EVALUACIÓN DE HERRAMIENTAS DE SOFTWARE LIBRE PARA CÁLCULO NUMÉRICO

C. MEDRANO[1], J.M. VALIENTE[2], I. PLAZA[1] Y P. RAMOS[2]
[1]*EduQTech, Dpto Ingeniería Electrónica y Comunicaciones, EUPT, Universidad de Zaragoza.*
[2]*Dpto Ingeniería Electrónica y Comunicaciones, EUPT, Universidad de Zaragoza.*

En este trabajo se presenta una evaluación de herramientas de software libre para cálculo numérico. El procedimiento seguido se ha dividido en varios pasos. Primero se ha realizado una búsqueda de las herramientas disponibles, seleccionando tres de ellas atendiendo a su difusión y a tener una diversidad en la elección: Octave, Scilab, y Scipy. Posteriormente, se ha definido un modelo de calidad a través de una encuesta para saber la importancia que dan los usuarios a diferentes características, que se han evaluado dando una nota a cada aspecto considerado, una vez analizadas las posibilidades que ofrece cada programa. Finalmente, se ha realizado una comparación con MatLab, la herramienta más conocida, y se han encontrado los puntos fuertes y débiles de cada una de ellas.

1. Introducción

Los programas de software libre están alcanzando una amplia difusión en los últimos años. Estos programas se caracterizan por la difusión libre de su código, que da libertad a los usuarios para redistribuirlo o cambiarlo, por el acceso generalmente libre al programa, y por la contribución de muchas personas al proyecto para su desarrollo. Una discusión de la filosofía de código libre puede encontrarse en [1], donde se destacan aspectos relacionados también con el futuro de estas herramientas en proyectos comerciales y educativos. En el ámbito universitario, el uso de software libre está sin duda ampliamente implantado para asignaturas de corte informático, donde tanto alumnos como profesores están habituados a usar sistemas operativos como Linux, o compiladores como *gcc*, que además destacan por su calidad. Por el contrario, en asignaturas de electrónica o de procesado de señal, es más difícil encontrar casos de uso de software libre. Entre las razones para ello podemos citar:

- Menor conocimiento del software libre por parte de los profesores.
- Herramientas con menor grado de desarrollo, y que además interesan a un menor número de personas, comparado por ejemplo con un compilador de C como *gcc*. Esto hace que cueste más tiempo desarrollar los programas y éstos no han alcanzado la calidad necesaria.

Sin embargo, la introducción de programas de software libre presentaría algunas ventajas, como el acceso libre de los alumnos a la herramienta para su trabajo personal (de gran importancia en enseñanza a distancia), la contribución de una comunidad al desarrollo del programa y la independencia de casas comerciales y de sus políticas de precios por licencia.

En este contexto, los autores se han planteado realizar una revisión y evaluación de distintos tipos de herramientas para su uso en docencia e investigación: lenguajes de descripción hardware, simulación de circuitos digitales y analógicos, diseño de placas de circuito impreso y herramientas de cálculo numérico. En este trabajo presentamos los resultados relacionados con este último grupo, aunque el método utilizado, que se describe en las siguientes secciones, será similar en todos los casos.

2. Elección de las herramientas.

Entre las herramientas de software libre para cálculo numérico podemos encontrar las siguientes: Octave, R-Lab, Scilab, Scipy, la librería gsl en C, etc. Hemos escogido tres de ellas, atendiendo a su difusión y a tener una diversidad en la filosofía que inspira cada programa:

- Octave [2]: quizás el programa más conocido y con una gran compatibilidad con MatLab. Se ha utilizado la versión 2.1.72.
- Scilab [3]: tiene una vocación de integrar en una sola herramienta todos aquellos paquetes que puedan ser de interés. La versión utilizada es la 3.1.1.
- Scipy [4]: un proyecto más joven, que tiene como base el lenguaje de programación Python. Nuestra versión de Scipy es la 0.4.6, apoyándose sobre Numpy 0.9.5 y Python 2.4.2.

Octave es prácticamente un clon de MatLab, por lo que programas escritos en éste último pueden pasarse casi de forma directa a Octave. Octave fue originalmente concebido hacia 1988 para incluirlo en un libro de texto sobre diseño de reactores químicos. El principal desarrollador es John W. Eaton. La orientación original cambió para construir una herramienta más flexible. La primera versión alpha apareció en 1993, y actualmente se encuentra en la versión 2.1.73 (dentro de las versiones denominadas 'testing', que son las que se recomiendan), y se distribuye con licencia GPL (General Public License) [1]. Es una herramienta muy utilizada y sobre la que otros desarrolladores están realizando paquetes adicionales [5].

Scilab es un paquete de software científico para computación numérica que provee un entorno de trabajo computacional abierto y potente para aplicaciones científicas y de ingeniería. Ha sido desarrollado desde 1990 por investigadores pertenecientes al INRIA (Institut National de Recherche en Informatique et Automatique) y al ENPC (École Nationale des Ponts et Chaussées). Desde Mayo de 2003, es mantenido y desarrollado por Scilab Consortium, que engloba a centros de investigación y empresas. Las tres entidades previamente citadas son francesas. La versión actual es la 4.0. Aunque su licencia no es exactamente GPL, las diferencias sólo afectarían a los usos comerciales de versiones de Scilab modificadas por el usuario. La sintaxis de Scilab es también similar a la de MatLab.

Scipy es un software científico que comenzó su desarrollo en 2001. Está basado en el lenguaje de programación Python, que es ya de por sí un lenguaje de comandos interpretado y orientado a objetos. Sus orígenes se remontan al paquete con extensiones numéricas para Python denominado Numeric. Posteriormente apareció Numarray, con la intención de construir un paquete más flexible y de limpiar el código, aunque resultó ser más lento para cálculos matriciales en pocas dimensiones. En el año 2005, el principal impulsor de Scipy, Travis Oliphant, reunificó ambos en un único paquete que integrase sus respectivas ventajas, y que se denomina Numpy, pudiéndose considerar como el núcleo de Scipy [4]. Scipy en sí mismo, se concibe actualmente como una extensión de las funcionalidades de Numpy. La versión actual es la 0.4.8 para Scipy, y la 0.9.6 de Numpy, aunque se avanza rápidamente a obtener la versión 1.0. Scipy está patrocinado por una empresa, Enthought (Scientific Computing Solutions) y se ofrece con licencia BSD (Berkeley Software Distribution, una licencia libre menos restrictiva que la GPL para el uso del software en productos propietarios). Es el programa más diferente de MatLab de los tres que hemos considerado.

3. Evaluación de las herramientas

Para evaluar las herramientas, se ha definido un modelo de calidad basado en la norma ISO/IEC 9126 [6]. Esta norma da pautas generales sobre qué aspectos deben ser tenidos en cuenta a la hora de evaluar un programa: funcionalidad, confiabilidad, eficiencia, facilidad de uso, mantenimiento y portabilidad. Evidentemente, estas características deben concretarse para cada caso específico.

ENCUESTA DE EVALUACIÓN DE HERRAMIENTAS DE CÁLCULO NUMÉRICO

Se trata de una encuesta para ayudar a conocer cuáles son las características más relevantes para los usuarios de programas de cálculo numérico (como MatLab por ejemplo). Para ello se pide que se evalúe de 1 (poco importante) a 5 (muy importante) cada uno de los aspectos que a continuación se detallan.

1. Funcionalidad.

1.1. Funcionalidades básicas.
Matrices (operaciones con matrices, vectores y valores propios), funciones (ceros, mínimos), polinomios, procesado de señal básico.

1.2. Funcionalidades avanzadas.
Funciones más específicas (equivalente a los ToolBox de MatLab), como por ejemplo redes neuronales, procesado de imagen y de señal.

1.3. Gráficos e imágenes.
Posibilidad de mostrar gráficos de funciones e imágenes.

1.4. Potencia del lenguaje de programación.
Tipos de datos, estructuras de control, orientación a objetos, modularidad.

2. Confiabilidad.

2.1. Control de la precisión.
Control de la precisión de los resultados.

2.2. Fiabilidad.
Existencia de 'bugs', 'cuelgues' del programa.

3. Eficiencia.

3.1. Rapidez.
Velocidad del procesamiento de datos.

4. Facilidad de uso.

4.1. Información.
Manuales, libros, ayuda dentro del programa, información en internet.

4.2. Facilidad de manejo.
La facilidad para introducirse en el programa sin un conocimiento a priori, existencia de depuradores.

5. Mantenimiento.

5.1. Licencia y facilidad de obtención.
Tipo de licencia, acceso al programa, precio, versiones de estudiante.

5.2. Desarrollo y madurez del proyecto.
Aquí englobamos aspectos como existencia de versiones nuevas (non-stable), la frecuencia con la que salen nuevas versiones, compatibilidad entre ellas.

6. Portabilidad.

6.1. Instalación.
Facilidad en la instalación, necesidad de saberse 'trucos'.

6.2. Compatibilidad con otros programas y formatos estándar.
Posibilidad de adaptarse o comunicarse con otros programas, uso de formatos estándares.

6.3. Integración de otros lenguajes.
Integración de código en C, Fortran u otros lenguajes.

7. Otros aspectos.
Añade otros aspectos que consideres convenientes y la importancia que le concedes.

Figura 1. Encuesta entregada a usuarios de programas de cálculo numérico.

3.1. Encuesta a los usuarios

De la reflexión que hemos realizado sobre aquellos aspectos que pueden ser relevantes en un programa para cálculo numérico, ha surgido una encuesta que se ha entregado a usuarios de este tipo de programas (todos ellos usan MatLab en realidad). En esta encuesta se pregunta sobre la importancia que se concede a cada uno de los aspectos, valorándolos entre 1 (poco importante) y 5 (muy importante). Resaltamos que esta encuesta no es sobre una herramienta concreta, sino sobre lo que los usuarios demandan en general.

En la figura 1 se puede observar la encuesta detallada, junto con los comentarios que se adjuntaban para orientar a los usuarios sobre el sentido de las preguntas. Se han obtenido un total de 14 respuestas de personal de la Universidad de Zaragoza, 5 de profesores y 9 de alumnos y becarios de investigación. El valor medio obtenido en las respuestas aparece en la tabla 1.

Podemos ver que los aspectos que más importan a nuestros usuarios son las funciones básicas, la fiabilidad del programa y los gráficos e imágenes; por contra, la licencia, el desarrollo y la madurez, y la integración de otros lenguajes, son los aspectos menos valorados. La baja valoración que se le da al aspecto de la licencia merece un comentario. Probablemente, en el caso de profesores, se debe a que la Universidad de Zaragoza posee una licencia de universidad para MatLab, por lo que no se deben preocupar directamente de ello. En otros casos, creemos que la baja valoración indica la utilización de copias piratas del programa, una práctica habitual pero no recomendada.

Aspecto	Valoración media (entre 1 y 5)
Funcionalidades básicas	4.58
Funcionalidades avanzadas	3.85
Gráficos e imágenes	4.35
Potencia del lenguaje de programación	3.19
Control de la precisión	3.00
Fiabilidad	4.00
Rapidez	3.71
Información	3.98
Facilidad de manejo	3.08
Licencia y facilidad de obtención	2.46
Desarrollo y madurez del proyecto	2.50
Instalación	2.98
Compatibilidad con otro programas	3.10
Integración de otros lenguajes	2.50

Tabla 1. Valoración media de los aspectos consultados en la encuesta de la figura 1.

3.2. Evaluación comparativa

Tras realizar la encuesta los usuarios, se ha procedido a realizar una evaluación comparativa entre las herramientas. Se ha calificado de 1 a 3 cada uno de los aspectos en cada una de ellas, incluyendo MatLab en la comparación (versión 6.0 R12). Para dar esta nota se ha tenido en cuenta la experiencia adquirida en el uso de estas herramientas en aspectos básicos, la información que se encuentra sobre ella para aspectos más especializados, y se ha realizado una puesta en común para todos los programas. En los siguientes párrafos detallamos esta evaluación.

3.2.1. Funcionalidad: funcionalidades básicas

Todas las herramientas presentan las funcionalidades básicas que permiten trabajar con matrices: operaciones entre matrices, valores propios etc. Asimismo se puede trabajar con funciones y encontrar ceros y mínimos. Los polinomios también existen en todos los programas, así como funciones elementales de procesado de señal (convolución, correlación, transformada de Fourier).

Se puntúa por igual a todas ellas.

3.2.2. Funcionalidad: funcionalidades avanzadas

En Octave es necesario descargar paquetes adicionales para obtenerlas, y entre librería avanzadas y paquetes podemos encontrar 62, un número que se considera elevado. Scilab posee hasta 87 paquetes, incluyendo contribuciones de usuarios. Scipy junto con Numpy tiene 24 paquetes. En concreto hemos buscado si existen ampliaciones referentes a tres aspectos, de especial interés para nosotros: procesado de señal, redes neuronales y procesado de imágenes. Octave posee una librería de procesado de señal muy potente, posibilidad de enlazar con FANN 1.2 [7] (Fast Artifical Neural Network library), y un paquete de procesado de imágenes. Scilab también posee un paquete de procesado de señal, de procesado de imágenes [8] y de redes neuronales. Scipy tiene un módulo de procesado de señal, y no incluye directamente procesado de imágenes o redes neuronales. Sin embargo, al basarse en Python, tiene acceso a todas las funcionalidades para las que exista o bien paquetes o bien enlazadores desde Python. Así podemos encontrar la librería PIL (Python Imaging Library) [9] y un enlace a la librería FANN mencionada anteriormente. Se echa en falta una información integrada de los módulos accesibles en Python-Scipy. Por este motivo pondremos a Scipy por debajo de los demás.

Recalcamos que aquí evaluamos sólo la existencia de paquetes y su accesibilidad. La evaluación en detalle de cada módulo avanzado constituye un trabajo futuro. Podemos avanzar que la primera impresión es que MatLab posee el mayor número de funciones, siendo Scilab y Scipy los que tienen menos desarrollo.

3.2.3 Funcionalidad: Gráficos e Imágenes

En Octave podemos dibujar gráficos en 2D y 3D, usando el conocido programa Gnuplot [10]. Las imágenes se pueden editar con la aplicación ImageMagick [11]. Con Scilab, también podemos editar gráficos, así como imágenes y videos con el ToolBoox denominado SIVP (Scilab Image and Video Processing)[12]. Scipy ha decidido recientemente no incluir directamente los gráficos, y dejar que esta funcionalidad se apoye en otras herramientas, recomendándose Matplotlib [13]. Existe también la posibilidad de enlazar desde Python con Gnuplot [14], o bien usar Gnuplot directamente si se crean los ficheros adecuados. Sin embargo nuestra experiencia es peor con Scipy, bien por la falta de información, por la confusión ante el gran número de posibilidades, o por no haber sido capaces de hacer funcionar todas las herramientas en profundidad. Respecto a la edición de imágenes, ésta se puede realizar con la ayuda de la librería PIL.

Así, podemos concluir que la capacidad de trabajar con gráficos e imágenes en realidad se apoya casi siempre en otros programas libres, con lo cual se aprovecha un trabajo ya realizado.

En este apartado colocaremos a Scipy por debajo del resto de herramientas.

3.2.4. Potencia del lenguaje de programación

Todos ellos poseen un gran número de estructuras de control de flujo de programa, y de tipos de datos. Las diferencias comienzan cuando se demandan características más especializadas. No hemos encontrado información sobre programación orientada a objetos en Octave y en Scilab. En MatLab, también es un aspecto poco tratado, donde la mejor explicación encontrada se encuentra en [15]. De todas maneras, la definición del objeto y su forma de uso es un tanto peculiar, muy diferente de C++, por ejemplo: la organización de directorios necesaria para el objeto, con el camino definido en la aplicación; y la llamada a un método donde el primer argumento debe ser el objeto. En muchas listas de correo se considera que MatLab no es muy elegante. En este aspecto, Scipy es superior a los demás, puesto que se basa en Python, y por tanto recoge todas las características de programación orientada a objetos de un buen lenguaje de programación. Otros aspecto donde MatLab se queda algo atrás, es en

la posibilidad de definir procesos concurrentes, algo que sí tenemos en Octave, en Scilab con máquinas virtuales de procesamiento en paralelo, y Python con 'Multi threading'.

En este apartado hemos decidido poner a Scipy por encima de los demás. En [16] tenemos ejemplos de la capacidad de Python para, en unas pocas líneas, crear una página web con los resultados de un cálculo y enviar un correo para advertir del fin de una simulación.

3.2.5. Control de la precisión

Aunque en todos ellos existen varios tipos de datos con distinta precisión, a la hora de la verdad las operaciones con un tipo dado están muy limitadas, y muchos cálculos acaban haciéndose con la máxima precisión. Tan sólo hemos encontrado que en Scipy el poner un tipo de datos u otro tiene efectos sobre la rapidez de ciertas operaciones, por lo que le colocamos por encima de los demás.

3.2.6. Fiabilidad

En este punto nos hemos fijado en los fallos ('bugs') y en la estabilidad del propio programa durante su ejecución, es decir en que no haya 'cuelgues'. Tanto Octave como MatLab apenas tienen fallos. En SciLab, no existe mucha información en las listas de correo, por lo que no podemos concretar nada. Scipy es un proyecto más joven, por lo que aparecen más fallos. No obstante hay que decir que éstos no son críticos, sino pequeños problemas de interfaz que los desarrolladores solucionan en unas pocas líneas de código en Python. Conviene recordar que, como en muchos de estos programas, los cálculos básicos usan rutinas en Fortran o en C ampliamente probadas desde hace años. Respecto a la estabilidad de los programas, Scilab también ha resultado cerrarse con la ejecución de algunas demos y con algunos cálculos, sin que hayamos podido determinar la causa exacta. No obstante, hemos de precisar que estos fallos ocurren en contadas ocasiones, de forma que no es un hecho que impida trabajar con la herramienta.

En este apartado colocaremos a MatLab y Octave por encima, y a Scilab por debajo.

3.2.7. Rapidez

Hemos encontrado una comparación entre MatLab, Octave y Scilab en [17], en la que se evalúan operaciones con matrices y aspectos de programación. MatLab es el más rápido de los tres. En la lista de correo de Scipy, podemos encontrar algunos test sobe operaciones básicas de álgebra lineal [18] y comentarios de los usuarios, desprendiéndose que Scipy es comparable a MatLab. Nosotros hemos adaptado los programas de test de [17] para Scipy, obteniendo una conclusión similar, utilizando el mismo sistema operativo (WindowsXP). Scipy es incluso algo superior en muchos test, pero en unos pocos muy concretos (encontrar autovalores por ejemplo), es bastante peor. No obstante, al ir progresando en nuestro conocimiento de Scipy, hemos cambiado el código de los programas y obtenido mejores resultados, ya que, como ocurre con MatLab, es importante evitar el uso de bucles, y utilizar funciones que trabajen directamente con matrices y vectores. Por otro lado, Scipy es superior también al poder hacer algunas operaciones con enteros de longitud variable o con coma flotante de simple precisión, reduciendo los tiempos. Además, los argumentos a funciones se pasan por referencia por lo que se pueden modificar dentro de la función, mientras que MatLab, si modifica un argumento, necesita realizar una copia de él. Por contra, para obtener la máxima rapidez con Scipy debemos tener previamente instaladas y adaptadas las librerías BLAS (Basic Linear Algebra Subprograms) o LAPACK (Linear Algebra PAckage) [19], [20], [21]. En caso contrario, no obtendremos tan buenos resultados.

En este apartado, colocaremos a MatLab y a Scipy por delante de Octave y Scilab.

3.2.8. Información

MatLab es de sobra conocido. Octave tiene buenos manuales de introducción y existen numerosas páginas web. Echamos en falta ejemplos aclaratorios en la ayuda del programa. En Scilab ocurre lo

contrario en cierta manera: la ayuda del programa es buena y fácil de usar al ser gráfica, pero no se encuentran muchos manuales ni páginas web en castellano. Scipy es el peor de todos, ya que la ayuda del programa no es muy explícita y falta un tutorial completo. Las listas de correo es una valiosa fuente de información para programas libres, y esto es aún más cierto para Scipy. Como una pequeña ventaja sobre MatLab, a las listas de correo de los programas libres suelen responder los propios desarrolladores del programa.

En este apartado puntuamos a MatLab como el mejor, y a Scipy como el peor.

3.2.9. Facilidad de manejo

Octave es muy fácil de usar si se conoce previamente MatLab. Echamos en falta un depurador gráfico. En Scilab, la línea de comandos es algo más peculiar, pero es un aspecto menor. El depurador no está implementado de forma completa todavía. Respecto a Scipy, su uso es más diferente de MatLab. Existe un entorno gráfico de edición y depuración, denominado IDLE (Integrated DeveLopment Environment) [22], pero es necesario instalarlo como paquete aparte y la depuración está incompleta todavía, debiendo recurrir a la depuración desde una línea de comandos.

En este apartado colocaremos a MatLab por encima de los demás.

3.2.10. Licencia y facilidad de obtención

En este aspecto destacan las tres herramientas de software libre, mientras que MatLab no tiene, hasta donde nosotros sabemos, ni siquiera una versión de estudiante.

3.2.11. Desarrollo y madurez del proyecto

En este apartado tenemos en cuenta si es un proyecto consolidado, pero que a la vez se renueve. Todas las herramientas están activas. No hay cambios dramáticos de una versión a otra. En las tres herramientas de software libre, podemos encontrar versiones no estables (no probadas todavía), pero con los últimos cambios que los desarrolladores han incluido. Esto permite a los usuarios acceder a las funcionalidades más nuevas. Quizás, Scipy entrega versiones con demasiada frecuencia.

Pondremos a Scipy ligeramente por debajo.

3.2.12. Instalación

Las versiones de Windows se instalan sin problemas, aunque en algunos casos sea necesario incluir programas adicionales. En el caso de utilizar Linux, la instalación se puede realizar a partir de paquetes, de la compilación del código fuente o de binarios. Con las versiones basadas en la distribución Debian hemos encontrado paquetes adecuados. Sin embargo, han surgido algunos problemas con los paquetes de la distribución Fedora para Scilab y Scipy. La instalación a partir de la compilación de las fuentes tampoco es siempre elemental, es necesario conocer el sistema operativo, y en el caso de SciLab ha habido que recurrir a versiones binarias. Hay que tener cuidado con muchos detalles, como el compilador de Fortran que se utiliza, las opciones de compilación, etc. Realmente este aspecto puede ser molesto para personas no iniciadas en Linux, e incluso para gente más experimentada que debe recurrir a las listas de correo para encontrar ayuda. MatLab en Linux no ha podido ser evaluado porque ... ¡no tenemos la licencia para Linux! Este hecho lo consideramos como un aspecto negativo.

En este apartado colocaremos a Octave el primero y a Scilab el último.

3.2.13. Compatibilidad con otros programas y uso de formatos estándar

En este apartado valoramos el uso de formatos estándar (por ejemplo de ficheros de imágenes), así como la facilidad para importar o exportar ficheros de otras aplicaciones. Octave puede trabajar con distintos tipos de formato de imágenes y audio, así como guardar datos en formato de MatLab. Lo

mismo puede decirse de Scilab. Scipy puede aprovecharse de otros paquetes para trabajar con formatos estándares de imágenes o de audio. Sin embargo, el propio código del programa es más difícil de portar a MatLab, aunque sí permite leer y guardar ficheros MAT.

En este apartado colocamos a Scipy por debajo de los otros.

3.2.14. Integración de otros lenguajes de programación

Octave es capaz de integrar C++ y Fortran (en realidad tras una traducción a C++). Scilab puede integrar también ambos lenguajes. MatLab añade a la lista anterior Java. Scipy también puede integrar C++ y Fortran, y existen una implementación de Python que busca la integración con el lenguaje Java [23]. Scipy destaca por tener varias opciones a la hora de realizar la integración [16]: f2py y PyFort para Fortran, SWIG, SIP, Pyrex, boost y weave para C++. Una de ellas, weave, permite incluso tener código en C directamente dentro del código de Python. Python puede ser además embebido en aplicaciones de C++.

En este apartado destacamos a Scipy por la variedad de posibilidades.

3.3. Resumen de la evaluación comparativa

Un resumen de la evaluación comparativa se muestra en la tabla 2.

Tras realizar esta comparativa, hemos dado una nota final a cada programa, pesando los valores de la tabla 2 con el valor relativo que los usuarios dan a cada característica. Esta valoración es sólo una orientación y aparece en la tabla 3. Cada usuario debe decidir qué características son más importantes para él y fijarse en ellas para tomar una decisión.

Característica	MatLab	Octave	Scilab	Scipy
Funcionalidades básicas	3	3	3	3
Funcionalidades avanzadas	3	3	3	2
Gráficos e imágenes	3	3	3	2
Potencia del lenguaje de programación	2	2	2	3
Control de la precisión	1	1	1	2
Fiabilidad	3	3	1	2
Rapidez	3	2	2	3
Información	3	2	2	1
Facilidad de manejo	3	2	2	2
Licencia y facilidad de obtención	1	3	3	3
Desarrollo y madurez	3	3	3	2
Instalación	2	3	1	2
Compatibilidad con otros programas	3	3	3	2
Integración de otros lenguajes de programación	2	2	2	3

Tabla 2. Nota comparativa entre las herramientas.

MatLab	Octave	Scilab	Scipy
2.59	2.52	2.23	2.26

Tabla 3. Nota promedio (sobre 3) de las herramientas, ponderada por la opinión de los usuarios.

En la tabla 4 hemos presentado los que, a modo de valoración más personal, son los puntos fuertes y débiles de cada programa, tras la experiencia adquirida en su uso. Aunque en este trabajo hemos pensado en estas herramientas como lenguajes de programación en modo texto, en dicha tabla hemos

resaltado que Scilab tiene también una herramienta gráfica para construir y simular sistemas dinámicos [24], constituyendo una alternativa al Simulink de MatLab.

Al mismo tiempo que hemos aprendido el uso de estas herramientas, se han desarrollado guías de introducción accesibles a través de la red [25]. Estas guías permiten obtener una visión de las operaciones más importantes, con ejemplos concretos para su uso. Incluyen la creación y operaciones con matrices y vectores, funciones de librería, programación de funciones de usuario, gráficos y otros. Cualquier comentario sobre estas guías será bienvenido por parte de los autores.

Programa	Puntos fuertes	Puntos débiles
MatLab	Gran número de funciones y ToolBox Amplia difusión, diseño apoyado en interfaces gráficas Información sobre el programa	Licencia
Octave	Gran cantidad de funciones y paquetes Alta compatibilidad con MatLab Integración de y en otros programas Licencia	Rapidez
Scilab	Integración de todos los paquetes en un sólo programa Modelado gráfico de sistemas dinámicos Ayuda del programa Licencia	Falta de información en la red en castellano Mala impresión por los 'cuelgues' de las demos Rapidez
Scipy	Lenguaje de programación Facilidad para integrar otros programas, lenguajes Licencia	Información sobre el programa Proyecto en desarrollo

Tabla 4. Puntos fuertes y débiles de los programas analizados.

4. Conclusiones y trabajo futuro

En este trabajo hemos realizado un repaso a las características de varias herramientas de software libre para cálculo numérico. Los programas seleccionados han sido instalados y se han realizado operaciones básicas con ellos. También se ha buscado información sobre características avanzadas. Tras estos pasos se ha procedido a la evaluación de los programas. Hemos realizado una encuesta entre los usuarios para saber lo que demandan de este tipo de programas y la importancia que conceden a diferentes aspectos, definiendo un modelo de calidad. Posteriormente hemos realizado una evaluación comparativa, a partir de la experiencia adquirida.

Hemos elaborado también manuales de introducción de todas las herramientas, que son accesibles a través de la red. Las referencias citadas en este artículo sirven también de base para encontrar información sobre los programas.

Tras la evaluación realizada, podemos concluir que MatLab es el programa más completo y con una mayor información. La tradición y los años en el mercado sin duda influyen en ello. El usuario debe decidir si estas características compensan los problemas para adquirirlo y proporcionarlo a los estudiantes, su punto más débil. Desde otra perspectiva, podemos plantearnos la pregunta de si es posible realizar las prácticas básicas de asignaturas de matemáticas y procesado de señal con otros programas. La respuesta es que sí. Con cualquiera de las tres herramientas seleccionadas es posible. Octave permite pasar un programa de MatLab de forma casi directa, y puede ser una opción para aquellas personas que no quieran dedicar tiempo a aprender otro lenguaje muy distinto a MatLab. SciLab es un entorno completo, con comandos bastante parecidos a MatLab, que ha creado sus propias interfaces gráficas, su propio lenguaje etc. Scipy, aunque algo menos desarrolado, tiene la ventaja de tener detrás un buen lenguaje de programación, Python, y ser un proyecto muy activo. Se podría pensar incluso en una colaboración entre asignaturas de diferentes áreas, en las que Python serviría como base tanto para asignaturas de programación orientada a objetos como para asignaturas que necesiten cálculo numérico.

Como trabajo futuro, se plantea:

- Realizar un estudio de cada herramienta centrado en aspectos especializados (procesado de imágenes, redes neuronales, procesado de señal avanzado, etc). De este modo podríamos profundizar en cada uno de ellos, y repetir el mismo proceso descrito en nuestro trabajo: aprendizaje de las herramientas, comparación y elaboración de guías de introducción.
- Extender este tipo de evaluaciones a otro tipo de herramientas: compiladores para lenguajes de descripción hardware, simuladores y diseño de placas de circuito impreso. Este trabajo ya se ha iniciado.

Finalmente, nos gustaría invitar a todas aquellas personas que estén interesadas en hacer estas evaluaciones o aportar su experiencia, para que se sumen al trabajo y poder desarrollar una visión global de las herramientas y un conjunto de tutoriales que faciliten su uso.

Referencias

[1] Varios autores, *Open Sources: Voices from the Open Source Revolution*, O'reilly, 1999.
[2] http://www.octave.org/, visitada por última vez en mayo de 2006.
[3] http://www.scilab.org/, visitada por última vez en mayo de 2006.
[4] http://www.scipy.org/, visitada por última vez en mayo de 2006.
[5] http://octave.sourceforge.net/, última visita en mayo de 2006.
[6] ISO/IEC 9126-1:2001, Software engineering - Product quality - Part 1: Quality model.
[7] http://leenissen.dk/fann/, última visita en mayo de 2006.
[8] http://siptoolbox.sourceforge.net/, última visita en mayo de 2006.
[9] http://www.pythonware.com/products/pil/, última visita en mayo de 2006.
[10] http://www.gnuplot.info/, última visita en mayo de 2006.
[11] http://www.imagemagick.org/script/index.php, última visita en mayo de 2006.
[12] http://sivp.sourceforge.net/, última visita en mayo de 2006.
[13] http://matplotlib.sourceforge.net/, última visita en mayo de 2006.
[14] http://gnuplot-py.sourceforge.net/, última visita en mayo de 2006.
[15] http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_prog/f6-4018.html, última visita en mayo de 2006.
[16] E. Jones, T. Oliphant, *Introduction to Scientific Computing with Python*, Parts 1 and 2, Presentación accesible en https://www.nanohub.org/resources/?id=99, última visita en mayo de 2006.
[17] http://www.sciviews.org/benchmark/, última visita en mayo de 2006.
[18] Paulo Jose da Silva e Silva, *Numpy x Matlab: some synthetic benchmarks*, mensaje a la lista de correo de numpy (http://www.scipy.org/Mailing_Lists), 18-1-2006.
[19] BLAS, Basic Linear Algebra Subprograms, http://www.netlib.org/blas/, última visita en mayo de 2006.
[20] LAPACK, Linear Algebra PACKage, http://www.netlib.org/lapack/, última visita en mayo de 2006.
[21] ATLAS, Automatically Tuned Linear Algebra Software, http://www.netlib.org/atlas/, última visita en mayo de 2006.
[22] Página web de Python, http://www.python.org/idle/, última visita en mayo de 2006.
[23] Página web de Jython, http://www.jython.org/, última visita en mayo de 2006.
[24] SCICOS, Scilab Connected Object Simulator, http://scilabsoft.inria.fr/doc/scicos/
[25] 'Docencia' en la web personal de Carlos Medrano, http://eupt2.unizar.es/cmedrano/docencia.html, última visita en mayo de 2006.