UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
DEPARTAMENTO DE SISTEMA OPERATIVO Y DESARROLLO DE TECNOLOGÍAS LIBRES
Facultad 10

MECANISMO DE INTEGRACIÓN ENTRE ASISTENTES MATEMÁTICOS DE NATURALEZA SIMBÓLICA Y NUMÉRICA: UNA PROPUESTA DESDE EL SOFTWARE LIBRE

Trabajo presentado en opción al grado científico de Máster en Informática Aplicada

Autor:
Ing. Leansy Alfonso Pérez

Tutor:
MSc. Alexeis Companioni Guerra

Ciudad de La Habana
Febrero de 2010

"Nosotros necesitamos una manera sustancialmente distinta de pensar para que la humanidad pueda sobrevivir".

Albert Einstein

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firman la presente a los _____ días del mes de _____ de 2009.

Ing. Leansy Alfonso Pérez	Msc. Alexeis Companioni Guerra

Resumen

Los asistentes matemáticos son utilizados principalmente en centros de investigaciones científicas y múltiples carreras pertenecientes a centros de enseñanza técnica y superior. Matlab es un asistente numérico de licencia privativa y uno de los más utilizado a nivel mundial y particularmente en nuestro país. Debido a sus privativos precios así como a las innumerables restricciones a que son sometidos los usuarios cubanos de esta herramienta, que van desde las negativas de licencias hasta los cierres de accesos a su sitio oficial, se hace necesario la búsqueda de alternativas capaces de suplir la ausencia de este producto en nuestras entidades docentes y de investigación. Atendiendo esta necesidad, en el proyecto productivo EIDMAT de la facultad 10 se desarrolla una interfaz gráfica para el asistente matemático distribuido bajo licencia GPL GNU Octave que recrea los aspectos fundamentales del ambiente de trabajo de Matlab; no obstante, persisten algunas brechas por cubrir en términos del potencial de cálculo simbólico que brinda este asistente y al que los usuarios de Matlab están habituados.

Siguiendo esta línea se propone un mecanismo que permite la ejecución de comandos de GNU Maxima a través de la plataforma EIDMAT cuando se trabaja en el ambiente de GNU Octave y la recuperación de los resultados en variables propias del ambiente de éste último, facilitándose así la expansión y potenciación de sus capacidades de cálculo simbólico. Por otra parte, y en el empeño de continuar integrando los universos de las matemáticas simbólicas y numéricas, se propone un mecanismo que permite la ejecución de comandos de GNU Octave a través de la plataforma EIDMAT mientras se trabaja desde el ambiente de GNU Maxima. Al igual que en el caso anterior, los resultados arrojados durante las operaciones solicitadas son retornados e incluidos en variables propias del ambiente simbólico. De manera adicional, se propone una interfaz gráfica para la plataforma EIDMAT que permite utilizar el asistente GNU Maxima mediante sus mecanismos propios de interacción con el usuario. Esta interfaz propone el uso de menús y funcionalidades encontradas hoy en el programa wxMaxima, además de brindar un área de trabajo que emula el comportamiento de GNU Maxima cuando se ejecuta como programa interactivo en modo texto.

ÍNDICE

ÍNDICE DE FIGURAS

Introducción

En la actualidad los asistentes matemáticos son utilizados ampliamente en centros de investigaciones científicas y múltiples carreras universitarias. Matlab, de licencia privativa, es uno de los asistentes numéricos más utilizado a nivel mundial [1] por el número de prestaciones tan elevado que posee y el nivel de integración, extensibilidad y facilidad de uso que incorpora (ver figura 1). En contraste con estos atractivos el mismo posee precios muy elevados [2] que lo hacen completamente inasequibles para universidades y centros de investigación o producción de países en vía de desarrollo como el nuestro por lo que pensar en la posibilidad de su sustitución por un similar libre de costos se convierte más que en un romántico deseo de estar a la moda, en una necesidad imperiosa en aras de un desarrollo profesional sostenible en nuestras instituciones. Obedeciendo a esta ley natural se decide migrar a software libre todas aquellas herramientas de cálculo simbólico y numérico de uso frecuente en nuestros centros de enseñanza e investigación.

El proyecto productivo EIDMAT [3][4] de la facultad 10 propone una alternativa viable para la sustitución del asistente matemático Matlab por un homólogo distribuido bajo licencia GNU/GPL: el GNU Octave (ver figura 2 (a)). Éste último es un asistente numérico con similares funcionalidades que Matlab y compatible en un alto porcentaje [5]; pero dada la poca interactividad que brinda y la carencia de un entorno visual, muchas veces es renegado a la última opción por profesionales y estudiantes. GNU Octave tiene definido un conjunto de funciones que permiten resolver diversos problemas algebraicos lineales y no lineales; no obstante, puede ser extendido mediante la utilización del mismo lenguaje de Octave para definir nuevas funciones o implementando módulos en C++, C u otros lenguajes

de programación los cuales son cargados dinámicamente.

EIDMAT en su versión inicial se presenta como una interfaz gráfica para GNU Octave con un importante número de similitudes al software Matlab; no obstante, las riquezas emanadas de la práctica diaria han conducido al pensamiento y concepción de una herramienta mucho más poderosa a manos de usuarios ávidos de conocimientos y empeños de cálculo.

Matlab a pesar de ser un asistente matemático esencialmente numérico, permite realizar cálculos simbólicas a través de la caja de herramientas (Symbolic Math). Esta extensión le permite incorporar cientos de funciones simbólicas que permiten realizar tareas como: diferenciación, integración, operaciones algebraicas lineales, simplificación, transformaciones y resolver sistemas de ecuaciones entre otras más [6].

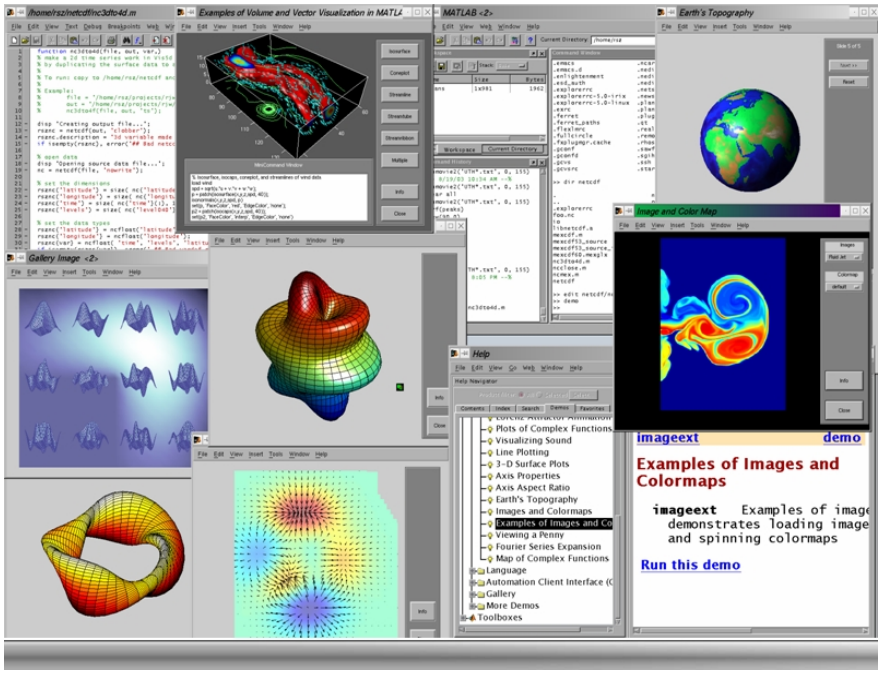

Figura 1: Interfaz gráfica de Matlab.

Por su parte GNU Octave para realizar cálculos simbólicos debe tener incorporado el módulo complementario (symbolic) y éste módulo depende a su vez de las bibliotecas de

funciones GiNaC y CLN. Con symbolic se incorporan funciones geométricas, se pueden plotear funciones simbólicas y se pueden resolver sistemas de ecuaciones simbólicas; hasta el momento las funciones implementadas por symbolic son: *vpa*, *sym*, *is_vpa*, *is_sym*, *is_ex*, *to_double*, *to_char*,*digits*, *Cos*, *Sin*, *Tan*, *aCos*, *aSin*, *aTan*, *Sinh*, *Tanh*, *aCosh*, *aSinh*, *aTanh*, *Exp*, *Log*, *subs*, *differentiate*, *expand*, *collect*, *coeff*, *lcoeff*, *tcoeff*, *degree*, *ldegree*, *quotient*, *remainder*, *premainder*, *Pi* y *splot* [7]. A pesar de las facilidades que nos brinda éste módulo, no es posible tener en GNU Octave la mayoría de las funcionalidades disponibles en la caja de herramientas Symbolic Math de Matlab y mucho menos las encontradas en los sistemas de cálculos simbólicos como GNU Maxima (ver figura 2 (b)) lo cual se convierte de antemano en una limitante para la sustitución del Matlab y para el ansiado desarrollo profesional.

GNU Máxima es distribuido bajo licencia GNU/GPL, es un asistente matemático diseñado para la realización de cálculos simbólicos y desarrollado en Common Lisp. Permite la manipulación de polinomios, matrices, funciones racionales, integrales, entre otros y está basado en el legendario asistente Macsyma que fue desarrollado por el MIT en los años '70. Al igual que otros asistentes, proporciona un conjunto de funciones básicas y puede ser extendido mediante la creación nuevos ficheros con las funciones deseadas.

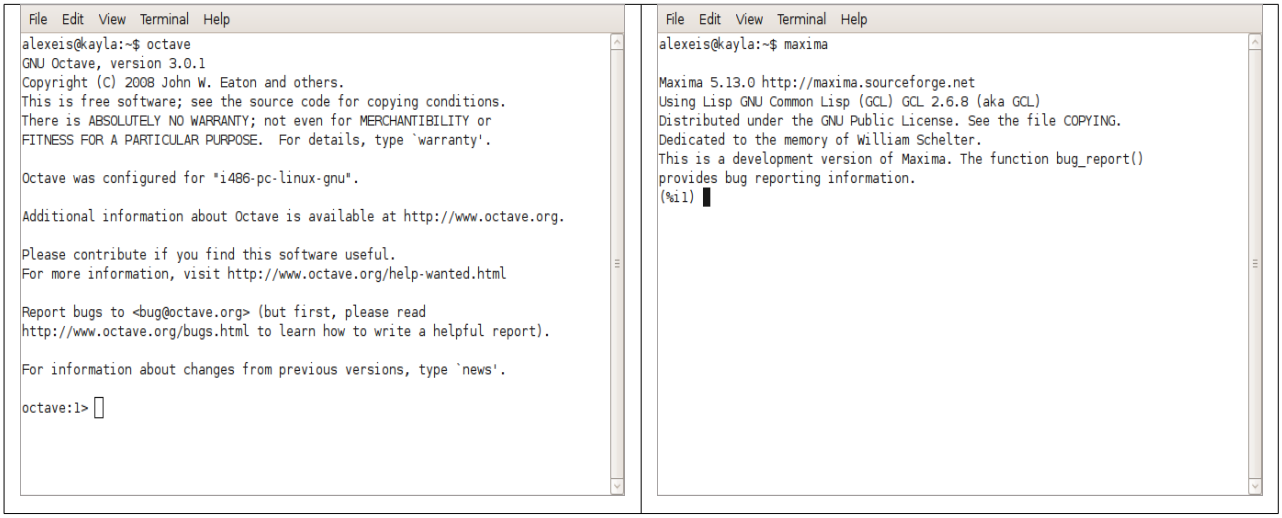

Figura 2: Interfaz de trabajo de los asistentes matemáticos GNU Octave (a) y GNU Maxima (b)

Analizando las características y comportamientos de los asistentes matemáticos de naturaleza numérica y simbólica hoy disponibles bajo licencia GNU/GPL y de modo particular los ya tratados, no sería sorpresa encontrar intentos de unificar estos universos matemáticos mediante la interconexión de estas herramientas. El programa Euler Math Toolbox[8] es un asistente numérico que ha devenido pionero en la realización de cálculos simbólicos a través de una integración con GNU Maxima. En su concepción incorpora un mecanismo interno de conexión pero presenta como principales inconvenientes el hecho que su sintaxis no es compatible con Matlab, y a pesar de ser software libre está implementado para correr únicamente sobre sistemas operativos Windows. Para poder ejecutarlo en sistemas GNU Linux se hace necesario emularlos con programas como el Wine u otros de similares propósitos. A pesar de que Euler Math Toolbox no cumple con las expectativas de una herramienta moderna, extensible, de elevada potencia y sobre todo, compatible con los líderes internacionales del tema, nos muestra un camino posible para realizar cálculos simbólicos desde asistentes numéricos. Otra propuesta para ejecutar comandos de GNU Maxima desde plataformas numéricas la encontramos en QtOctave [9], dicho programa consiste en otra interfaz gráfica de usuario para GNU Octave, la conexión es lograda a través de la función interna **"system"** que permite realizar llamadas a comandos del sistema operativo. Por esta vía es llamado el GNU Maxima y el resultado es recuperado como una cadena de caracteres que puede ser manipulada posteriormente en GNU Octave[1].

Siguiendo estos pasos y ante la determinante necesidad de extender las capacidades de cálculo de GNU Octave, EIDMAT redirecciona su concepción inicial de una simple interfaz gráfica de usuario a una plataforma de trabajo que toma como base un asistente matemático numérico y se propone extender sus potencialidades a través de mecanismos de conexión con otros asistentes especializados en el trabajo en universos matemáticos de naturaleza disímil.

[1]Realmente la manipulación posterior es muy limitada

Problema científico

Inexistencia de un mecanismo de integración entre los asistentes matemáticos GNU Octave y GNU Maxima que posibilite el efectivo trasiego de información entre los ambientes de la matemática simbólica y numérica.

Hipótesis

Contar con un mecanismo de integración entre los asistentes matemáticos GNU Octave y GNU Maxima que posibilite el efectivo trasiego de información entre los ambientes de la matemática simbólica y numérica proveería al sector de la docencia y la investigación de una herramienta libre de prestaciones comparables a las poseídas por los líderes internacionales de similar naturaleza.

Objetivo general

Diseñar e implementar un mecanismo que permita la integración entre los asistentes matemáticos GNU Octave y GNU Maxima posibilitando el efectivo trasiego de información entre los ambientes de la matemática simbólica y numérica.

Objetivo específicos

- Desarrollar un módulo de comunicación para el asistente GNU Octave que permita aprovechar las potencialidades de GNU Maxima desde su propio ambiente.
- Desarrollar un módulo de comunicación para el asistente GNU Maxima que permita aprovechar las potencialidades de GNU Octave desde su propio ambiente.
- Proponer una interfaz de usuario para el trabajo con GNU Maxima desde el ambiente de EIDMAT que contemple la ejecución de instrucciones y recopilación de resultados dentro de una misma región de trabajo.

- Proponer los mecanismos de integración de la interfaz de usuario para el trabajo con GNU Maxima con el núcleo de EIDMAT.

Resultados esperados

- Obtención de un mecanismo que permita la integración entre los asistentes matemáticos GNU Octave y GNU Maxima posibilitando el efectivo trasiego de información entre los ambientes de la matemática simbólica y numérica.
- Propuesta de una interfaz para el trabajo con GNU Maxima desde el ambiente de EIDMAT.
- Propuesta de un mecanismo de integración de la interfaz de usuario para el trabajo con GNU Maxima con el núcleo de EIDMAT.

Estructura del documento

El presente documento se estructura en introducción, tres capítulos, conclusiones y un grupo de anexos, donde se expone y da cumplimiento de forma progresiva y elocuente a la totalidad de los objetivos propuestos:

- Capítulo 1. Fundamentación teórica: Presenta cómo está implementado EIDMAT. Se analizan las distintas opciones para la incorporación de la comunicación entre los programas GNU Octave y GNU Maxima y cómo insertar esta nueva funcionalidad respetando la arquitectura ya existe en EIDMAT. También se muestran detalles de la sintaxis de ambos asistentes matemáticos y cómo extender las funcionalidades de GNU Octave y GNU Maxima.
- Capítulo 2. Solución propuesta: Se describe la solución propuesta para ejecutar cálculos simbólicos en GNU Octave con la asistencia de GNU Maxima. Se explica además cómo se adicionó a EIDMAT la opción de ambiente de trabajo para cálculos simbólicos, convirtiéndose éste también en una interfaz gráfica de usuario para GNU Maxima. En

la solución propuesta se incluye un mecanismo de llamadas a GNU Octave desde GNU Maxima para realizar cálculos numéricos con su asistencia.

- Capítulo 3. Pruebas del sistema: Como su nombre lo indica en este apartado se realizan un conjunto de pruebas al sistema a modo de verificar sus funcionalidades. De modo particular se presentan y resuelven tres situaciones cuyas soluciones precisan del libre trasiego de información entre los universos de las matemáticas numéricas y simbólicas, exponiéndose así de forma palpable el cumplimiento del objetivo general del trabajo.

Capítulo 1

Fundamentación teórica

En éste capítulo se presenta cómo está implementado EIDMAT. Se analizan además las distintas opciones para la incorporación de la comunicación entre los programas GNU Octave y GNU Maxima y cómo insertar esta nueva funcionalidad respetando la arquitectura ya existe en EIDMAT. Por último se describe cómo extender las funcionalidades de GNU Octave y GNU Maxima.

La implementación de EIDMAT se realiza empleando el lenguaje de programación de alto nivel Python y la biblioteca de clases para construir interfaz gráfica de usuario GTK[1] en su versión 2.0. Python es un lenguaje orientado a objetos diseñado por Guido van Rossum a principios de los años '90 que a pesar de ser un lenguaje script se caracteriza por ser muy rápido [10][11], multiplataforma y contiene un conjunto muy amplio de bibliotecas de clases y funciones que permiten un desarrollo ágil de aplicaciones para fines diversos. De todas las bibliotecas utilizadas se destacan PyGTK[2][12] que se compone de un conjunto de módulos Python que proporcionan una interfaz para GTK y la VTE[3] que se describe como una implementación de un control para una terminal, que permite controlar la ejecución de comandos en esta así como los resultados obtenidos.

[1] Del inglés Graphic ToolKit
[2] Binding de Python para GTK
[3] Del inglés Virtual Terminal Emulator

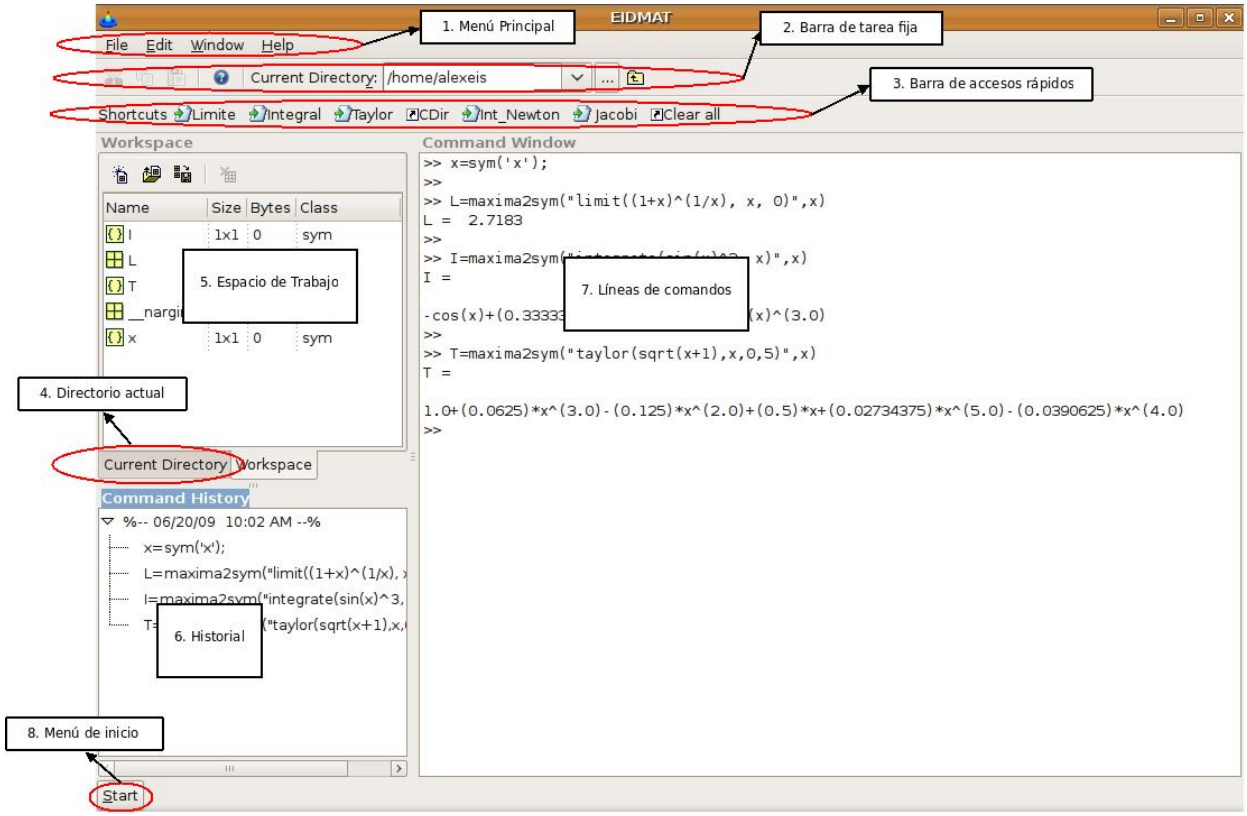

Figura 1.1: Interfaz gráfica de EIDMAT para GNU Octave.

1.1. EIDMAT como interfaz gráfica de usuario de GNU Octave

La interfaz gráfica de usuario que presenta EIDMAT para GNU Octave se divide en las siguientes partes (ver figura 1.1).

1. Menú principal: presenta cuatro categorías, a través de las cuales se pueden acceder a todas las opciones que brinda EIDMAT

2. Barra de tarea fija: presenta accesos rápidos a las acciones “Cortar”, “Copiar” y “Pegar” textos en el área de trabajo que se esta ubicado. Un vínculo para llamar la ayuda de GNU Octave y por último un formulario que permite cambiar el directorio de trabajo

3. Barra de accesos rápidos: da la posibilidad de crear accesos rápidos personalizados por el usuario
4. Directorio actual: permite cambiar el directorio de trabajo
5. Espacio de trabajo: muestra el nombre, tamaño y tipo de las variables definidas
6. Historial: contiene un listado de los comandos ejecutados
7. Línea de comandos: permite la entrada de comandos y muestra los resultados obtenidos
8. Menú de inicio: vínculos a la ayuda y a los accesos rápidos creados por el usuario

En la plataforma EIDMAT se cuenta con un mecanismo para la conexión con GNU Octave, que permite ejecutar en éste último las órdenes recibidas. La comunicación entre la interfaz gráfica de usuario de EIDMAT y GNU Octave se realiza de la forma siguiente: con la ayuda de un control GTK disponible en la ventana principal de la aplicación y una instancia de la clase VTE donde se mantiene ejecutando GNU Octave, se capturan los comandos proporcionados a través de la interfaz gráfica de usuario y se ejecutan en el asistente matemático. Finalmente el resultado devuelto por GNU Octave se muestra en la ventana principal. El diagrama UML de comunicación correspondiente se muestra en la figura 1.2.

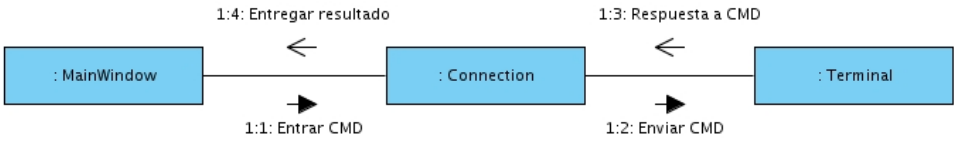

Figura 1.2: Diagrama UML de comunicación entre la interfaz gráfica de usuario de EIDMAT y GNU Octave.

1.2. Ejecución de GNU Octave y GNU Maxima

Como muchas otras aplicaciones sobre SO[4] GNU/Linux, tanto GNU Octave como GNU Maxima pueden ser ejecutados en distintos modos:

1. **Programa interactivo modo texto**: En esta forma de ejecución la aplicación posee el control total de la entrada y salida de datos, el usuario interactúa directamente con la interfaz del programa a partir de la sintaxis definida por el mismo.

2. **Comando en lote**: Por esta vía la aplicación se ejecuta como un comando del sistema operativo donde las opciones de ejecución se pasan a través de parámetros. Una vez terminado el procesamiento se devuelve el resultado como cadena de caracteres y el control pasa nuevamente al sistema operativo.
   Existen varias formas de lograr ejecutar GNU Octave como comando en lote, una forma de conseguirlo es pasándole la opción **"--eval"** y como parámetro la expresión a evaluar, por ejemplo, si en la línea de comandos tecleamos lo siguiente:

   octave -q --eval x=4*4

   Con esta sentencia se le ordena a GNU Octave de que resuelva $(4 * 4)$ y lo asigne a la variable "x". La opción **"-q"** es utilizada para indicar la no inclusión del mensaje de inicio de GNU Octave en la respuesta impresa en la salida estándar.
   Utilizando las tuberías del sistema operativo **"|"** se consigue el mismo resultado. El comando se construye de la forma siguiente:

   echo 'x=4*4' | octave -q

   En ambos casos el resultado es el mismo: "$x = 16$".
   Este comportamiento se puede lograr de forma similar empleando GNU Maxima. Si ejecutamos en un terminal el comando:

[4] Abreviatura correspondiente a: Sistema Operativo

maxima --batch-string='string(x=y+y);' --very-quiet

se obtiene como respuesta:

$$string(x = y + y)$$
$$x = 2 * y$$

La opción **"--batch-string"** que aparece en la sentencia indica a GNU Maxima que tiene que ejecutarse en el modo de comando en lote para resolver la expresión que aparece después del signo de asignación, en el ejemplo se calcula "y+y" y se asigna el resultado a "x". Se utiliza la función interna de GNU Maxima **"string"** para transformar la expresión a cadena de caracteres para que la respuesta sea también una cadena de caracteres, de esa forma la sintaxis de la sentencia en la respuesta obtenida es idéntica a como debe escribir las expresiones un usuario para interactuar con GNU Maxima, de esta forma se logra que se pueda utilizar posteriormente dicha respuesta en la construcción de un nuevo comando; de no utilizarse la función **"string"** la respuesta se obtiene como "x = 2 y" en lugar de "x = 2*y". La segunda opción **"--very-quiet"** permite eliminar tanto el mensaje de inicio como el prompt de dicho sistema para que solo se obtenga como salida el comando ejecutado y a continuación el resultado devuelto.

Otra forma de lograr ejecutar GNU Maxima como comando en lote es haciendo uso de las tuberías "|"; por ejemplo, si en la línea de comandos se teclea:

echo 'string(x=y+y);' | maxima --very-quiet

se obtiene esta vez solo el resultado: "x=2*y".

En esta segunda variante no se incluye la orden recibida en la respuesta, devolviendo únicamente el resultado.

3. **Comunicación por Sockets**: En este tipo de comunicación las aplicaciones se ejecutan como programas de fondo utilizando un puerto específico para la comunicación; por su parte la entrada y salida de datos se realiza escribiendo y leyendo por el puerto

definido.

Para obtener comunicación por sockets en GNU Octave se debe instalar el módulo ***octave-sockets*** el cual brinda la totalidad de las funciones para el trabajo con sockets. Por su parte para la comunicación por sockets con GNU Maxima es necesario pasar la opción **"-s"** y como parámetro el puerto por el cual se desea establecer la comunicación. En la línea de comandos se pondría lo siguiente:

maxima -s 50000

y la comunicación con GNU Maxima la lograríamos por el puerto 50000 en éste caso, suponiendo que exista un servidor sockets creado por ese puerto.

El Anexo 1 muestra fragmentos de código en Python desarrollados para ejecutar GNU Maxima a través de la comunicación por sockets. En el mismo se puede encontrar una clase para definir un cliente de GNU Maxima que pueda comunicarse por sockets, una clase para crear un servidor sockets y una clase para manejar la interfaz de usuario y la comunicación por sockets.

1.3. Opciones de comunicación entre GNU Octave y GNU Maxima

Para establecer la comunicación entre GNU Octave y GNU Maxima se manejan tres opciones, las cuales se pueden aplicar en cualquiera de los dos sentidos, desde GNU Octave hacia GNU Maxima y desde GNU Maxima hacia GNU Octave:

1. **Trabajo con funciones del sistema**: tanto GNU Octave como GNU Maxima incorporan una función **"system"**, a la cual se le puede pasar un comando del sistema para la ejecución y obtención de la respuesta en formato de cadena de texto. Esta función y la posibilidad de ejecución de estos dos programas como comandos en lotes permiten que desde uno se pueda ejecutar el otro y obtener una cadena de texto como respuesta. De esta forma no se afecta el flujo de datos en EIDMAT, la comunicación se

realiza de forma directa entre los dos programas y son independientes de EIDMAT. La principal desventaja que presenta esta vía es que se puede tornar muy difícil procesar la cadena de texto obtenida con el lenguaje propio de GNU Octave o GNU Maxima. Al obtenerse la cadena respuesta de la acción desencadenada por un usuario se hace necesario transformar la sintaxis de un lenguaje a otro; para ello se deben identificar las variables que se devuelven, el tipo de dato de cada una y generar la misma expresión que se recibió en un lenguaje con una sintaxis válida para el otro.
Para ejecutar comandos del sistema desde GNU Octave, haciendo uso de función interna **"system"**, se define en el lenguaje Octave la sintaxis siguiente:

[status, result] = system(command)

donde:

- *status*: toma valor 0 si el comando se pudo ejecutar sin problemas y otro número en caso contrario
- *result*: toma el resultado de la ejecución del comando, el tipo de éste resultado siempre llega a GNU Octave como cadena de caracteres
- *command*: una cadena de caracteres con una secuencia válida de comandos del sistema operativo

Poder obtener el resultado en una variable de GNU Octave, permite realizar un procesamiento de la misma y lograr transformaciones hasta llevarla a una expresión con sintaxis válida de su propio lenguaje.

La sintaxis de la función interna **"system"** en GNU Maxima es la siguiente:

status = system(command)

donde:

- *status*: toma valor 0 si el comando se pudo ejecutar sin problemas y otro número en caso contrario
- *command*: una cadena de caracteres con una secuencia válida de comandos del sistema operativo

A diferencia de su similar en GNU Octave, la función **"system"** de GNU Maxima no almacena el resultado en una variable, simplemente imprime la cadena de caracteres del resultado por la salida estándar. Esta característica hace que la recuperación del resultado asignado en una variable del sistema sea un proceso complicado con el lenguaje nativo.

2. **Trabajo con tuberías**: consiste en la realización de llamadas al sistema operativo desde la aplicación principal para ejecutar comandos en el mismo. En esta variante el diagrama UML de comunicación entre la interfaz gráfica de usuario de la plataforma EIDMAT y GNU Octave presentado en la figura 1.2 se mantiene inalterable.
   La incorporación de un nuevo procesamiento de comandos enviados desde un control GTK presente en la ventana principal de la aplicación al VTE se realiza mediante la clase **"Connection"**. En ella se analiza el comando de entrada en busca de ordenes para GNU Maxima, de encontrar alguna se extrae y se realizan llamadas al sistema operativo haciendo uso de las tuberías proporcionadas por Python al mismo donde se ejecuta GNU Maxima como comandos en lote pasándole como parámetro el comando a procesar; el resultado devuelto es transformado a una sintaxis válida de GNU Octave que es enviada a la VTE. El diagrama de actividad correspondiente se muestra en la figura 1.3.

   El código en Python para ejecutar un comando de GNU Máxima haciendo uso de las tuberías al sistema operativo se reduce básicamente a la sentencia siguiente:

   **result = Popen(["maxima", "--very-quite", "--batch-string", cmd], **
   stdout=PIPE).communicate()[0]

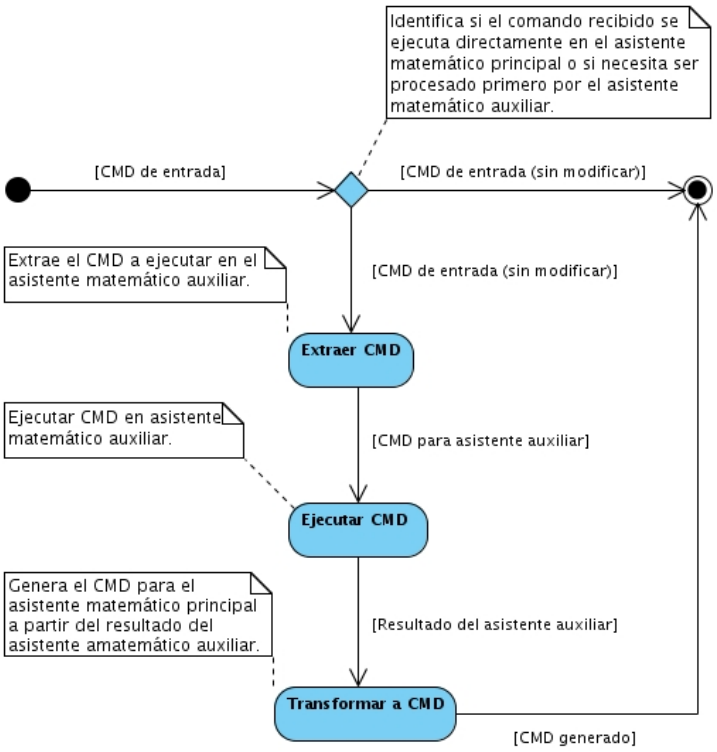

Figura 1.3: Diagrama de actividad de la clase 'Connection' para realizar llamadas a GNU Maxima desde EIDMAT antes de ejecutar GNU Octave.

el comando a ejecutar se pasa mediante la variable **"cmd"** y la respuesta obtenida se almacena en **"result"**[5].

3. **Trabajo con comunicación por sockets**: al igual que la opción de trabajo con tuberías, el diagrama UML de comunicación entre la interfaz gráfica de usuario de la plataforma EIDMAT y GNU Octave mostrado en la figura 1.2 no sufre alteración.

   De manera similar a como se procede en el trabajo con tuberías, se analiza el comando de entrada en busca de ordenes para GNU Maxima, de encontrarse alguna se extrae y se establece comunicación con GNU Maxima por un puerto específico por el cual se encuentra escuchando las peticiones. Con posterioridad se escriben por dicha conexión

[5]Éste mecanismo explicado para ejecutar comandos de GNU Maxima sobre GNU Octave, también es válido en sentido inverso.

los comandos obtenidos y se leen las respuestas a los mismos; el resultado devuelto es transformado a una sintaxis válida de GNU Octave que es enviado a la VTE[6]. El diagrama de actividad correspondiente se muestra en la figura 1.3.

1.4. Observaciones sobre GNU Octave

En GNU Octave existen diferencias en la sintaxis lineal que se utiliza para la definición de las sentencias de entradas y la representación de las mismas cuando se muestran por la salida estándar. Algunas de las diferencias que se pueden encontrar se muestran en los ejemplos siguientes:

- Para crear una matriz se encierran entre corchetes los valores que la forman, para definir las filas se utiliza el punto y coma ";" y para diferenciar las columnas se usa la coma "," o el espacio " ". La respuesta obtenida muestra cada fila en una línea independiente y los valores separados por espacios (ver figura 1.4)

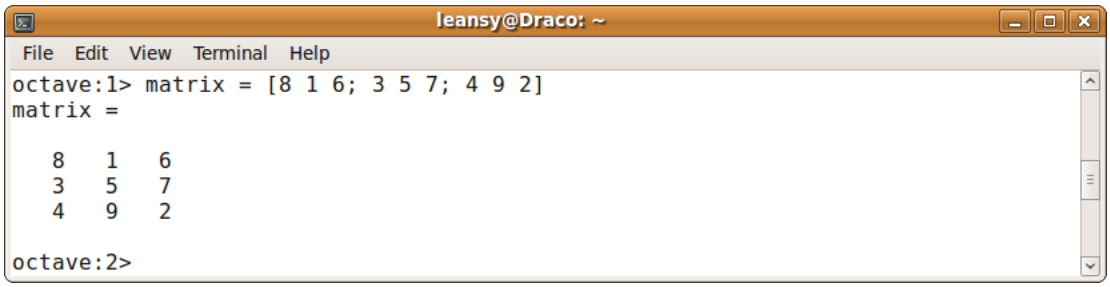

Figura 1.4: Sentencia para crear una matriz en GNU Octave y su representación.

- Cuando una sentencia de GNU Octave devuelve más de un valor cada uno de ellos se pueden recuperar utilizando un vector con una variable por cada valor que es devuelto, de no utilizar un vector solo se recupera el primero de todos los valores y si el vector tiene menos variables que valores devuelva la función solo se recuperan los valores de

[6]Éste mecanismo explicado para ejecutar comandos de GNU Maxima sobre GNU Octave, se puede hacer en el sentido inverso de la misma forma.

los primeros valores hasta asignar todas las variables. El vector que recibe el resultado se define con las variables entre corchetes separadas por una coma “,“ o el espacio “ “ (ver figura 1.5)

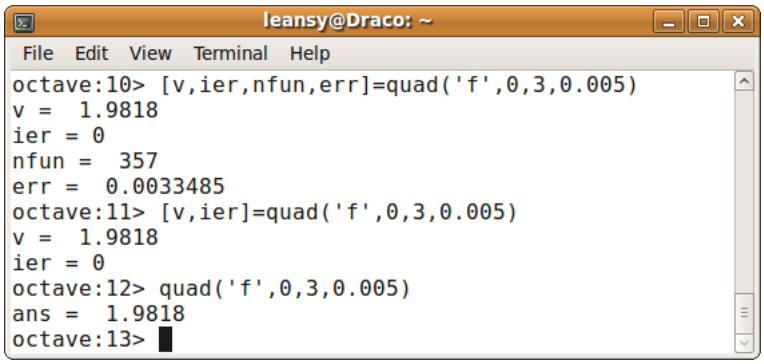

Figura 1.5: Llamada a una función en GNU Octave que devuelve cuatro valores.

- En el trabajo con expresiones simbólicas se escriben los nombres de algunas funciones con una letra en mayúscula como se muestran en la lista siguiente: Cos, Sin, Tan, aCos, aSin, aTan, Sinh, Tanh, aCosh, aSinh, aTanh, Exp, Log [7]. Cuando se recupera la respuesta todas los nombres de las funciones aparecen en minúscula (ver figura 1.6)

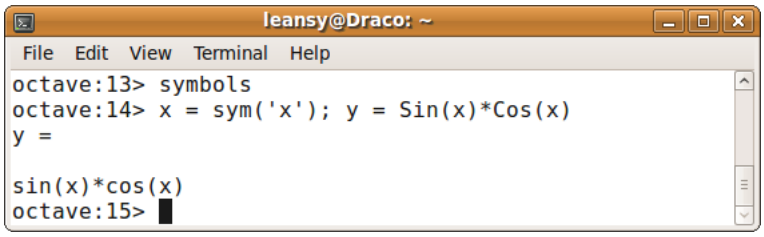

Figura 1.6: Uso del módulo **“symbolic“** de GNU Octave para cálculo simbólico donde se evidencia la diferencia en la nomenclatura del nombre de las funciones en la sentencia de entrada y la respuesta.

1.4.1. Extender GNU Octave

Para extender GNU Octave utilizando su propio lenguaje de programación se deben crear nuevas funciones, las mismas se deben crear en un fichero con extensión ".m" el cual puede ser un fichero *script* o un fichero de función. Las principales diferencias entre estos dos tipos de ficheros son:

1. En los ficheros *script* se pueden definir varias funciones a la vez y en los ficheros de función solo una función es visible desde GNU Octave, la misma debe nombrarse con el nombre del fichero que la contiene y además ser la primera que se defina dentro de dicho fichero

2. El alcance de las definiciones realizadas dentro de un fichero *script* es global, su comportamiento es el mismo a si se hubiera definido directamente en la línea de comandos de GNU Octave; mientras que lo que se define dentro de un fichero función tiene un alcance local, solo es visible dentro de ese fichero

Cuando se ordena ejecutar una función, GNU Octave primero busca en su tabla de símbolos, si la función no está incluida entonces busca en cada uno de los directorios definidos en la variable "path" por un fichero que se llame exactamente igual al nombre de la función con la extensión ".m". Si se desea incluir un nuevo directorio de búsqueda en la variable "path", se debe utilizar el comando **addpath**. Si la función que se quiere ejecutar está en un fichero *script*, entonces en lugar de poner el nombre de la función para que se realice la búsqueda por los directorios correspondientes, se pone el nombre del fichero *script* que contiene la definición de la función deseada, se cargan todas las funciones definidas en el fichero y después puede realizarse la llamada directamente a la función.

1.5. Observaciones sobre GNU Maxima

En GNU Maxima también existen diferencias en la sintaxis lineal que se utiliza para la definición de las sentencias de entradas y la representación de las mismas cuando se muestran

por la salida estándar. Algunas de las diferencias que se pueden encontrar se muestran en los ejemplos siguientes:

- Cuando se escribe una sentencia es necesario utilizar los operadores aritméticos para conformarla. La respuesta obtenida es una expresión donde desaparecen los signos de los monomios. De ser necesario utilizar la respuesta en la construcción de una nueva sentencia manipulando la respuesta como cadena de caracteres es necesario utilizar la función interna **"string"** para que la respuesta se recupere con la misma sintaxis lineal en que fue escrita (ver figura 1.7)

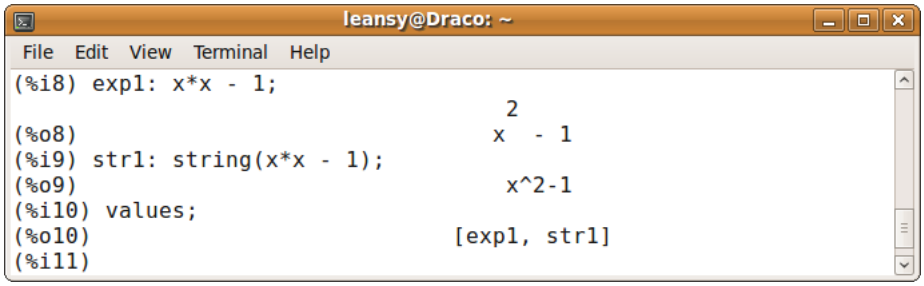

Figura 1.7: Diferencia en GNU Maxima de la sintaxis de la respuesta cuando el tipo de datos es una expresión y cuando es una cadena de caracteres.

- Para crear una matriz en GNU Maxima se utiliza la función interna **"matrix"** y se le pasa como parámetro las filas separadas por coma "," donde cada fila encierra entre corchetes los valores separados por comas. La respuesta obtenida muestra cada fila en una línea independiente encerrando entre corchetes los valores separados por espacios. Entre una fila de contenido y otra aparece una fila vacía (ver figura 1.8)

1.5.1. Extender GNU Maxima

GNU Maxima está escrito en Common Lisp, es posible extender éste asistente matemático programando en lenguaje Lisp y/o el propio lenguaje de GNU Maxima.
Las nuevas funciones se escriben en ficheros, los cuales no necesitan estar en un directorio

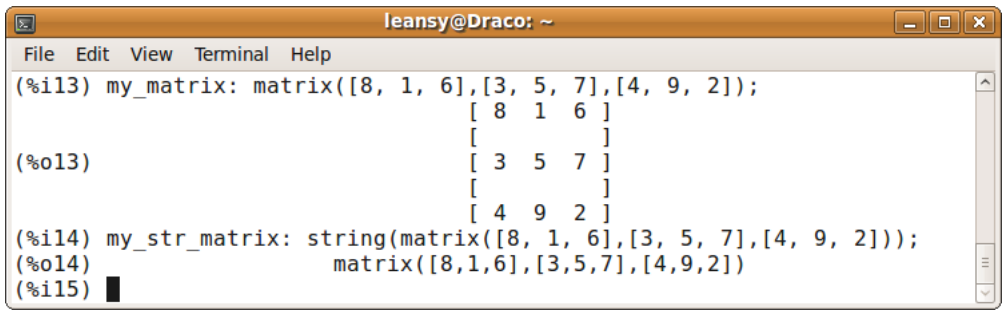

Figura 1.8: Sentencia para crear una matriz en GNU Maxima y su representación.

específico para ser reconocidos por GNU Maxima, existen funciones internas para cargar estos ficheros pasándole el camino a los mismos como parámetros, una vez incorporados estos ficheros, todas las funciones definidas en ellos pasan a formar parte del asistente matemático.

De las funciones internas disponibles para cargar los ficheros la más genérica es **"load"**, que sirve tanto para ficheros escritos en Lisp como los escritos en el lenguaje de GNU Maxima.

1.6. Conclusiones

A modo de conclusiones parciales de este primer capítulo se presentan los aspectos siguientes:

- De las tres vías analizadas para establecer la comunicación entre los asistentes matemáticos GNU Octave y GNU Maxima:

  1. Trabajo con la función interna **"system"**.
  2. Trabajo con tuberías al sistema operativo.
  3. Trabajo con servidores sockets.

  La primera opción debe ser la primera candidata a considerar en la búsqueda de la solución debido a las ventajas que presenta sobre las otras dos: no altera el flujo datos

de EIDMAT y es independiente de esta plataforma (perfectamente puede utilizarse ejecutando GNU Octave o GNU Maxima como programas interactivos)

- Las llamadas a GNU Maxima desde GNU Octave se deben realizar directamente utilizando la función interna **"system"**, porque el resultado devuelto por dicha función puede ser almacenado en una variable de GNU Octave de forma inmediata

- Las llamadas a GNU Octave desde GNU Maxima no se deben realizar directamente utilizando la función interna **"system"**. Se recomienda el empleo de una de las otras dos variantes debido a la dificultad que representa asignar el resultado obtenido por la función **"system"** a una variable de GNU Maxima

- La sintaxis de las sentencias de entrada de GNU Octave y GNU Maxima se diferencian en muchos aspectos, por lo que se hace necesario una traducción de la sentencia obtenida en un asistente matemático para ser utilizada en el otro

- Cuando se ejecute GNU Maxima como comando en lote, la sentencia que se le envíe debe colocarse como parámetro de su función interna **"string"** para obtener el resultado con el tipo cadena de caracteres, y asegurar así una sintaxis lineal en la respuesta, que se corresponda con la forma que se escribe la orden por un usuario cuando interactúa con el asistente matemático

Capítulo 2

Solución propuesta

En éste capítulo se describe la solución propuesta para ejecutar cálculos simbólicos en GNU Octave con la asistencia de GNU Maxima. Se explica además como se adicionó a EIDMAT la opción de ambiente de trabajo para cálculos simbólicos, convirtiéndose éste en una interfaz gráfica para GNU Maxima; la solución propuesta incluye un mecanismo de llamadas a GNU Octave desde GNU Maxima para realizar cálculos numéricos con su asistencia.

2.1. Comunicación desde GNU Octave a GNU Maxima

La solución propuesta para realizar llamadas a GNU Maxima desde GNU Octave, hace uso de la opción **"Trabajo con funciones del sistema"** vista en el capítulo anterior. En esta dirección, se extendieron las funcionalidades de GNU Octave creando un fichero tipo *script* y varios ficheros de tipo función, donde de modo particular se creó el fichero script **_maxima_utils.m** dentro del cual se implementaron varias funciones complementarias que permiten escribir el código una sola vez y utilizarlo varias veces.

Funciones recogidas en el fichero **_maxima_utils.m**:

- **_maximaFormatCommand**: Toma como argumento un comando de GNU Maxima y construye una cadena formateada con la sintaxis para ejecución en lote de GNU

Maxima:

echo 'string(command);' | maxima --very-quiet

donde "command" constituye el comando pasado como parámetro a la función.

- **_maximaFormatOutput**: Toma una cadena como argumento y devuelve la misma cadena, pero eliminando los espacios en blancos y tabuladores de la misma
- **_maxima2str**: Toma como argumentos comandos de GNU Maxima y devuelve una cadena de caracteres con el resultado calculado por GNU Maxima
- **_maxima2num**: Toma como argumento una cadena de caracteres y la convierte a número si es posible realizar la conversión o devuelve un error en caso contrario
- **_maximaMapFunctions**: Devuelve una matriz de dos columnas, en la primera columna se almacena el nombre de una función definida por el módulo **symbolic** y en la segunda columna de la misma fila se tiene el nombre de la misma función pero con la nomenclatura que define GNU Maxima; por ejemplo, para la función seno se tendría **"Sin"** y **"sin"** respectivamente
- **_maximaCaptitalizeSymFunc**: Toma como argumento una cadena de caracteres que representa una expresión simbólica con nomenclatura de GNU Máxima y devuelve la misma expresión simbólica con nomenclatura válida para el módulo symbolic de GNU Octave
- **_maximaReplaceUnaryMinusOperator**: Toma como argumento una cadena de caracteres que representa una expresión simbólica y si aparece el operador unario $(-)$ como primer caracter de la cadena lo remplaza por "$(-1)*$" como vía para que el módulo **symbolic** de GNU Octave funcione correctamente a partir de los datos obtenidos de GNU Maxima

Es necesario destacar que las funciones **_maximaReplaceUnaryMinusOperator**, **_maximaMapFunctions** y **_maximaCaptitalizeSymFunc** fueron necesarias para lograr convertir correctamente una expresión simbólica generada por Maxima a la sintaxis definida por el módulo symbolic de GNU Octave. Como valor agregado se incorpora la función cotangente a las expresiones del módulo symbolic, el mismo no cuenta con esta función, pero a partir de la función **_maximaMapFunctions** es posible mapear a la función (**cot**) que devuelve Máxima con (**1/Tan**) que sí es reconocible por el módulo symbolic de GNU Octave.

De manera adicional al fichero script **_maxima_utils.m** y como parte de este trabajo fueron creados cuatro ficheros de tipo función que dependen del primero. Estos ficheros de funciones definen la totalidad de las funciones que deben ser utilizadas directamente por los usuarios de GNU Octave a modo de alejar a estos de la complejidad de la programación de los atajos de la comunicación entre los asistentes. En estos ficheros se incluye además una ayuda de la función que se implementa la cual se encuentra en el formato requerido por GNU Octave para que la misma pueda ser encuestada con el comando **"help"**.

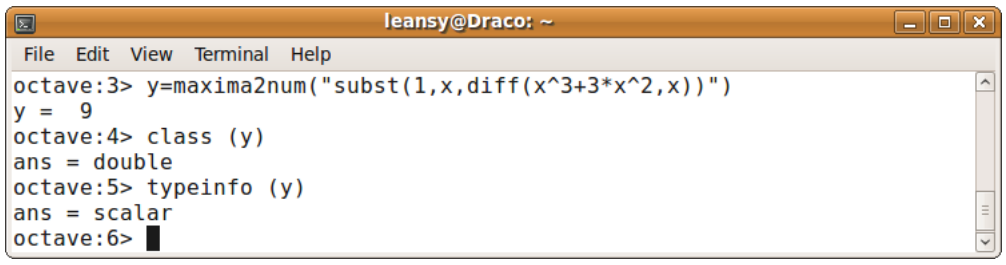

Figura 2.1: Ejecución en GNU Octave del comando **maxima2num**.

Ficheros de funciones creados:

1. **maxima2num.m**

   Ejecuta un comando en GNU Maxima y devuelve un valor numérico que es capturado

en GNU Octave con el tipo **scalar** o clase **double** que representan lo mismo (ver figura 2.1)

2. **maxima2str.m**

   Ejecuta un comando en GNU Maxima que es capturado en GNU Octave con el tipo **string** o clase **char** que representan lo mismo (ver figura 2.2).

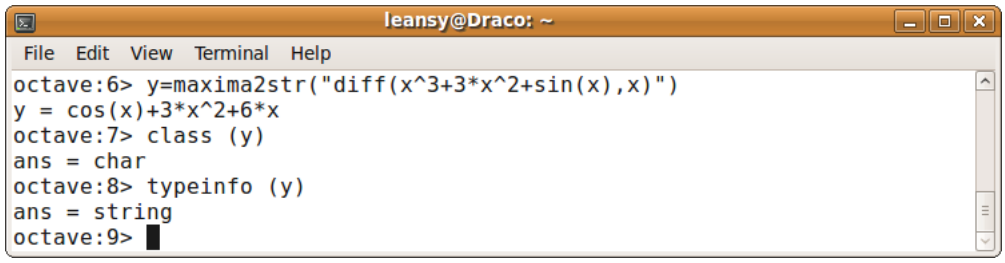

Figura 2.2: Ejecución en GNU Octave del comando **maxima2str**.

3. **maxima2sym.m**

   Ejecuta un comando en GNU Maxima y devuelve una expresión simbólica que es capturado en GNU Octave con el tipo **ex** o clase **sym** que representan lo mismo. Existe la excepción cuando se le pasa un solo argumento a la función; en éste caso devuelve un valor numérico de modo similar a la función **maxima2num** (ver figura 2.3).

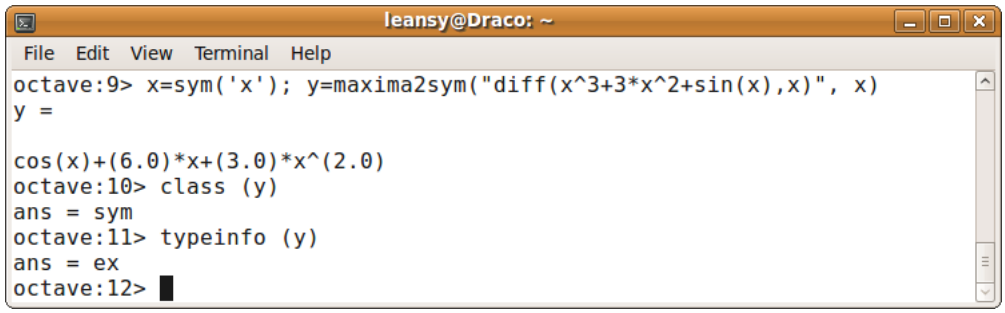

Figura 2.3: Ejecución en GNU Octave del comando **maxima2sym**.

4. **maximaFromFile.m**

   Recibe como primer argumento un fichero con funciones de GNU Maxima y en el resto de los argumentos se pueden hacer llamadas a estas funciones cargadas previamente.

En estas cuatro funciones la ejecución de los comandos se realiza directamente en GNU Maxima y el resultado devuelto por este programa es procesado en la función y posteriormente devuelto a la línea de comando de GNU Octave.
Para integrar esta solución a EIDMAT se creó un nuevo directorio dentro de la dirección de los códigos fuentes de EIDMAT dentro del cual se copiaron los archivos ".m" analizados. En los parámetros de inicio definidos para GNU Octave cuando se ejecuta en la VTE se le adiciona una nueva opción que permite incorporar el directorio creado con las nuevas funcionalidades en los caminos de búsqueda de GNU Octave para que pueda encontrar dichas funciones.

Las nuevas funciones agregadas a GNU Octave se encuentran disponibles desde la plataforma EIDMAT. En la línea de comandos aparecen las llamadas a las funciones **"maxima2num"**, **"maxima2str"** y **"maxima2sym"** y en el espacio de trabajo se muestran las variables "w","x", "y" y "z" y su tipo de dato (ver figura 2.4).

2.2. Interfaz gráfica para GNU Maxima en EIDMAT

EIDMAT se desarrolla en sus inicios como una interfaz gráfica para GNU Octave con similitud en muchos aspectos a Matlab. Basado en la arquitectura utilizada para ese fin, se reutilizan las clases y relaciones entre ellas extendiendo sus funcionalidades para obtener un nuevo ambiente de trabajo para GNU Maxima, esta nueva interfaz gráfica mantiene las opciones disponibles para GNU Octave, exceptuando las funcionalidades de cambio de directorio y el área de trabajo donde se muestran las variables que se van utilizando durante la ejecución del programa. Al menú principal se le añaden siete nuevas opciones específicas para GNU Maxima las cuales han sido tomadas del programa wxMaxima que es otra interfaz

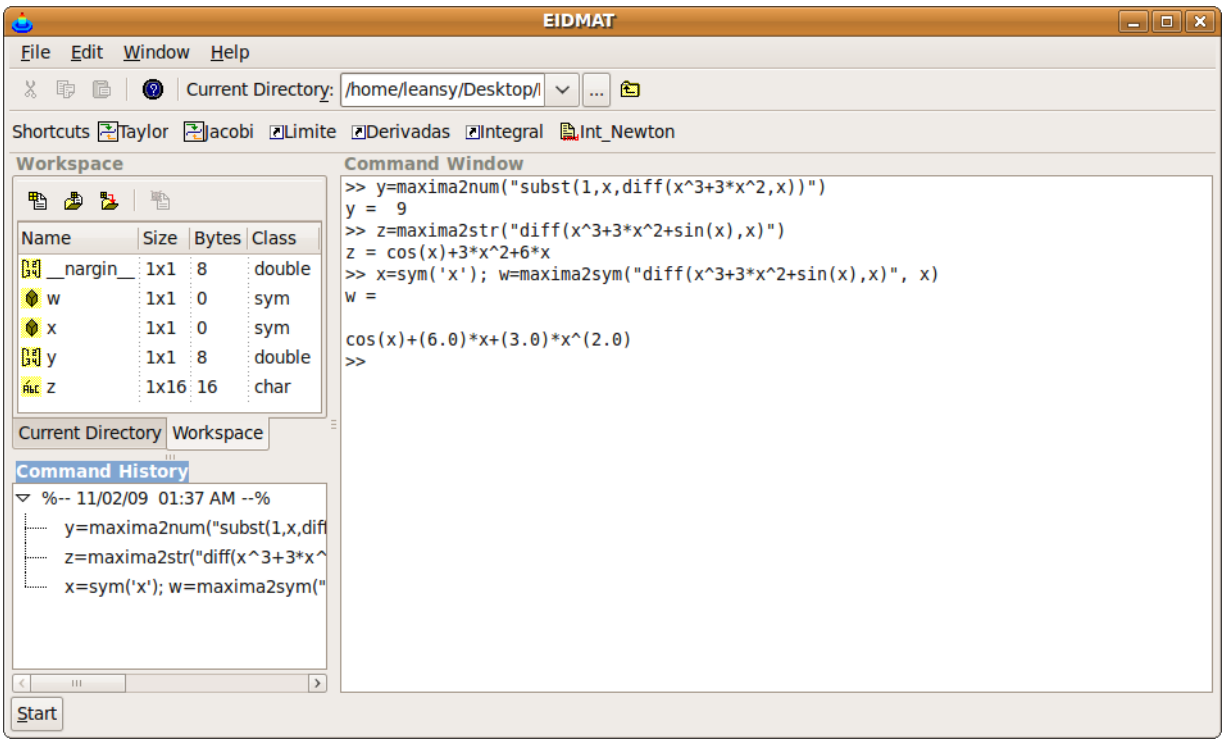

Figura 2.4: Ejecución de las nuevas funciones de GNU Octave en la plataforma EIDMAT.

gráfica de usuarios para GNU Maxima.

La interfaz de EIDMAT propuesta para el trabajo con GNU Maxima presenta las partes siguientes (ver figura 2.5).

1. **Menú principal**: presenta once categorías, a través de las cuales se pueden acceder a todas las opciones que brinda EIDMAT

2. **Barra de tarea fija**: presenta accesos rápidos a las acciones *Cortar*, *Copiar* y *Pegar* textos en el área de trabajo que se esta ubicado y un vínculo para llamar la ayuda de GNU Maxima

3. **Barra de accesos rápidos**: da la posibilidad de crear accesos rápidos personalizados por el usuario

4. **Historial**: contiene un listado de los comandos ejecutados

5. **Línea de comandos**: permite la entrada de comandos y muestra los resultados obtenidos

6. **Menú de inicio**: vínculos a la ayuda y a los accesos rápidos creados por el usuario

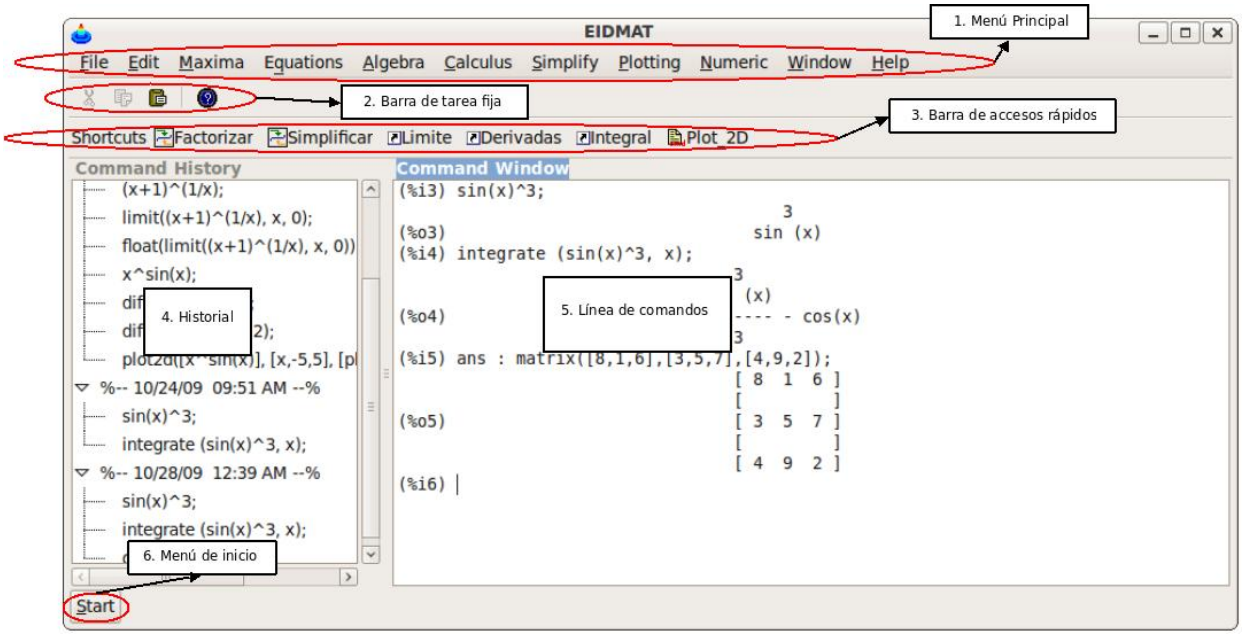

Figura 2.5: Interfaz gráfica de GNU Maxima en EIDMAT.

A continuación se listan los ficheros que debieron ser modificados de su versión original para obtener la interfaz gráfica de usuario mostrada en la figura 2.5.

```
                run_maxima.py
./cmd
                clear_cmdwindow_maxima.py
./cmdhistory
                cmdhistory_toolbar_maxima.py
                command_history_maxima.py
./cmdwindow
```

```
                command_window_maxima.py
./images
                gui_maxima.glade
./menubar
                cmdhistory_menu_bar_maxima.py
                cmdwindow_menu_bar_maxima.py
./menubar/menu
                edit_menu_maxima.py
                file_menu_maxima.py
                help_menu_maxima.py
                websrc_submenu_maxima.py
                window_menu_maxima.py
./menubar/menu/cmdhistory
                cmdhistory_edit_menu_maxima.py
                cmdhistory_file_menu_maxima.py
                cmdhistory_help_menu_maxima.py
                cmdhistory_window_menu_maxima.py
./menubar/menu/cmdwindow
                cmdwindow_edit_menu_maxima.py
                cmdwindow_file_menu_maxima.py
                cmdwindow_help_menu_maxima.py
                cmdwindow_window_menu_maxima.py
./mwindow
                main_window_maxima.py
                popup_menu_maxima.py
./toolbar
                cmdhistory_toolbar_maxima.py
                cmdwindow_toolbar_maxima.py
                context_toolbar_maxima.py
```

El siguiente listado muestra los ficheros nuevos que se adicionan a la solución. Estos ficheros junto a los que se han modificado anteriormente permiten obtener la nueva interfaz gráfica de usuario mostrada en la figura 2.5. Los nuevos ficheros contienen las clases necesarias para añadir las siete nuevas categorías del menú principal y los tres submenús que aparecen como opciones de la categoría de simplificación del menú principal.

```
./menubar/menu
                algebra_menu_maxima.py
                calculus_menu_maxima.py
                complex_simplification_submenu_maxima.py
                equations_menu_maxima.py
                factorial_submenu_maxima.py
                maxima_menu_maxima.py
                numeric_menu_maxima.py
                plotting_menu_maxima.py
                simplify_menu_maxima.py
                trigonometric_submenu_maxima.py
./menubar/menu/cmdhistory
                cmdhistory_algebra_menu_maxima.py
                cmdhistory_calculus_menu_maxima.py
                cmdhistory_equations_menu_maxima.py
                cmdhistory_maxima_menu_maxima.py
                cmdhistory_numeric_menu_maxima.py
                cmdhistory_plotting_menu_maxima.py
                cmdhistory_simplify_menu_maxima.py
./menubar/menu/cmdwindow
                cmdwindow_algebra_menu_maxima.py
                cmdwindow_calculus_menu_maxima.py
                cmdwindow_equations_menu_maxima.py
```

```
cmdwindow_maxima_menu_maxima.py
cmdwindow_numeric_menu_maxima.py
cmdwindow_plotting_menu_maxima.py
cmdwindow_simplify_menu_maxima.py
```

2.3. Comunicación desde GNU Maxima a GNU Octave

Para decidir qué tipo de conexión establecer desde GNU Maxima a GNU Octave se desarrolla una interfaz gráfica de usuario para GNU Maxima haciendo uso de la comunicación por sockets y la misma interfaz se implementa también haciendo uso de las tuberías de Python al sistema operativo.

Las clases y funciones principales de la implementación usando sockets se muestran en el Anexo 1. En él se aprecia el uso de hilos de ejecución para trabajar de forma asincrónica, la utilización del módulo subprocess para realizar llamadas al sistema operativo y poder ejecutar a GNU Maxima y por último, el uso del módulo socket para crear un servidor socket.

En la solución de comunicación con tuberías al sistema operativo se obtiene idéntico resultado utilizando solamente el módulo subprocess, la nueva solución permite eliminar el trabajo asincrónico al no tener que utilizar la clase **MaximaClient** que su instancia se ejecuta en un hilo diferente al principal en la versión de sockets; también se elimina la clase **MaximaServer** que se encarga de crear el servidor sockets y por último se modifica la clase **MaximaWindow** para eliminar el trabajo con las instancias de las dos clases suprimidas y sustituir en el método **sendCommand(self, cmd)** las líneas **self.maxima.send(cmd)** y **data=self.maxima.recv()** por:

**data=Popen([“maxima”, “--quiet”, “--batch-string”, cmd], **
stdout=PIPE).communicate()[0]

De esta forma se obtiene el mismo resultado con menor cantidad de líneas de código y menor gastos de recursos de la computadora para procesar las acciones de comunicación.

A raíz de la comparación anterior se decide realizar las llamadas a GNU Octave cuando se trabaja en el ambiente gráfico de GNU Maxima de la plataforma EIDMAT a través de tuberías al sistema, como se explicó en el primer capítulo (ver figura 1.3).
Las ficheros que se modifican para lograr este comportamiento son:

```
./conn
            terminal_maxima.py
            connection_maxima.py
```

Mientras que de modo particular dentro de estos ficheros se modifican las clases **Terminal** y **Connection** respectivamente, siendo

1. **Terminal**: donde se realizan los cambios necesarios para ejecutar GNU Maxima como asistente matemático en lugar de GNU Octave
2. **Connection**: toda la lógica para el procesamiento de los comandos de GNU Octave que se ejecutan desde GNU Maxima

La sintaxis definida para las llamadas a GNU Octave cuando se trabaja en el ambiente gráfico de GNU Maxima de EIDMAT es la siguiente:

octave(oct_cmd);

donde "oct_cmd" constituye un comando válido de GNU Octave.

Cada vez que se envía una orden por el editor de EIDMAT es procesada en la clase **Connexion** para definir si es enviada directamente para ser ejecutada en GNU Maxima o si necesita ser atendida primeramente por GNU Octave. Para identificar dicha orden se procesa la cadena de caracteres recibida en busca de la palabra *octave*, de encontrarse se puede asegurar que dicha sentencia es para ejecutar en GNU Octave debido a que en GNU Maxima no existe ninguna palabra reservada con ese nombre.

Cuando la sentencia recibida es de GNU Maxima se sigue el flujo normal de EIDMAT y se pone en la cola de los comandos que van a ser atendidos por GNU Maxima que se encuentra corriendo en la VTE. De lo contrario, si el comando se identifica como una orden para GNU Octave, se toma de la cadena de caracteres recibida la subcadena que se encuentra entre los paréntesis después de la palabra *octave*. En esta subcadena se pueden encontrar varios comandos para GNU Octave, los cuales aparecen separados por comas "**,**" y/o punto y comas "**;**", los comandos que aparecen terminados en punto y coma no devuelven el resultado, mientras que los terminados en comas sí. De estos últimos se necesita: obtenerlos por separados, encontrar el nombre de la variable al que se le asigna el resultado e identificar el tipo de dato almacenado en la variable. Los pasos de las transformaciones necesarias a la subcadena se enumeran y explican a continuación:

1. *Identificar los comandos terminados en comas dentro de la subcadena*: no se puede picar la subcadena cada vez que se encuentre una coma porque los parámetros de las funciones de GNU Octave se dividen por comas y cuando una función devuelve un vector los valores también se separan por comas, por tanto se debe procesar la subcadena para identificar que coma representa el fin de un comando, para ello cuando se encuentra una coma en la subcadena se asume que es un fin de comando válido si y solo si los paréntesis y corchetes desde el inicio de la subcadena hasta la posición de la coma están correctamente balanceados.

2. *Dividir la subcadena por comandos terminados en comas*: una vez identificada la posición del fin de un comando terminado en coma se pica la subcadena por dicho punto y se almacena como un elemento de una lista, obteniendo un lista donde cada línea de la misma es un comando de GNU Octave o varios comandos terminados por punto y coma, más uno adicional que era el que terminaba en coma

3. *Encontrar la variable que recibe el resultado*: cuando hay varios comandos separados por punto y coma, solo interesa la variable que recibe el resultado del comando que se encuentra después del último punto y coma, porque los anteriores no van a devolver valor alguno. Se pica la cadena contenida en el elemento de la lista que se procesa por

el último punto y coma, de las dos subcadenas obtenidas se toma la segunda y se busca la existencia del signo de asignación en ella, si se encuentra, la parte anterior a dicho signo puede ser el nombre de una variable o un vector, de no encontrar una asignación significa que GNU Octave asigna el resultado a una variable predefinida con nombre **ans**

4. *Identificar el tipo de dato*: para identificar el tipo de dato de la respuesta de los comandos de cada línea de la lista, se adiciona un nuevo comando de GNU Octave al final de la cadena de cada línea separado por una coma del anterior para que devuelva los dos resultados, el que debe devolver normalmente más la respuesta al comando adicionado. Hay varios comandos que permiten devolver el tipo de datos, **typeinfo** es uno de ellos, recibe como parámetro la variable asignada y devuelve el tipo de la misma

Al ejecutar los pasos descritos obtenemos como resultado una lista donde cada elemento es una cadena con varios comandos de GNU Octave, donde el último va a encuestar por el tipo devuelto en el comando anterior de la misma línea.

El siguiente paso consiste en construir una cadena única con todos los comandos de la lista. Para ello se crea una cadena vacía y se van adicionando cada una de las líneas separadas por comas; esta nueva cadena se utiliza como comando a ejecutar en GNU Octave a través de las tuberías de Python al sistema operativo y la línea de código en Python quedaría de la forma siguiente:

**result = Popen([“octave”, “-q”, “--eval”, cmd], **
stdout=PIPE).communicate()[0]

donde:

- “cmd” contiene la cadena construida previamente
- “result” recibe la respuesta de GNU Octave a los comandos ejecutados

Los resultados obtenidos de esta forma tienen que procesarse en Python para construir los comandos equivalentes en la sintaxis de GNU Maxima.

La variable "result" contiene el resultado de varios comandos donde la mitad de ellos dan respuesta al tipo de dato devuelto en el comando anterior de la cadena, por tanto, para construir las sentencias a ejecutar en GNU Maxima se tienen que agrupar los resultados obtenidos por parejas, el primero de los dos contiene el comando a ejecutar y el segundo el tipo de dato de dicha sentencia. La respuesta al tipo de datos siempre se devuelve en una sola línea, la estructura es: nombre de la variable, signo de asignación y el tipo de dato. El nombre de la variable donde se va a recibir el tipo de datos se construye con un nombre suficientemente largo más un número generado a partir de su índice en la lista de forma que no haya forma de que se repita en toda la cadena y se pueda utilizar de marca para dividirla y obtener la respuesta por separado de cada uno de los comandos GNU Octave ejecutados previamente. Como es conocida la cantidad de comandos ejecutados, se sabe a priori la cantidad de parejas de respuestas que se obtendrán. Los pasos que se repiten a continuación el número de veces definido por la cantidad de parejas, permiten dividir la cadena "result" en tuplas de subcadenas que contienen la expresión matemática y su tipo de datos:

1. Se corta la cadena almacenada en "result" por la marca fabricada a partir del nombre de la variable que recibe el tipo de datos, al realizarse este corte se logra obtener el resultado del primer comando aislado, en la otra parte obtenida se encuentra el resto de los resultados que faltan por procesar, encabezando esa subcadena aparece el tipo de dato del resultado ya separado

2. Para obtener el tipo de dato aparte del resto de la cadena, se tiene que picar por el primer salto de línea que aparezca en la cadena. Quedando en "result" el resto de las tuplas que aún no se han procesado

Cuando se termina de repetir estos paso, se obtiene una nueva lista, cada elemento es una tupla obtenida por cada iteración realizada. Por su parte, los nombres de los tipos de datos devueltos por GNU Octave utilizando la función **typeinfo** que interesan son:

1. “scalar”: para los valores numéricos
2. “sq_string”: para las cadenas de caracteres
3. “ex”: para las expresiones simbólicas
4. “matrix”: para las matrices

También se puede recibir como tipo de dato “list”, el mismo se obtiene cuando la respuesta del comando ejecutado en GNU Octave es un vector. En este caso excepcional no se utiliza **typeinfo** para conocer el tipo de dato debido que el valor devuelto no diferencia correctamente entre una matriz y un vector o entre un vector y un escalar.

En los tres primeros casos no hay necesidad de traducción de sintaxis, el resultado devuelto por GNU Octave es reconocido directamente por GNU Maxima. En el caso del tipo “matrix” y “list” sí es necesario transformar el formato de construcción de la sentencia. Para construir los comandos a ejecutar en GNU Maxima se itera por cada una de las tuplas almacenada en la lista construida previamente. Aplicando a cada tupla los pasos siguientes:

1. Se pregunta si el tipo de datos es “matrix”
2. Si no es de éste tipo, se pregunta si es de tipo “list”
3. Si su tipo tampoco coincide con el tipo “list”, se almacena el comando en una nueva lista que contienen elementos de tipo de dato cadena de caracteres, en esta lista cada elemento contiene la sentencia definitiva a ejecutar en GNU Maxima
4. Si es de tipo “matrix” se transforma siguiendo la sintaxis propia de GNU Maxima teniendo presente que la forma de construir una matriz vacía es diferente a una con elementos. Para procesar la cadena que contiene la matriz con el formato de GNU Octave se comprueba a partir de la primera línea de la cadena recibida si es una matriz nula o no, las matrices vacías en GNU Octave se representan de la forma siguiente: “[](0x0)” y las matrices con elementos cuando se devuelven como resultados de un

comando, contienen una línea por cada fila y las columnas se separan por espacios. Para especificar una matriz vacía en GNU Maxima se escribe "[]" y para construir una matriz de MxN se usa la sintaxis:

matrix([e11, e12, ... e1N], [e21, e22, ... e2N], ... [eM1, eM2, ... eMN]);

una vez realizada la transformación de la sentencia se almacena esta nueva cadena en la lista de comandos.

5. Si el tipo es "list" el vector devuelto desde GNU Octave contiene cada elemento del vector en una línea diferente, se toma cada valor y se construye una lista en la sintaxis de GNU Maxima. Una vez realizada la transformación de la sentencia se almacena esta nueva cadena en la lista de comandos

Esta lista de comandos de GNU Maxima ya lista para ejecución se pasan a la VTE para que se adicionen en la cola de comandos y comiencen a procesarse. El resultado de cada uno de ellos se muestra en la interfaz gráfica.

Ejemplo de ejecución (ver figura 2.6). En el historial a la izquierda de la línea de comando de la plataforma EIDMAT aparecen al final las dos últimas ordenes ejecutadas sobre GNU Maxima, esas dos sentencias ordenan ejecutar comandos en GNU Octave desde GNU Maxima, la primera adiciona un nuevo directorio a la variable "path" de GNU Octave y llama a una función definida en ese directorio y la última llama a la función interna de GNU Octave **"magic"** para crear un cuadrado mágico.

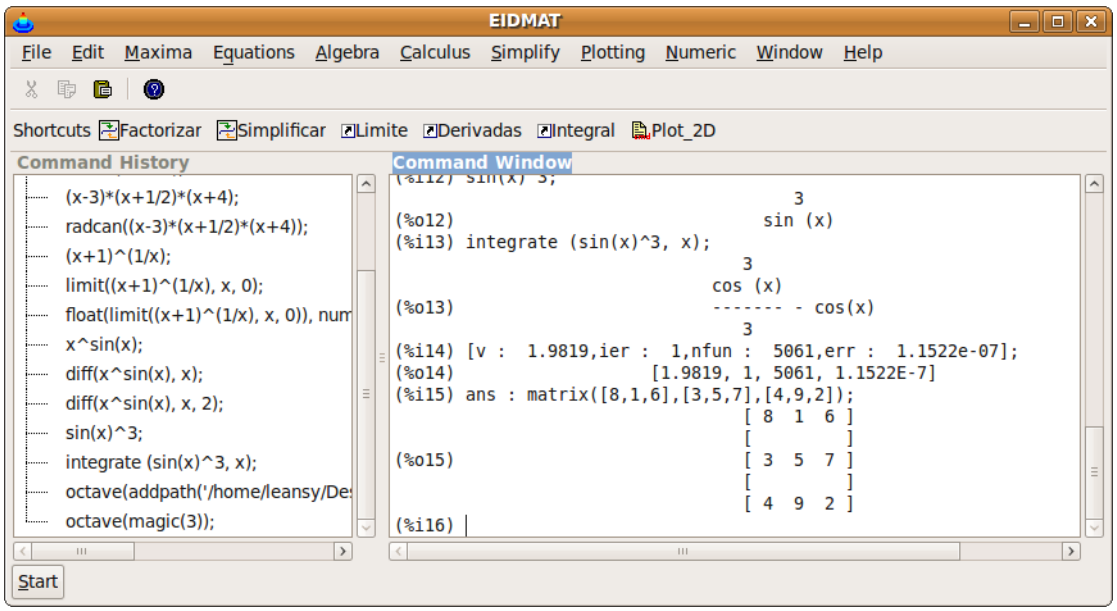

Figura 2.6: Llamadas a GNU Octave desde GNU Maxima ejecutándose en la plataforma EIDMAT.

2.4. Conclusiones

A modo de conclusiones parciales de éste capítulo se presentan los aspectos siguientes:

- Con las nuevas funcionalidades agregadas a GNU Octave, haciendo uso del asistente matemático GNU Maxima, es posible realizar cálculos simbólicos más complejos que los brindados por el módulo symbolic; permitiendo además utilizar funciones ya definidas en ficheros de GNU Maxima sin tener que reescribir los mismos para que funcionen en GNU Octave
- Con la nueva opción de utilizar en EIDMAT como interfaz gráfica para el sistema GNU Maxima, se obtiene una herramienta para cálculos matemáticos que puede ser utilizada tanto por usurarios de GNU Octave como por los usuarios de GNU Maxima
- El trabajo con sockets requiere de más esfuerzo para el desarrollo y el resultado obtenido por esta vía puede lograrse perfectamente con el trabajo por tuberías de forma más sencilla

- Las llamadas a GNU Octave desde GNU Maxima se realizan utilizando llamadas al sistema operativo a través de las herramientas propias del lenguaje Python haciendo uso de tuberías
- Con la disponibilidad de ejecutar comandos de GNU Octave desde GNU Maxima a través de EIDMAT, se amplía la cantidad de funciones disponibles para el cálculo numérico y se brinda la posibilidad de utilizar funciones definidas por usuarios de GNU Octave sin necesidad de reescribirlas para GNU Maxima

Capítulo 3

Pruebas del sistema

El presente capítulo recoge un conjunto de problemas resueltos donde se pone de manifiesto la necesidad de transitar del universo de las matemáticas numéricas al de las matemáticas simbólicas y viceversa. Los problemas, aunque poseen una componente matemática importante, esta se relega a un segundo plano exponiéndose con mayor detalle el tratamiento de los mismos desde el punto de vista computacional mediante el empleo de la plataforma EIDMAT.

3.1. Análisis de diversos problemas matemáticos

3.1.1. Situación problémica #1

En esta primera situación se analiza un caso donde se hace necesario invocar a GNU Maxima desde GNU Octave para realizar el cálculo del gradiente de cierta función simbólica.

Planteamiento del problema

La superficie de un lago está representada por una región **D** en el plano XY de manera que la profundidad (en metros) bajo el punto correspondiente a (x, y) viene dada por

$$f(x, y) = 300 - 2x^2 - 3y^2$$

Una niña se encuentra en el agua en el punto $(4, 9)$ referido al plano de referencia. ¿En qué dirección debe nadar para que la profundidad del agua bajo ella disminuya más rápidamente?.

Respuesta comentada

Para dar solución a la situación presentada se hace necesario contar con algunos conocimientos teóricos básicos sobre las funciones vectoriales de variable escalar y su tratamiento; no obstante, dado los objetivos que se persiguen en este trabajo solo se presenta el método de solución con énfasis en los cálculos y se deja al lector la profundización del tema a través de [13] u otra bibliografía afín.

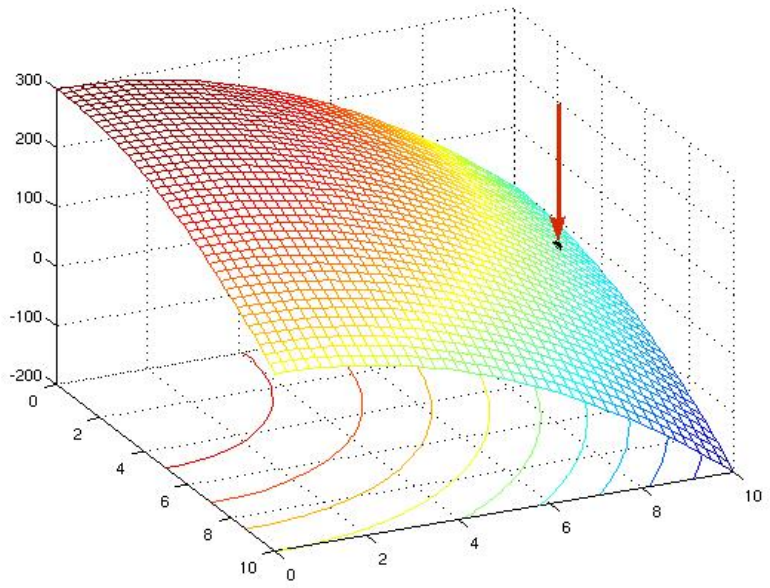

Figura 3.1: Comportamiento de la profundidad del lago en función de la posición.

A partir de una representación de la función $f(x, y)$ en forma de malla colorimétrica podemos tener un mejor acercamiento hacia el comportamiento de la profundidad del lago así como la situación concreta en la posición actual de la niña (ver figura 3.1). Luego, para alcanzar la solución definitiva del problema basta con determinar el gradiente de la función

$f(x, y)$ en el punto $(4, 9)$ lo cual es representativo de la dirección en la cual la función experimenta el mayor crecimiento (figura 3.2 (Izq.)). Este procedimiento puede llevarse a cabo empleando la plataforma EIDMAT y los mecanismos de interconexión entre los asistentes matemáticos GNU Octave y GNU Maxima toda vez que el problema no puede ser resuelto entera y únicamente desde el ambiente de GNU Octave dada sus limitaciones para el tratamiento de expresiones simbólicas.

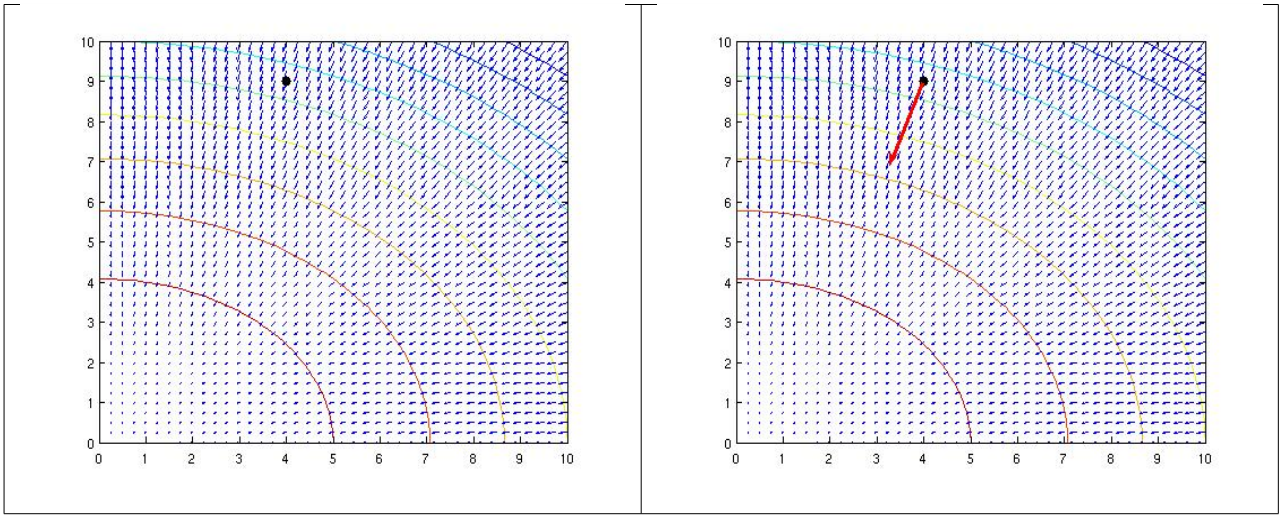

Figura 3.2: Campo de gradientes de la función $f(x, y)$ para una malla arbitraria. (Izq) Campo vectorial, (Der) Gradiente en el punto $(4, 9)$

Concretamente, se implementa la función **"*gradiente*"** como extensión de GNU Octave (ver Anexo 2) y se invoca ésta tomando como parámetros la función $f(x, y)$ así como la posición actual de la niña. Una vez ejecutado el cálculo tenemos como respuesta al problema $G = [-16, -54]$ lo cual es un vector que representa al gradiente de la propia función $f(x, y)$[1] en la posición actual de la niña (figura 3.2 (Der.)).

En términos del ángulo que forma el gradiente con el semieje "X" positivo, éste puede determinarse a partir de sencillas transformaciones geométricas dando como resultados un

[1]Se refiere a $\nabla f(x, y) = (f_x(x_0, y_0), f_x(x_0, y_0))$

ángulo de 253,5° (i.e. la dirección en que debe nadar la niña es $\theta = 253{,}5°$ respecto al semieje "X" positivo).

3.1.2. Situación problémica #2

En éste segundo caso se presenta y analiza una situación donde se hace necesario invocar al GNU Maxima desde el GNU Octave para realizar el cálculo de derivadas de diversos órdenes así como para la realización de sustituciones en ciertas funciones simbólicas. Se sigue este proceder en buena medida debido a las amplias potencialidades de la función ***"subst"*** que implementa GNU Maxima.

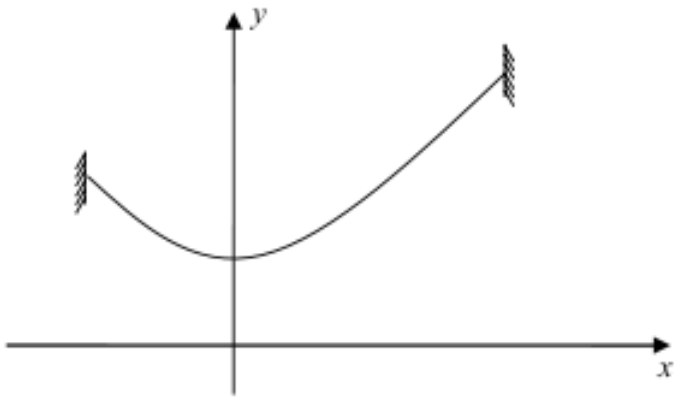

Figura 3.3: Características de la Catenaria.

Planteamiento del problema

Un cable que cuelga de sus extremos adquiere la forma que se muestra en la figura 3.3 y se demuestra que la ecuación de esta curva (llamada Catenaria) viene dada por

$$y = L + acosh(\frac{x}{a})$$

donde el parámetro a depende del peso por unidad de longitud del cable y de la tensión a que es sometido mientras que L depende de la posición del sistema de referencia, el cual en nuestro caso se encuentra colocado de manera que el origen de coordenadas está justamente

debajo del punto de altura mínima. Planteada esta situación determine la ecuación de la catenaria en cuestión si se conoce que su altura mínima es de 15 m y que 10 m más allá, la altura de la misma es de 17 m.

Respuesta comentada

Para dar solución al problema planteado primeramente se debe hacer uso de los datos aportados y transformar la expresión inicial de la Catenaria según:

Para $x = 0$ se tiene $y = 15$ y, por tanto: $L + a = 15$
Para $x = 10$ se tiene $y = 17$ y, por tanto: $L + acosh(\frac{10}{a}) = 17$

Basta restar ambas ecuaciones para eliminar la incógnita L y obtener:

$$acosh(\frac{10}{a}) - a = 2$$

una ecuación cuya raíz da como resultado el valor de a.

La ecuación anterior pertenece a la familia de las ecuaciones no-lineales cuya solución no puede ser determinada mediante el empleo de técnicas convencionales de cálculo sino necesariamente mediante el empleo de técnicas de cálculo numérico. De modo particular en este caso se selecciona el método de Newton[2] para solucionar la misma [14]. El método de Newton pertenece a los métodos de punto fijo[3] y es conocido también por el método de las tangentes debido a que su funcionamiento consiste en el trazado sucesivo de tangentes mientras se acerca paulatinamente a la solución definitiva de la ecuación siempre analizando una tolerancia como parámetro de finalización. Como es natural, la determinación de una tangente implica necesariamente el cálculo de una derivada la cual en muchos casos (y en particular el nuestro) parte de poseer cierta función simbólica como argumento de trabajo por lo que intentar la solución únicamente desde el universo numérico es por solo mencionarlo, un hecho infructuoso.

[2]Matemático, físico, filósofo y teólogo inglés (4 de enero de 1643 – † 31 de marzo de 1727).
[3]Refiérese a los métodos cuya solución tienen la forma $x_i = f(x_{i-1})$

Atendiendo a las premisas anteriores se resuelve el problema empleando la función "***newton***" implementada, al igual que la anterior, como extensión del GNU Octave (ver Anexo 3) de donde se obtienen los resultados siguientes:

Código de entrada:

```
>> fun='a*cosh(10/a)-a-2';
>> var='a';
>> [c,E,fc]=newton(fun,var,20,30,0.005,20)
```

Salida:

```
--------|----------|------------|
Iter.       Aprox.     Error.
--------|----------|------------|
ini        20.0000      10.0000
1          24.1567       4.1567
2          25.2701       1.1134
3          25.3264       0.0564
4          25.3269       0.0005
--------|----------|------------|
c =  25.327
E =  4.8720e-04
fc =  -4.1543e-05
```

A partir de éste resultado ($a = 25{,}327$) se plantea la relación

$$L = 15 - a = 15 - 25{,}327 = -10{,}327$$

de donde podemos finalizar

$$y = -10{,}327 + 25{,}327 * cosh(\frac{x}{25{,}327})$$

3.1.3. Situación problémica #3

Para mostrar las posibilidades que brinda la conexión desde GNU Maxima hacia GNU Octave, se presenta seguidamente una situación donde el paso al universo numérico es inevitable como parte del proceso de solución.

Planteamiento del problema

Supóngase que se se desea determinar el valor de la integral

$$\int_0^3 xsin(\frac{1}{x})\sqrt{\mid 1-x \mid}dx$$

por simple observación, es de notar que el procedimiento no es nada sencillo máxime si se desea realizar desde el punto de vista simbólico (véase la figura 3.4). Por tal razón se impone el empleo de los algoritmos numéricos para conseguir el resultado deseado.

Respuesta comentada

En el caso que se analiza se parte de considerar además que se trabaja desde el universo simbólico, esencialmente con GNU Maxima y se desea realizar una petición de ayuda a GNU Octave a través de los mecanismos implementados a tales efectos en la plataforma EIDMAT. El resultado es el siguiente.

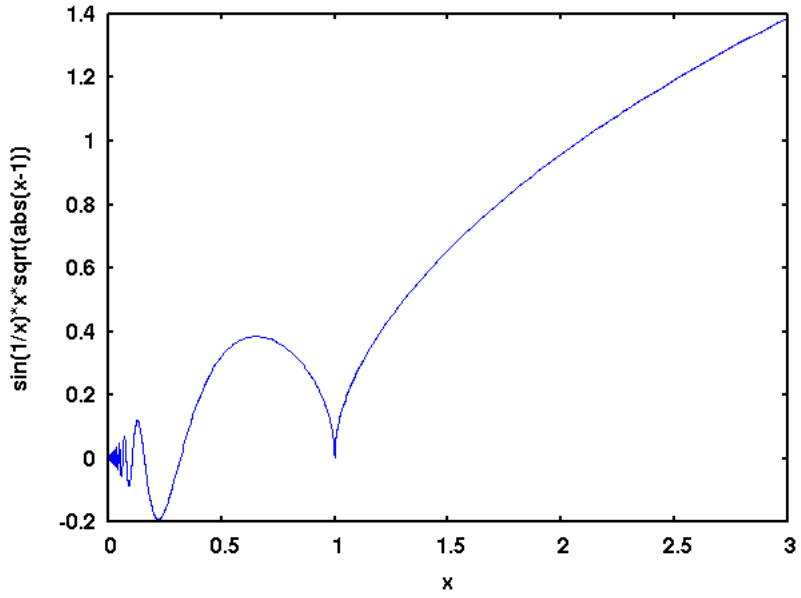

Figura 3.4: Representación gráfica de la función $xsin(\frac{1}{x})\sqrt{\mid 1-x \mid}$ en el intervalo {0,3}.

Código de la solicitud:

```
(%i1) octave(addpath('/home/alexeis/Work/Leansy_Tesis'); ...
      [I, Ier, Nfun, Err]=quad("f",0,3,0.005))
```

Salida:

```
(%i1) [I: 1.9818, Ier: 0, Nfun: 357, Err: 0.0033485];
(%o1)                    [1.9818, 0, 357, 0.0033485]
```

Aquí la dirección *"home/alexeis/Work/Leansy_Tesis"* representa el directorio donde se ha definido la función **"f"**; mientras que la definición de ésta última se puede consultar en el Anexo 4. Por su parte los parámetros de la salidas representan

I	$\Longrightarrow$	Valor aproximado de la integral
Ier	$\Longrightarrow$	Denota la existencia o no de errores durante el cálculo (1 si existieron, 0 si no)
Nfun	$\Longrightarrow$	Representa el número total de evaluaciones
Err	$\Longrightarrow$	Error de la aproximación

Conclusiones

La realización de éste trabajo satisfizo en su totalidad los objetivos trazados, pudiéndose destacar de manera general las conclusiones siguientes:

1. Se implementó un mecanismo que permite la integración entre los asistentes matemáticos GNU Octave y GNU Maxima que posibilita el efectivo trasiego de información entre los ambientes de la matemática simbólica y numérica

2. Se creó un mecanismo de ejecución de comandos de GNU Maxima desde GNU Octave

   a) Con las nuevas funcionalidades agregadas a GNU Octave, haciendo uso del asistente matemático GNU Maxima, es posible realizar cálculos simbólicos más complejos que los que pueden efectuarse con las funciones brindadas por el módulo "symbolic" de GNU Octave; permitiendo además utilizar funciones ya definidas en ficheros de GNU Maxima sin tener que reescribir los mismos

3. Se creó un mecanismo para la ejecución de GNU Maxima dentro de la plataforma EIDMAT que permite además realizar llamadas a comandos de GNU Octave e integrar sus respuestas al ámbito de GNU Maxima

   a) Con la disponibilidad de ejecutar comandos de GNU Octave desde GNU Maxima a través de EIDMAT, se amplía la cantidad de funciones disponibles para el cálculo numérico y se brinda la posibilidad de utilizar funciones definidas por usuarios de GNU Octave sin necesidad de reescribirlas para GNU Maxima

4. Se propuso una interfaz para el uso de GNU Maxima en EIDMAT que mantiene las funcionalidades presentes en la aplicación wxMaxima e incorpora mejoras en torno a

la interactividad

a) La interfaz propuesta se ajusta a las pautas de diseño definidas para el uso de GNU Octave desde la plataforma EIDMAT

Recomendaciones

El desarrollo de herramientas de cálculos que aunen bajo un mismo ambiente las bondades de diversos universos matemáticos constituyen un aporte considerable a la calidad de la enseñanza en la UCI y el Ministerio de Educación Superior Cubano. En la búsqueda de este anhelo el presente trabajo constituye un paso importante y en lo particular un hito para el desarrollo posterior de la herramienta EIDMAT, por lo que se recomienda:

1. Completar la integración de la interfaz para GNU Maxima en el ambiente de EIDMAT
2. Redefinir la arquitectura de EIDMAT a fin de soportar un crecimiento de forma modular
3. Añadir un módulo que permita el efectivo trasiego de información entre los universos de la matemática numérica y estadística

Referencias

[1] R. Goering, "*Matlab edges closer to electronic design automation world.*" http://www.eetimes.com/news/design/showArticle.jhtml?articleID=4940039, Enero 2009.

[2] MathWorks, "*The MathWorks Store.*" http://www.mathworks.com/store/priceListLink.do, Julio 2009.

[3] G. EIDMAT, "*La cara de EIDMAT en internet.*" http://www.eidmat.wordpress.com, Septiembre 2009.

[4] A. Companioni, "Entorno integrado para el desempeño matemático," *Aceptado para publicar*, 2009.

[5] J. W. Eaton, "*About GNU Octave.*" http://www.gnu.org/software/octave/about.html, Julio 2009.

[6] I. The MathWorks, "*Symbolic Math Toolbox.*" http://www.mathworks.com/access/helpdesk/help/toolbox/symbolic/index.html?/access/helpdesk/help/toolbox/symbolic/f0-49321.html, Septiembre 2009.

[7] T. O. Community, "*Octave Symbolic Manipulation Toolbox.*" http://octave.sourceforge.net/symbolic/local/symbolic.html, Julio 2009.

[8] R. Grothmann, "*Euler Math Toolbox.*" http://eumat.sourceforge.net/, Junio 2009.

[9] P. L. Lucas, "*Traductor de máxima a Octave.*" http://qtoctave.wordpress.com/2008/02/06/traductor-de-maxima-a-octave/, Julio 2009.

[10] P. S. Foundation, "*About Python.*" http://www.python.org/about/, Julio 2009.

[11] *Aprenda a Pensar Como un Programador con Python.* Green Tea Press, 2002.

[12] P. S. Foundation, "*PyGTK 2.0 Reference Manual.*" http://www.pygtk.org/, Agosto 2007.

[13] *CÁLCULO CON GEOMETRÍA ANALÍTICA.* Grupo Editorial Iberoamérica S.A. de C.V., 2003.

[14] *CURSO DE ANÁLISIS DE MATEMÁTICO.* MIR. Moscú, 1988.

Bibliografía

[1] *Manual de Referencia de Maxima.*, `http://maxima.sourceforge.net/docs/manual/en/maxima.pdf`, Marzo 2008.

[2] Dodier, R. *Minimal Maxima.*, Septiembre 2005.

[3] Belanger, J. *Maxima and Emacs.*, Septiembre 2005.

[4] Rodríguez Riotorto, M. *Primeros pasos en Maxima.*, Marzo de 2008.

[5] Eaton, John W.; Bateman, D. & Hauberg, S. *GNU Octave, A high-level interactive language for numerical computations*, 3er Edition for Octave version 3.0.1, Julio 2007.

[6] Eaton, John W.; Bateman, D. *Octave FAQ. Frequently asked questions about Octave*, Septiembre 2007.

[7] Jiménez, A. J. *Computación numérica bajo Linux y Windows.*, Universidad de Cádiz.

[8] Juárez, M. C.; Herrera, W. G. & Sánchez, S. T. *Software Libre vs. Software Propietario. Ventajas y desventajas.*, México, 2006.

[9] Nogueras, G. B. *Octave: Una alternativa real a Matlab a coste de cero.*, 7 p.

[10] Rebollo, T. C.; Vera, E. C. & Mármol, M. G. *Notas sobre Matlab y Octave.*, 29 p.

[11] Pérez, M.; Saus, E. A. & Pérez, E. G. *Sistemas de Computación Algebraicos. Evolución y Aplicaciones.*, Universidad de Castilla-La Mancha.

[12] Escobar, H. M. M. *Introducción a SCILAB*, Departamento de Matemáticas, Universidad Nacional de Colombia, Bogotá, 2005.

Anexos

Anexo 1: Comunicación por sockets a GNU Maxima

Solo se muestran los fragmentos del código relacionado con la comunicación por sockets, eliminando los detalles de la interfaz gráfica.

Nombre del fichero: SocketsMaxima.py

```
import socket
import subprocess
import threading
...
class MaximaClient(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.start()

    def run(self):
        subprocess.call(["maxima","-s","50000"])

class MaximaServer:
    def __init__(self):
        self.host = ''
        self.port = 50000
        self.server = None
```

```
        self.client = None
        self.conn = None
        self.bufferSize = 4096
        self.buffer = ""

    def startServer(self):
        self.server = socket.socket()
        self.server.bind((self.host, self.port))
        self.server.listen(1)

    def startMaxima(self):
        self.client = MaximaClient()

    def connect(self):
        self.conn, self.addr = self.server.accept()

    def send(self, command):
        self.conn.sendall(command)

    def recv(self):
        data = self.conn.recv(self.bufferSize)
        return data

    def close(self):
        self.conn.close()

class MaximaWindow:
    def __init__(self):
        ...
```

```
    self.maxima = MaximaServer()
    self.maxima.startServer()
    self.maxima.startMaxima()
    self.maxima.connect()
...
def sendCommand(self, cmd):
    self.maxima.send(cmd)
    data = self.maxima.recv()
    iter = self.buffer.get_end_iter()
    self.tvConsole.get_buffer().insert(iter, data)
    self.tvConsole.scroll_to_iter(iter,0.0)
```

Anexo 2: Cálculo del gradiente auxiliado por GNU Maxima

```
function G=gradiente(fun,point)
% syntaxis: G=gradiente(fun,point)
% ------------------------------------------------------------------------
% Esta funcion permite calcular el gradiente de cierta funcion de varias
% variables (2 vars) mediante llamadas a GNU Maxima.
%
% Entrada:
% fun    --> Funcion en forma de cadena
% point  --> Punto donde se desea determinar el gradiente
%
% Salida:
% G  --> Vector con las componentes del gradiente determinado

%% Validacion de las entradas
if nargin<2
    error('Verifique los datos de entrada.');
else
    if min(size(point))>1
        error('<point> debe ser un vector.');
    end

    if (length(point)<2) || ((length(point)>2))
        error('El vector <point> debe ser de dimension [1x2] o [2x1].');
    end

    if ischar(fun)<1
```

```
        error('Verifique que el parametro <fun> sea una cadena.');
    end
end

point=point(:)';
xp=num2str(point(1,1));
yp=num2str(point(1,2));

StrY=maxima2str( strcat('subst(',xp,',x,diff(',fun,',x))') );
G(1,1)=maxima2num( strcat('subst(',yp,',y,',StrY,')') );

StrX=maxima2str( strcat('subst(',yp,',y,diff(',fun,',y))') );
G(1,2)=maxima2num( strcat('subst(',xp,',x,',StrX,')') );
```

Anexo 3: Algoritmo numérico del método de Newton auxiliado por GNU Maxima

```
function [c,E,fc]=newton(fun,var,a,b,err,n_iter)
% syntaxis: [c,E,fc]=newton(fun,var,a,b,err,n_iter)
% ------------------------------------------------------------------------
% Esta funcion permite determinar de manera aproximada el valor de la
% raiz de una ecuacion no lineal f(x)=0 mediante el metodo de Newton.
% De manera adicional se proporciona una tabla con el valor de la
% aproximacion y el error corespondiente en cada iteracion.
%
% Entrada:
% fun    --> Funcion en forma de cadena expresada de la forma f(x)=0
% var    --> Variable de trabajo
% a, b   --> Extremos del intervalo
% err    --> Tolerancia del calculo {default | 0.00001}
% n_iter --> Numero maximo de iteraciones {default | 1000}
%
% Salida:
% c  --> Aproximacion encontrada
% E  --> Error de la aproximacion "c"
% fc --> Valor de la funcion en la aproximacion

%% Validacion de la entrada
if (nargin<4)
    error('Revise los datos de entrada.');
else
    if ischar(fun)<1
        error('La funcion fun debe expresarse como cadena.');
```

```
    end

    if sum(size(a))≠2 || sum(size(b))≠2
        error('Los extremos del intervalo deben ser escalares.');
    end

    if ischar(var)<1 || length(var)>1 || var==' '
        error('Revise la definicion de la variable de trabajo.');
    end
end

if (nargin<4)
    err=0.00001;
    n_iter=1000;
elseif (nargin<5)
    n_iter=1000;
    if sum(size(error))≠2
        error('La tolerancia debe ser un escalar.');
    end
else
    if sum(size(err))≠2
        error('La tolerancia debe ser un escalar.');
    end

    if round(n_iter)-n_iter≠0
        error('El numero de iteraciones debe ser un entero.');
    end
end
```

```
%% Teorema de Bolzano
if subs_temp(a,var,fun)*subs_temp(b,var,fun)>0
    error('Imposible de aplicar el metodo: note que --> f(a)*f(b)>0');
end

%% Metodo de Newton
if subs_temp(a,var,fun)*...
   subs_temp(a,var,maxima2str(strcat('diff(',fun,',',var,',2)')))>0
    x=a;
else
    x=b;
end

diff_fun=maxima2str(strcat('diff(',fun,',',var,')'));
E=b-a;
xa=x; i=0;
RES(1,1)=x; ERR(1,1)=E;

while (E>=err) && (i<n_iter)
    if subs_temp(x,var,fun)==0
        fprintf('La raiz es exactamente: %f',x);
        break;
    else
        x=xa-( subs_temp(xa,var,fun) / subs_temp(xa,var, diff_fun) );
    end
    E=abs(x-xa);
    xa=x;
    i=i+1;
    RES(i+1,1)=x; ERR(i+1,1)=E;
```

```
end

%% Formateo de las salidas
disp('--------|----------|-----------|');
fprintf('Iter.       Aprox.     Error.   \n');
disp('--------|----------|-----------|');
fprintf(' %-10s  %-11.4f  %-12.4f \n','ini',RES(1,1),ERR(1,1));
for i=2:max(size(RES))
    fprintf(' %-10.0d  %-11.4f  %-12.4f \n',i-1,RES(i,1),ERR(i,1));
end
disp('--------|----------|-----------|');

c=x; fc=subs_temp(x,var,fun);

%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------

function N=subs_temp(num,var,fun)
N=maxima2num( strcat('subst(',num2str(num),',',var,',',fun,')') );
```

Anexo 4: Definición de la función auxiliar “f“

```
function y=f(x)
% syntaxis: y=f(x)
% ---------------------------------------------------------------------
% Esta funcion auxiliar se emplea para poder realizar evaluaciones durante
% los calculos numericos. En este caso se ha definido para poder evaluar
% variables vectoriales.
%
% Entrada:
% x --> Vector de valores
%
% Salida:
% y --> Vector de valores
%

y=x.*sin(1./x).*sqrt(abs(1-x));
```