prueba formal, dice Hales, asegura a editores y revisores alguna certeza acerca de que el código se comporta conforme a sus especificaciones, incluso si éste no es revisado en detalle. Para estas demostraciones se usan demostradores automáticos (Isabelle, COQ, HOL, ACL2,...), herramientas cuyo código ha de ser, evidentemente, libre. Y, aunque estas demostraciones no estén exentas de error, cabe decir que tampoco lo está el habitual sistema de revisión de resultados cuando el código no es examinado.

5. Conclusiones

En esta nota hemos pretendido destacar que el software libre en general, y en particular el software libre matemático, está teniendo una importancia cada vez mayor, no solamente desde el punto de vista práctico, sino también desde un punto de vista conceptual. No hemos pretendido dar un listado exhaustivo de todo el software libre matemático disponible, más bien hemos pretendido mostrar unas piezas valiosas de software que pueden ser de gran utilidad para la comunidad matemática. Sólo nos queda animar a los matemáticos a acercarse al software libre, y a utilizarlo, modificarlo, mejorarlo y difundirlo, con toda libertad.

Agradecimientos

Este trabajo ha sido desarrollado con el apoyo del proyecto MTM2008-04699-C03-03/MTM.

[8] El objetivo del proyecto *flyspeck* es producir una prueba formal de la Conjetura de Kepler. El nombre "flyspeck" proviene de la correspondencia del patrón / f. p. * * k / con un diccionario Inglés. FPK, a su vez, es un acrónimo de "La prueba formal de Kepler".

REFERENCIAS

[1] G. D. Allen, J. Herod, M. Holmes, V. Ervin, R. J. Lopez, J. Marlin, D. Meade, D. Sanches, (1999), Strategies and guidelines for using a computer algebra system in the classroom. *International Journal of Engineering Education*, 15(6), 411-416 (1999)

[2] M. Artigue, The integration of symbolic calculators into secondary education: some lessons from didactical engineering. En D. Guin, K. Ruthven, y L. Trouche (Ed.), *The didactical challenge of symbolic calculators - Turning a computational device into a mathematical instrument* (pp. 197-231). New York, NY: Springer Inc. (2005).

[3] C. Beade, C. Martínez, M. Fernández, M. T. Lozano, *La Gaceta de la RSME*, vol. 1, no. 3, pp. 467-488 (1998).

[4] I. Chrysanthou, *The use of ICT in primary mathematics in Cyprus: the case of GeoGebra*, Master's thesis, University of Cambridge, UK (2008).

[5] O. Ciaurri, J.L. Varona. ¿Podemos fiarnos de los cálculos efectuados con ordenador?, *La Gaceta de la RSME*, 9(2), pp. 484-513 (2006).

[6] M. Delgado, J.L. Vicente, I. Luengo, *La Gaceta de la RSME*, vol. 1, no. 1, pp. 113-128 (1998).

[7] European Comission, *Use of Computers and the Internet in Schools in Europe.* 2006.

[8] J. González, J. Seoane, G. Robles, *Introducción al software libre*, UOC (2003).

[9] Haken, Appel, Kock. Every planar map is four-colorable, *Illinois Journal of Mathematics*, 21 (84), pp. 429-567 (1977).

[10] M. Hohenwarter, *Open Source and Online Collaboration: The Case of GeoGebra*, plenary talk, 4th International Workshop on Mathematical and Scientific e-Contents (MSEC 2008), Trondheim, Norway (2008).

[11] ICE, Universidad Complutense de Madrid, *Formación de profesores en educación secundaria. Didáctica de las Matemáticas.* (2008)

[12] D. Joyner, W. Stein: Open Source Mathematical Software, *Notices of the AMS*, p. 1279 (2007)

[13] M. Kendal, K. Stacey, The impact of teacher privileging on learning differentiation with technology. *International Journal of Computers for Mathematical Learning*, 6, 143-165 (2001).

[14] *Ley Orgánica de Educación*, BOE 4 mayo 2006.

[15] J.L. Llorens, Tecnología para el realismo en la enseñanza del Cálculo Integral, *La Gaceta de la RSME*, vol. 5, no. 2, pp. 455-462 (2002).

[16] R. Losada, GeoGebra: la eficiencia de la intuición, *La Gaceta de la RSME*, vol. 10, no. 1, pp. 223-239 (2007).

[17] Y.W.A. Lu, *Linking geometry and algebra: a multiple-case study of upper-secondary mathematics teachers' conceptions and practices of GeoGebra in England and Taiwan*, Master's thesis, University of Cambridge, UK (2008).

[18] C. Moreno, MatLab en el Cálculo Científico, *La Gaceta de la RSME*, vol. 3, no. 2, pp. 351-361 (2000).

[19] I. Moreno, Posibilidades didácticas de la informática en educación. En: A. Monclús (Coord.), Educación y Sistema educativo, pp. 383-397. Universidad Complutense de Madrid (2007).

[20] A. Pérez, El profesorado de matemáticas ante las Tecnologías de la Información y la Comunicación, *La Gaceta de la RSME*, vol. 9, no. 2, pp. 521-544 (2006).

[21] J. Preiner, *Introducing Dynamic Mathematics Software to Mathematics Teachers: the Case of GeoGebra*, Doctoral dissertation in Mathematics Education. Faculty of Natural Sciences, University of Salzburg, Austria (2008).

[22] A. Wiles, Modular elliptic curves and Fermat's last theorem. *Ann. of Math.* (2) 141, no. 3, 443–551 (1995).

[23] `http://www.gnu.org/gnu/manifesto.html`, Acceso 20/03/2009

[24] `http://sci.tech-archive.net/Archive/sci.math.symbolic/2008-01/msg00085.html`, Acceso 10/04/2008

[25] `http://reference.wolfram.com/mathematica/tutorial/WhyYouDoNotUsuallyNeedToKnowAboutInternals.html`, Acceso 10/04/2008

[26] `http://acgeogebra.cat/moodle`, Acceso 20/03/2009

[27] `http://www.bth.se/llab/easa\_2002.nsf`, Acceso 20/03/2009

[28] `http://clic.xtec.net`, Acceso 20/03/2009

[29] `http://www.geogebra.org`, Acceso 20/03/2009

[30] `http://www.geogebra.org/igi`, Acceso 20/03/2009

[31] Grupo de trabalho de Imagem e Conhecimiento, Tecnologia e Metodologia (vídeo en YouTube), `http://www.youtube.com/watch?v=IJY-NIhdw_4`, Acceso 20/03/2009

[32] `http://icme11.org`, Acceso 20/03/2009

[33] `http://www.isftic.mepsyd.es`, Acceso 20/03/2009

[34] `http://www.opensourceworldconference.com`, Acceso 20/03/2009

[35] `http://www.sagemath.org`, Acceso 20/03/2009

[36] `http://www.sagemath.org/links-components.html`, Acceso 20/03/2009

[37] `http://webs.uvigo.es/adg2006/invited.html`, Acceso 18/3/2009

[38] `http://code.google.com/p/flyspeck/`, Acceso 18/03/2009

[39] `http://www.openoffice.org/`, Acceso 20/03/2009

[40] `http://www.maplesoft.com/support/downloads/GMP.html`, Acceso 10/03/2009

[41] `https://magma.maths.usyd.edu.au/magma/export/mpfr_gmp.shtml`, Acceso 10/03/2009

[42] `http://www.mathworks.com/company/aboutus/policies_statements/thirdparty_agreement.pdf`, Acceso 10/03/2009

[43] `http://sagemath.blogspot.com/2008/05/can-there-be-viable-free-open-source.html`, Acceso 10/03/2009

[44] http://www.uam.es/personal_pdi/ciencias/engonz/sagedays16/, Acceso 10/03/2009

MIGUEL Á. ABÁNADES, CENTRO DE ESTUDIOS SUPERIORES FELIPE II, UNIVERSIDAD COMPLUTENSE DE MADRID
Correo electrónico: mabanades@cesfelipesegundo.com

FRANCISCO BOTANA, DPTO DE MATEMÁTICA APLICADA I, E.U.T.I. FORESTAL, UNIVERSIDAD DE VIGO, CAMPUS A XUNQUEIRA, PONTEVEDRA
Correo electrónico: fbotana@uvigo.es

JESÚS ESCRIBANO, DPTO DE SISTEMAS INFOMÁTICOS Y COMPUTACIÓN, FACULTAD DE INFORMÁTICA, UNIVERSIDAD COMPLUTENSE DE MADRID
Correo electrónico: jesus_escribano@mat.ucm.es

LUIS F. TABERA, DEPARTAMENTO DE MATEMÁTICAS, ESTADÍSTICA Y COMPUTACIÓN, FACULTAD DE CIENCIAS, UNIVERSIDAD DE CANTABRIA