



HERRAMIENTAS INFORMÁTICAS PARA MATEMÁTICAS

(Apuntes elaborados por el equipo docente para el curso 2020-21)

Tema 1

INTRODUCCIÓN A LOS PROGRAMAS DE CÁLCULO MATEMÁTICO

Los programas de cálculo matemático son programas de computador o de calculadora avanzada para el cálculo numérico y/o simbólico. Estos programas nos permiten, a diferencia de una calculadora tradicional no programable, automatizar manipulaciones repetitivas, tediosas o difíciles. Pero entre ellos también hay grandes diferencias, así la principal diferencia entre un programa de cálculo simbólico y uno de cálculo numérico es la habilidad del primero para trabajar con ecuaciones y fórmulas simbólicamente, en lugar de numéricamente. De ahí que una expresión como $a + b$ pueda ser interpretada como “la suma de dos variables”, y no como “la suma de dos números” que ya tienen valores asignados.

Aunque existen programas especializados en cálculo numérico y en cálculo simbólico, la mayoría de los programas son capaces de realizar cálculos matemáticos, tanto numéricos como simbólicos. A continuación se recoge una lista de programas de cálculo matemático, clasificados por el ámbito en el que son más conocidos. La lista no pretende ser exhaustiva pues además de estos existen muchos programas especializados en las distintas áreas de las Matemáticas.

Tipo de cálculo	Programa	Web oficial	Tipo de software
Numérico	Matlab	http://www.mathworks.es/	Comercial
	Octave	http://www.gnu.org/software/octave/index.html	Libre
	Scilab	http://www.scilab.org/	Libre
Simbólico	Derive	http://www.derive.com/	Comercial
	Maple	http://www.maplesoft.com/	Comercial
	Mathematica	http://www.wolfram.com/	Comercial
	Maxima	http://maxima.sourceforge.net/es/	Libre

Los contenidos de la asignatura se van a cubrir en base a dos programas gratuitos, uno de cálculo numérico (Scilab) y otro de cálculo simbólico (Maxima), disponibles para varias plataformas. Con la particularidad de que ambas herramientas también están muy documentadas por sus propios desarrolladores y usuarios bajo los términos de la "GNU

General Public License" tal como lo publica la "Free Software Foundation". Ambos programas se apoyan en un lenguaje interpretado, ofrecen grandes opciones para las representaciones gráficas, permiten programación estructurada y están preparados para manipular archivos de programas.

1.1 DIFERENCIAS ENTRE LOS PROGRAMAS DE CÁLCULO NUMÉRICO Y DE CÁLCULO SIMBÓLICO

Con carácter introductorio, pues en otros temas se pondrán de manifiesto muchas más diferencias entre los programas de cálculo numérico y de cálculo simbólico, vamos a centrar las diferencias en los siguientes aspectos: el tratamiento de las expresiones numéricas, el tratamiento de las expresiones literales, la representación interna de la información y las características más representativas de ambos programas.

1.1.1 Tratamiento de las expresiones numéricas

Las expresiones numéricas se forman con números o estructuras de números y operadores que especifican los cálculos a ser realizados. Pueden ser evaluadas por cualquier programa de cálculo numérico, y el resultado será siempre un número que se presentará con un formato decimal concreto, el seleccionado en ese momento. Por ejemplo, la expresión numérica $4/12+sqrt(2)$ provoca el resultado 1.7475469 en Scilab. El resultado puede haber sufrido algún tipo de truncamiento debido a la precisión del cálculo.

En cambio los programas de cálculo simbólico, que también son capaces de evaluar las expresiones numéricas, devuelven los resultados numéricos de forma exacta, sin aproximaciones decimales, como fracción simplificada. Aunque también se les puede pedir que devuelvan el resultado numérico en formato decimal o que realicen el cálculo con una precisión concreta. Por ejemplo, la expresión numérica $4/12+sqrt(2)$; provoca el resultado $\sqrt{2} + \frac{1}{3}$ en Maxima.

1.1.2 Tratamiento de las expresiones literales

Las expresiones literales se forman con letras, números, o estructuras de letras y números, y operadores que especifican los cálculos a ser realizados. Los programas de cálculo numérico sólo son capaces de evaluar expresiones literales en las que todas las variables ya tengan valores asignados. Mientras que los programas de cálculo simbólico operan con las variables de forma analítica, tal como lo haríamos nosotros. Por ejemplo, la expresión literal $a/b+sqrt(c)$; provoca el resultado $\sqrt{c} + \frac{a}{b}$ en Maxima. Sin embargo para que pueda ser evaluada por Scilab tiene que estar precedida de las correspondientes asignaciones para las tres variables. Luego la siguiente expresión literal

$$a=4,b=12,c=2,a/b+sqrt(c)$$

donde a la variable a se le asigna el valor 4, a la variable b el valor 12 y a la variable c el valor 2 sí podría ser evaluada por Scilab, dando el mismo resultado 1.7475469 de la expresión numérica comentada anteriormente.

La versatilidad del programa de cálculo simbólico queda aún más de manifiesto si en la expresión literal es posible algún tipo de simplificación. Por ejemplo, la expresión literal $c:b\$ a/b*\text{sqrt}(c)$; provoca el resultado $\frac{a}{\sqrt{b}}$ en Maxima. Con la primera parte de la expresión le hemos comunicado a Maxima que la variable c es igual a la variable b , por tanto al evaluar la segunda parte de la expresión ha podido realizar la siguiente simplificación $\frac{a}{b} \sqrt{b} = \frac{a}{\sqrt{b}}$ y llegar a ese resultado final.

Los programas de cálculo simbólico están además preparados para realizar distintas representaciones de un mismo resultado, con especial atención a los formatos más empleados en la escritura habitual. De ahí que suelen contemplar, entre otras cosas, la simplificación de una expresión a la forma más simple o a una forma estándar y el cambio de formatos.

1.1.3 Representación interna de la información

Para sacar el máximo provecho a un programa de cálculo matemático es conveniente saber qué tipo o tipos de representación interna de la información emplea.

En Scilab todo se representa como una matriz (array en inglés) con excepción de las estructuras de datos más complejas como las listas que a su vez pueden contener matrices. La razón principal es que de esta forma el acceso a un dato cualquiera es directo, mientras que en las listas, al ser estructuras dinámicas, para acceder al n -ésimo dato, un puntero debe recorrer todos los elementos de la lista desde el que ocupa el primer lugar. Por tanto los tiempos de procesamiento de grandes cantidades de datos se reducen.

Por el contrario, toda expresión de Maxima se representa internamente como una lista, lo que no es de extrañar habida cuenta de que Maxima está programado en *Lisp* (*List Processing*). No obstante también permite el almacenamiento matricial de datos siempre y cuando el tamaño de la matriz se haya declarado previamente.

1.1.4 Las características más importantes de los programas de cálculo numérico

Entre las características comunes de los programas de cálculo numérico podemos destacar las siguientes:

- Emplean la notación científica o coma flotante, con mantisa (parte decimal con signo) y exponente, que facilita la representación tanto de números grandes como de pequeños pero que no es una representación exacta para todos los números.
- Los cálculos no son exactos, tienen inherentemente cierto error, de ahí que se hable de la precisión en el cálculo.
- Muchos de los cálculos involucran procesos de iteración, que pueden llegar a término (presentación de resultados) si han agotado el número de iteraciones previsto o si han alcanzado la precisión deseada (se habla entonces de convergencia de la solución).

- Son capaces de encontrar soluciones aproximadas a ciertos problemas matemáticos que, por su complejidad o porque no tienen solución, son imposibles de abordar analíticamente.
- Están capacitados para el análisis y tratamiento masivo de datos.


1.1.5 Las características más importantes de los programas de cálculo simbólico

Entre las características comunes de los programas de cálculo simbólico podemos destacar las siguientes:

- Emplean representación y manipulación exacta de la información.
- Están capacitados para los cálculos matemáticos básicos (límites, derivadas, integrales) sobre funciones de una y varias variables.
- Están capacitados para la resolución analítica de ciertos sistemas de ecuaciones y de ciertas ecuaciones diferenciales.

1.2 PRIMEROS PASOS CON SCILAB

Consiga el archivo correspondiente a su sistema operativo para instalar la versión de Scilab recomendada por el equipo docente. El apartado **Download** de la página web oficial de Scilab <http://www.scilab.org/> le ayudará a localizarlo y le facilitará información para su instalación. También le puede ser de gran ayuda el apartado 1.3 del documento **Introduction to Scilab**.

Durante el proceso de instalación de Scilab se recomienda elegir el idioma español y la instalación por defecto. Si ha elegido la versión 6.1.0 para Windows, la que ha utilizado el equipo docente al elaborar estos apuntes, la instalación le habrá creado el icono  en el escritorio y un grupo de programas. Mediante dicho icono o seleccionando *scilab-6.1.0 Desktop* en el grupo de programas estará en condiciones de utilizar el entorno Scilab.

En la figura 1.1 puede observar el aspecto que presenta la Consola de Scilab al arrancar de cualquiera de esas dos formas. La consola muestra información sobre la versión instalada de Scilab. Este aspecto cambiará conforme vayamos utilizando el entorno o conforme lo vayamos adaptando a nuestros gustos. Centramos por ahora la atención en la parte central donde se sitúa la **Ventana de Comandos**, mostrando que se ha cargado el entorno inicial y presenta el indicador (*prompt* en inglés) `-->` característico de que el entorno está listo para su utilización. Un click de ratón sobre dicha ventana provocará que aparezca el cursor con el símbolo “|” parpadeando inmediatamente después de `-->`, indicativo de que el entorno está preparado para que el usuario teclee una expresión en la línea de comandos y para que ordene su evaluación.

En la figura 1.2 se muestra la reacción de la ventana de comandos después de que el usuario haya compuesto la expresión $x=4;5+2*x-3$ y haya pulsado la tecla ↵ (ENTER). Se trata de una expresión compuesta por dos expresiones, con la primera $x=4$ se consigue asignar a la variable x el valor 4 y con la segunda $5+2*x-3$ se solicita el resultado de la operación $5+2x-3$ para ese valor de x . Observe que es imprescindible teclear el carácter “*” entre el 2 y la x para indicar que queremos realizar el producto de ambos. El carácter “;” se ha empleado como separador de ambas expresiones pero

además para indicar a Scilab que no muestre en ventana el valor asignado a la variable x . El resultado de la operación es el número decimal 10., que Scilab ha asignado a una variable por defecto llamada *ans* (del inglés *answer*, respuesta en español).

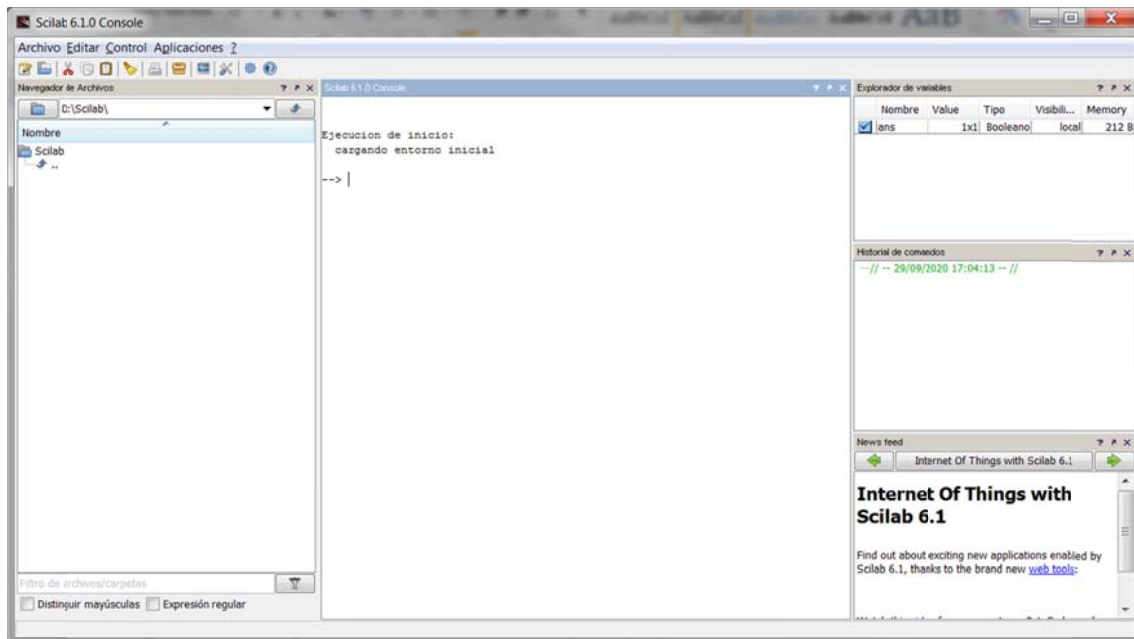


Figura 1.1 Aspecto inicial de la Consola de Scilab.

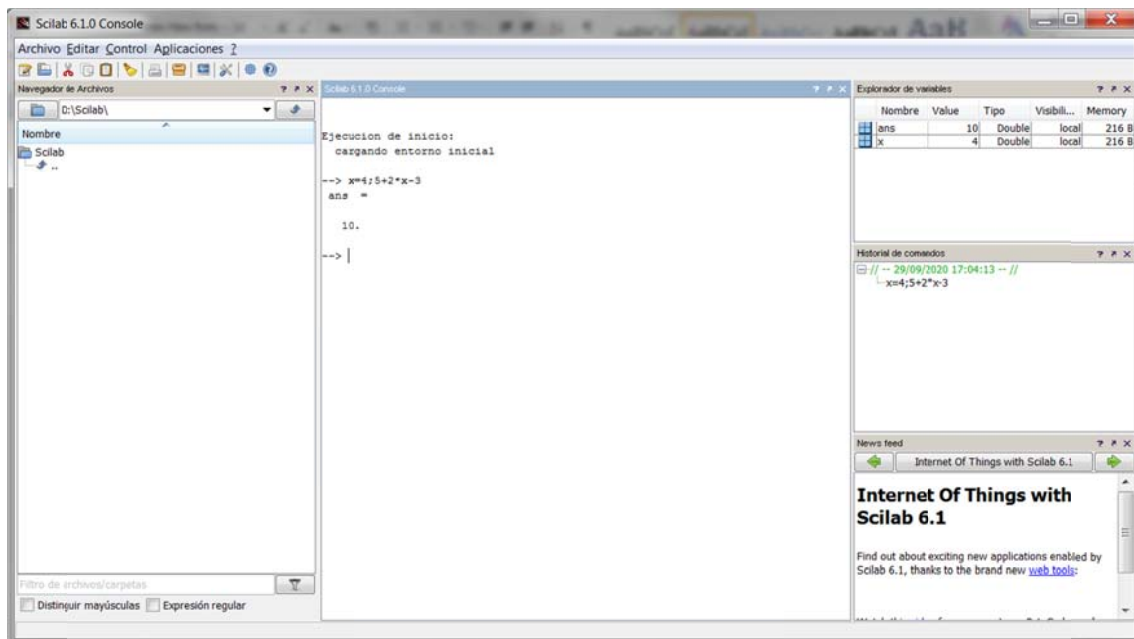


Figura 1.2 Reacción de la Consola de Scilab después de componer la expresión $x=4;5+2*x-3$ en la ventana de comandos y pulsar la tecla ↵ (ENTER).

En la parte derecha de la consola puede observar que Scilab también ha actualizado la información correspondiente al *Explorador de Variables* y al *Historial de Comandos*. El primero nos indica las dos variables *ans* y *x* actualmente en memoria, sus respectivos valores, el tipo, etc... El segundo irá recogiendo todas las sentencias que vayamos ejecutando en la ventana de comandos, por ahora muestra la única que hemos ejecutado, junto con el día y la hora en la que comenzamos la sesión.

Esta forma de trabajar se conoce como el modo calculadora o el modo línea de comandos y es habitual en cualquiera de los lenguajes de programación intérpretes. El entorno está ejecutando el típico ciclo secuencial de tres estados (entrada de datos, cálculo, presentación de resultado). Durante la entrada de datos el entorno está analizando los caracteres que llegan desde el teclado y va componiendo la expresión, este estado se abandona en el caso de Scilab cuando el entorno recibe la tecla ↵, momento que es interpretado como la orden de proceder al cálculo y a la posterior presentación del resultado. Finalizada esta presentación, el entorno muestra de nuevo el indicador --> y comienza un nuevo ciclo.

En el estado de entrada de datos, el usuario también puede hacer uso de los cursores. Por ejemplo, tecleando una vez el cursor hacia arriba “↑” puede recuperar la línea tecleada anteriormente y en pulsaciones sucesivas o dejándolo pulsado podrá recuperar cualquiera de las líneas tecleadas durante una sesión.

Ejercicio 1.1 a) Arranque Scilab y reproduzca el resultado de la figura 1.2. b) Pruebe a recuperar la línea ejecutada anteriormente, para modificar el “;” por una “,” y comparar por observación con lo ocurrido anteriormente.

El modo línea de comandos no es la única forma de trabajar en Scilab, ni siquiera es la más conveniente, cuando el número de expresiones va en aumento es mucho mejor trabajar sobre el editor **SciNotes**. Se accede a él a través del primer icono (Ejecutar SciNotes) de la barra de herramientas. La figura 1.3 muestra el aspecto inicial (en blanco) de la ventana de edición. Este editor es algo más que un editor de texto, pues al mismo tiempo que vamos tecleando nos auxilia en la sintaxis, enumera las líneas y mediante un código de colores distingue los distintos tipos de elementos que contempla el lenguaje de programación de Scilab. En el apartado 2.2 del documento **Introduction to Scilab** puede consultar más información sobre el editor.

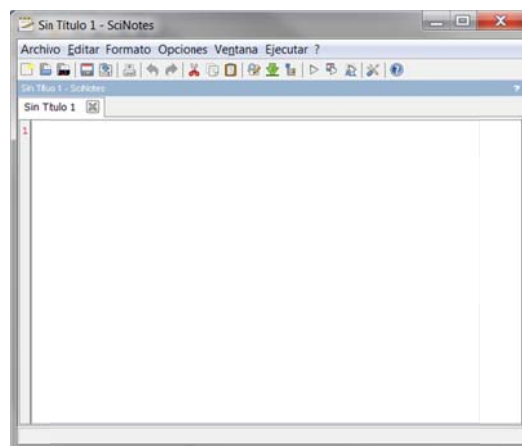


Figura 1.3 Aspecto inicial de la ventana de edición de Scilab.

En la figura 1.4 se muestra las dos líneas editadas y guardadas en el archivo “Prueba1Tema1.sce”, este paso previo es necesario para poderlas ejecutar. La extensión *sce* no es necesario ponerla pues es la extensión reservada por Scilab para los archivos ejecutables, creados con el editor. La ejecución se solicita con el último icono (Ejecutar)

de la barra de herramientas, y si aún no hemos guardado las líneas en el archivo, Scilab nos lo recordará con la siguiente mensajería

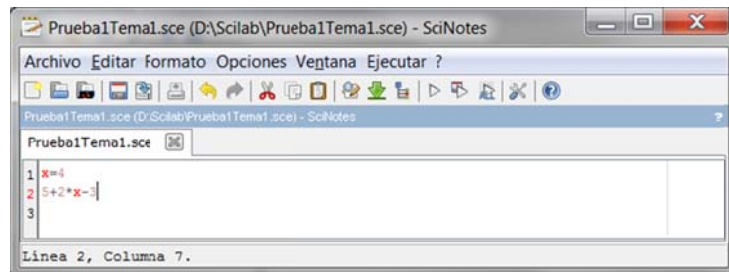
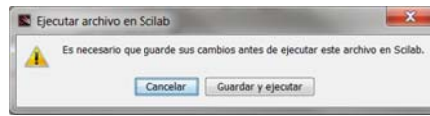


Figura 1.4 Primeros pasos en la edición del archivo “Prueba1Tema1.sce”.

Aparentemente no ha pasado nada pero sí ha pasado, la ventana de comandos habrá mostrado el mensaje `exec('D:\Scilab\Prueba1Tema1.sce', -1)`. Con la ejecución del archivo “Prueba1Tema1.sce” habremos conseguido lo mismo que al evaluar la expresión $x=4; 5+2*x-3$ en la ventana de comandos; la variable x existe y tiene el valor 4. Luego x ha pasado a ser una variable del usuario y tiene en ese momento el valor 4. Pero lo que no existe es el resultado de la operación. Para que la ventana de comandos muestre resultados numéricos es necesario incluir alguna instrucción como `disp` en el archivo. La figura 1.5 muestra una ampliación del primer archivo de prueba con estos fines. Las cuatro últimas líneas se utilizan para presentar los correspondientes mensajes y valores de las variables en la ventana de comandos de Scilab. Pero como además la línea de comandos en Scilab puede hacer la función de entrada de datos para un cálculo, se ha incluido una primera línea que hace uso de la instrucción `input` para provocar un diálogo con el usuario a través de la consola. En este caso concreto se le muestra el mensaje “introduzca el valor de x =”. La segunda instrucción realiza el cálculo y lo asigna a la variable y . En la figura 1.6 se puede observar el resultado que provoca en la ventana de comandos la ejecución del mismo archivo pero con el nuevo contenido. Si volvemos a ejecutar el archivo no tenemos más que teclear otro valor de x para conseguir un nuevo resultado para y . Este archivo puede considerarse como un primer ejemplo de programación en Scilab.

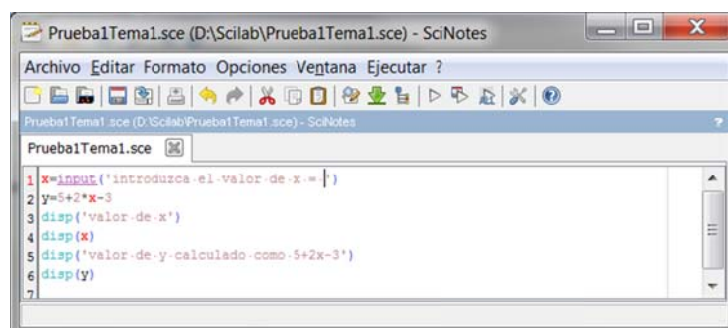


Figura 1.5 Contenido final del archivo “Prueba1Tema1.sce”.

Ejercicio 1.2 a) *Acceda al editor de Scilab y reproduzca el contenido final del archivo “Prueba1Tema1.sce”.* b) *Pruebe a ejecutarlo varias veces para distintos valores de x.*

```
--> exec('D:\Scilab\Prueba1Tema1.sce', -1)
introduzca el valor de x = 5

"valor de x"

5.

"valor de y calculado como 5+2x-3"

12.
```

Figura 1.6 Resultado, mostrado por la ventana de comandos, como consecuencia de ejecutar el archivo de la figura 1.5.

Llegado este punto es conveniente echar una ojeada al *Navegador de Archivos*, situado en la parte izquierda de la consola de Scilab. Con su ayuda podrá bucear por los contenidos de su equipo, en busca de directorios y/o de archivos donde tendrá almacenados los programas y funciones que vaya desarrollando. En especial los archivos con la extensión *sce*, editables y ejecutables en Scilab.

En cualquier aplicación y sobre todo en estos programas matemáticos de tanto alcance es muy importante tener una buena documentación de ayuda e incluso el auxilio de otros usuarios. Al manual de ayuda de Scilab se puede acceder de varias formas desde la consola; tecleando *help* en la ventana de comandos, o mediante el último icono (Navegador de ayuda) de la barra de herramientas. El manual se presenta en ventana aparte como un hipertexto sobre el que se puede navegar fácilmente y también se pueden realizar búsquedas. Los apartados 1.4 a 1.6 del documento **Introduction to Scilab** describe las distintas formas de solicitar ayuda y de conseguir documentación en Scilab. Explore también las opciones de ayuda que le ofrece el icono ? de la Consola de Scilab.

Su aprendizaje sobre Scilab continuará en el apartado 1.4 de este primer tema y en el tema 2 pero si desea avanzar un poco más puede hacerlo consultando los apartados 2 y 3 del documento **Introduction to Scilab**.

1.3 PRIMEROS PASOS CON Wxmaxima

Consiga el archivo correspondiente a su sistema operativo para instalar la versión de Maxima recomendada por el equipo docente. El apartado **Descargar** de la página web oficial de Maxima <http://maxima.sourceforge.net/es/> le ayudará a localizarlo y le facilitará instrucciones para su instalación. También le puede ser de gran ayuda el apartado 2.1 del documento **Primeros pasos en Maxima**.

Durante el proceso de instalación de Maxima se recomienda elegir la instalación por defecto. Si ha elegido la versión 5.43.0 para Windows, la que ha utilizado el equipo docente al elaborar estos apuntes, la instalación le habrá creado un grupo de programas

y un icono para acceder al entorno gráfico *wxMaxima (GUI for Maxima)*. Mediante dicho icono estará en condiciones de utilizar el entorno wxMaxima.

En la figura 1.7 puede observar el aspecto que presenta la ventana principal de wxMaxima. Este aspecto cambiará conforme vayamos utilizando el entorno o conforme lo vayamos adaptando a nuestros gustos. Se observa un mensaje de bienvenida y otro mensaje indicando que el entorno está preparado para que el usuario teclee una expresión y ordene su evaluación. También se observa una línea horizontal continua que actúa de cursor y servirá de separación entre las distintas expresiones. En la figura 1.8 se muestra la reacción de Maxima después de haber compuesto la expresión $5+2\cdot x-3$ y haber pulsado simultáneamente las teclas \Uparrow (SHIFT) y \Downarrow (ENTER). La sintaxis utilizada en esta expresión coincide con la empleada en Scilab, pero observe que ahora no ha sido necesario asignar valor numérico a la variable x . El resultado de la operación es otra expresión, la simplificada de la anterior. Pero además wxMaxima la presenta al usuario tal como él la escribiría, es decir como $2x+2$.

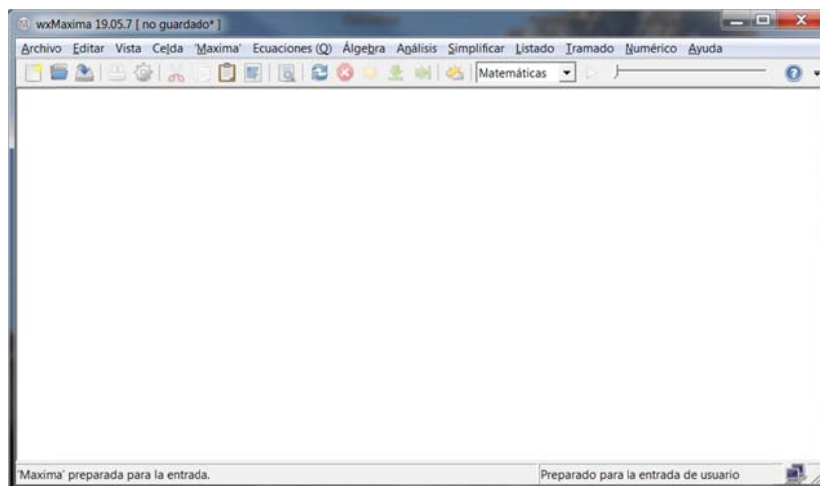


Figura 1.7 Aspecto inicial de la ventana principal de wxMaxima.

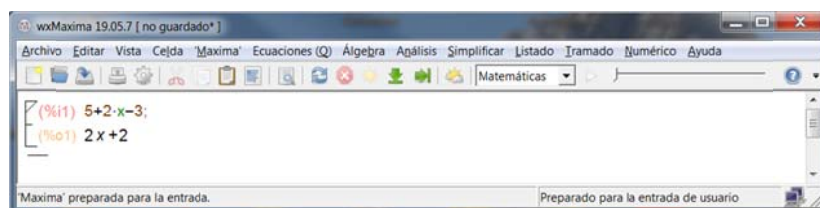


Figura 1.8 Reacción de Maxima después de componer la expresión $5+2\cdot x-3$ y pulsar simultáneamente las teclas \Uparrow (SHIFT) y \Downarrow (ENTER).

El conjunto formato por la expresión de entrada, etiquetada por Maxima como *%i1* por ser la primera entrada (del inglés *input*), y por la expresión resultado, etiquetada por Maxima como *%o1* por ser la primera salida (del inglés *output*), pertenecen a una misma celda en el entorno de wxMaxima. Las líneas a la izquierda delimitan el contenido de esta primera celda.

Además de la variable x tecleada por el usuario, wxMaxima considera a todas las entradas y resultados como variables, de manera que *%i1* y *%o1* están a partir de ese

momento a disposición del usuario como variables. Por tanto se nos presentan ahora diversas opciones para hacer que este resultado sea una nueva variable, la variable y . Estas opciones están recogidas en la figura 1.9. Al ejecutar $y:\%i1$ estamos asignando a la variable y la entrada $\%i1$, es decir la expresión inicial tal como fue tecleada. Al ejecutar $y:\%o1$ estamos asignando a la variable y la salida $\%o1$, es decir la expresión simplificada de la expresión inicial. Y al ejecutar $y:5+2*x-3$ estamos asignando a la variable y la expresión inicial, tecleándola de nuevo. En los tres casos, la y entre paréntesis de la izquierda indica la existencia de una variable de nombre y cuyo valor es $2x+2$.

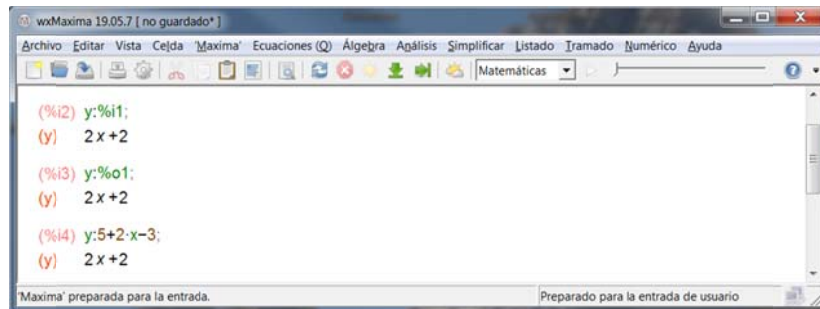


Figura 1.9 Formas alternativas de conseguir que la variable y tome el valor de la expresión $5+2x-3$.

Pero la mejor forma, observe la figura 1.10, es crear la variable y como una función de la variable x , con la siguiente sintaxis $y(x):=5+2*x-3$, pues en estas condiciones al solicitar $y(4)$ en wxMaxima nos ahorramos de tener que evaluar la expresión compuesta $x:4; y:5+2*x-3$. Además de esta forma podremos particularizar el cálculo de y para cualquier valor de x , por ejemplo $y(0.5)$, etc...

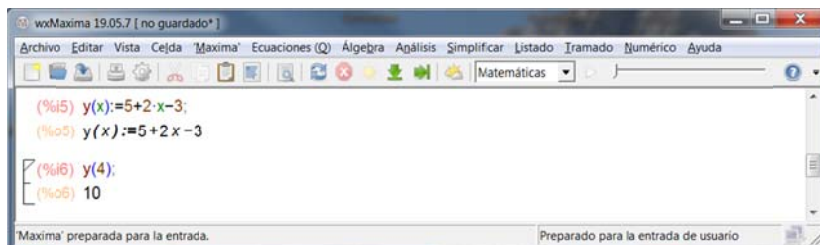


Figura 1.10 Definición de la variable y como función de x , dada por la expresión $5+2x-3$. Y la evaluación de la función para $x=4$.

El texto “no guardado*” en la ventana de wxMaxima nos está indicando que las expresiones tecleadas hasta ahora no se han guardado en ningún archivo. Es recomendable que procedamos a guardar estas seis entradas en un archivo, para ello se puede utilizar la opción Archivo o el segundo icono (Guardar documento) de la barra de herramientas. En la figura 1.11 puede comprobarse que ahora sí tenemos las seis entradas guardadas en el archivo “primerospasosconwxMaxima.wxmx”. La extensión *wxmx* no es necesaria ponerla pues es la extensión reservada por wxMaxima para los archivos de sesión del tipo *Documento completo*.

La figura 1.11 también nos sirve para poner de manifiesto que el concepto de celda es muy importante en wxMaxima. Tras teclear las seis entradas se han creado seis celdas, cada una tiene una expresión de entrada y una expresión de salida. Utilizando los cursores y/o el ratón el usuario se puede situar en cualquiera de las celdas para:

modificarla y/o volverla a evaluar, eliminarla, ocultarla, etc ... Por tanto la ventana principal de wxMaxima juega el doble papel, el de consola y el de ventana de edición.

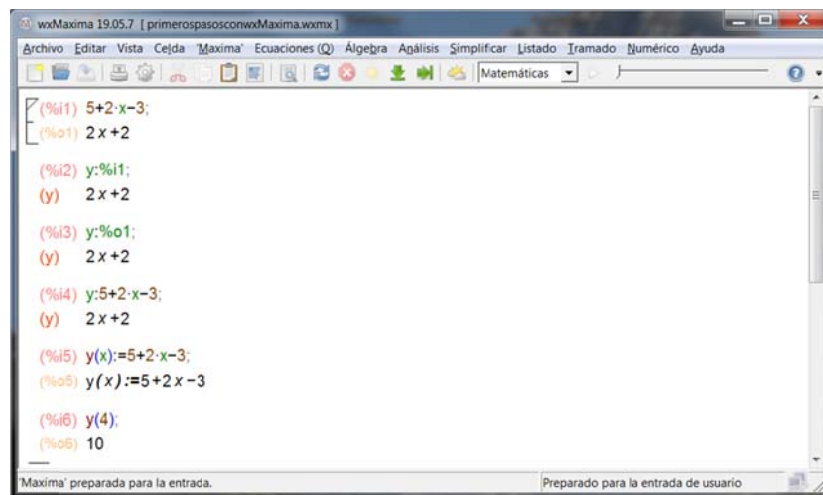


Figura 1.11 Aspecto de la ventana principal después de guardar las seis entradas en el archivo “primerospasosconwxMaxima.wxmx”.

La figura 1.12 muestra el aspecto que presenta la ventana de wxMaxima si abrimos, con la opción Archivo o el primer icono (Abrir documento) de la barra de herramientas, el archivo “primerospasosconwxMaxima.wxmx”. Se pueden ver las seis expresiones previamente guardadas, enumeradas y evaluadas. El símbolo \rightarrow es el indicativo empleado por wxMaxima para marcar la parte de las celdas que son editables, el resto son resultados de la anterior ejecución. En esta situación el usuario puede ir modificando y evaluando las celdas una a una, incorporar nuevas celdas, evaluar el conjunto de celdas seleccionadas, o evaluarlas todas. Dicha funcionalidad se consigue con el ratón, sus botones, y a través de la opción Celda. Tenga en cuenta que wxMaxima tiene memoria de las entradas y salidas previas, por lo que al ejecutar de nuevo las celdas se generarán nuevas entradas y salidas con la numeración que les corresponda. En la opción ‘Maxima’ tiene posibilidad de Reiniciar el programa y de Limpiar la memoria para evitar que siga creciendo la información asociada a la sesión de wxMaxima.

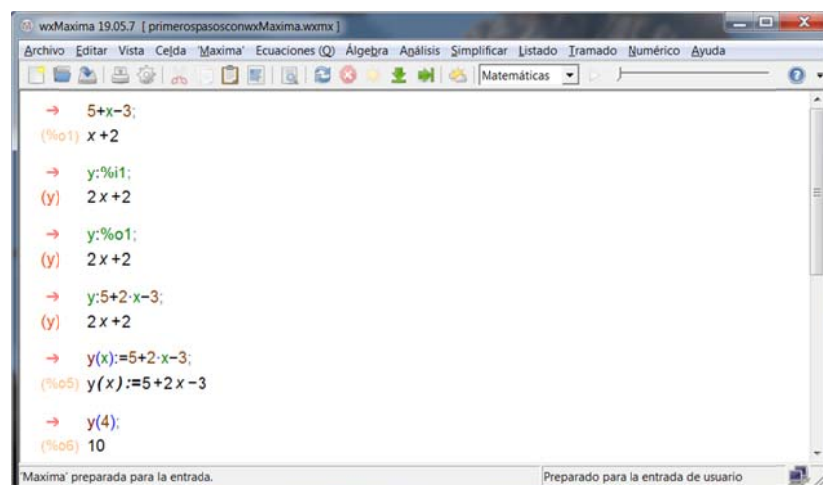


Figura 1.12 Aspecto de la ventana principal después de abrir el archivo “primerospasosconwxMaxima.wxmx”.

Los contenidos de una sesión en wxMaxima también se pueden guardar con la extensión wxm, extensión reservada por wxMaxima para los archivos de sesión del tipo *Entrada, leíble sin carga*. La diferencia es que en este caso no se almacenan los resultados de ejecución de las celdas, únicamente se almacenan los contenidos evaluables. Por ejemplo, si la sesión anterior la hubiéramos almacenado en el archivo “primerospasosconwxMaxima.wxm”, al abrirlo wxMaxima nos mostraría únicamente las partes con \rightarrow de las seis celdas de la figura 1.12.

Ejercicio 1.3 a) Arranque wxMaxima, teclee sucesivamente las mismas seis entradas de las figuras 1.8 a 1.10, y compruebe los resultados. B) Guarde las seis entradas en un archivo con el nombre y extensión que crea más oportuno, y continúe la sesión en wxMaxima evaluando la función y para otros valores de x .

Las seis expresiones de la figura 1.12 acaban con el carácter “;” para indicar a wxMaxima que muestre en ventana el resultado de evaluar la expresión. Si se desea que no muestre el resultado de una evaluación, la expresión debe ir terminada con el carácter \$. Las seis expresiones de la figura 1.12 son expresiones simples, si la expresión es compuesta, wxMaxima habrá generado tantas salidas como expresiones haya evaluado, y las habrá enumerado correlativamente, pero sólo mostrará las que se le soliciten. Pruebe a evaluar la siguiente expresión compuesta

$$y(x):=5+2*x-3\$y(1);y(2);y(3)\$y(4);$$

Al manual de usuario de Maxima se puede acceder de varias formas; desde la opción *Maxima documentation* creada durante la instalación o mediante el último icono (Mostrar la ayuda de Maxima) de la barra de herramientas. La primera opción abre un documento en PDF. Pero la segunda opción es más interesante porque presenta el manual en ventana aparte como un hipertexto sobre el que se puede navegar fácilmente y también se pueden realizar búsquedas.

Explore también las distintas formas de solicitar ayuda y de conseguir documentación que le ofrece la opción *Ayuda* de la ventana principal de wxMaxima. En especial es **importante que eche una ojeada al manual de wxMaxima para que se familiarice con el entorno gráfico, para que conozca todas sus funcionalidades y para que pueda personalizarlo según sus necesidades.**

Su aprendizaje sobre Maxima continuará en el apartado 1.4 de este primer tema y en el tema 2 pero si desea avanzar un poco más puede hacerlo consultando el apartado 2.2 del documento **Primeros pasos en Maxima**.

1.4 TRES EJEMPLOS REPRESENTATIVOS DEL ALCANCE DE LA ASIGNATURA

Los tres ejemplos siguientes, que se pueden abordar con lápiz y papel (sin ayuda de herramienta informática ni calculadora), son representativos del alcance de la asignatura y pretenden motivar el uso de Scilab y Maxima en la resolución de problemas matemáticos. De ahí que se comenten las tres posibles aproximaciones a la solución: mediante lápiz y papel, la numérica con Scilab y la simbólica con Maxima.

El estudiante encontrará en el curso virtual los correspondientes archivos Ejemplo*Tema1.sce y Ejemplo*Tema1.wxm para que pueda reproducir los ejemplos sin necesidad de programarlos, pues ya tendrá tiempo a lo largo de los temas 2 y 3 para familiarizarse con la sintaxis y las funciones de ambos programas.

Ejemplo 1. Se quiere determinar el área comprendida entre la recta $f(x) = 2x + 2$ y la parábola $g(x) = (x-1)^2 + 1$.

Ejemplo 2. Las siguientes funciones del tiempo describen respectivamente la posición instantánea horizontal y vertical que ocupa un proyectil lanzado con una velocidad inicial v_0 y un ángulo α respecto al suelo.

$$x(t) = v_0 (\cos \alpha) t \quad (1.1)$$

$$y(t) = v_0 (\sin \alpha) t - \frac{1}{2} g t^2 \quad (1.2)$$

Se desea comparar gráficamente las trayectorias que describe el proyectil en el plano XY para dos situaciones concretas de tiro: el tiro rasante, con ángulo $\alpha = \alpha_1 < 45^\circ$, y el tiro por elevación, con ángulo $\alpha = \alpha_2 > 45^\circ$, tal que $\alpha_1 + \alpha_2 = 90^\circ$. En ambos casos se puede suponer que la velocidad inicial v_0 fue de 10m/s. Además para facilitar los cálculos se recomienda utilizar un valor de 10m/s^2 para la aceleración g de la gravedad, $\alpha_1 = 30^\circ$ y $\alpha_2 = 60^\circ$.

Ejemplo 3: Se quiere descomponer el número 15 en tres sumandos naturales distintos cuyo producto sea máximo.

1.4.1 Soluciones al ejemplo 1

Si abordamos el problema mediante lápiz y papel podremos optar por:

1º) Determinar si ambas funciones se cortan, para ello desarrollamos la segunda función

$$g(x) = (x-1)^2 + 1 = x^2 - 2x + 1 + 1 = x^2 - 2x + 2$$

e igualamos ambas expresiones

$$f(x) = g(x) \Rightarrow 2x + 2 = x^2 - 2x + 2 \Rightarrow x^2 - 4x = 0 \Rightarrow x=0, x=4$$

Las soluciones son los puntos de corte.

2º) Calcular el área solicitada como la integral definida siguiente

$$\begin{aligned} \text{Área solicitada} &= \int_0^4 (f(x) - g(x)) dx = \int_0^4 (2x + 2 - (x^2 - 2x + 2)) dx = \\ &= \int_0^4 (4x - x^2) dx = 2x^2 - \frac{x^3}{3} \Big|_0^4 = 2 \cdot 4^2 - \frac{4^3}{3} = \frac{96 - 64}{3} = \frac{32}{3} \end{aligned}$$

Observación: el cálculo anterior está presuponiendo que la primera función (la de la recta) toma un valor superior al de la segunda en todo el rango $[0, 4]$. Hay muchas formas de comprobarlo, la más fácil es la comprobación gráfica, pero esto lo dejamos para cuando planteemos la solución con los programas matemáticos.

A continuación trataremos de reproducir los pasos anteriores en Scilab. La forma más fácil de determinar si ambas funciones se cortan pasa por la representación gráfica de ambas funciones. Pero aún así no tenemos seguridad de qué rango hay que utilizar en la variable independiente. Tras varios intentos hemos decidido utilizar el rango $[-2, 5]$, en ese caso, tecleando las sentencias

```
x=-2:.01:5;
fx=2*x+2;
gx=(x-1).^2+1;
plot(x,fx,x,gx)
```

habremos obtenido la representación gráfica de la figura 1.13, donde es fácil observar que los cortes de ambas funciones se producen en $x=0$ y $x=4$. También es fácil observar que la recta está por encima de la parábola en todo el rango $[0, 4]$. Con la primera sentencia generamos un conjunto finito de valores para la variable independiente x , que van desde -2 a 5 y están separados por una centésima. Con la segunda y la tercera sentencia generamos un conjunto finito de valores para las variables dependientes fx y gx usando las expresiones que definen las dos funciones del ejemplo. Y con la cuarta sentencia solicitamos la representación gráfica de los dos conjuntos de valores dependientes respecto al conjunto de valores independientes.

El área solicitada la calculamos como la integral definida entre los dos puntos de corte, haciendo uso de la función *integrate* tal como sigue. El resultado presentado por Scilab será el valor numérico 10.666667, que coincide con el resultado obtenido mediante lápiz y papel con un redondeo al entero más próximo en la sexta cifra decimal.

```
integrate('2*x+2-((x-1)^2+1)', 'x', 0, 4)
```

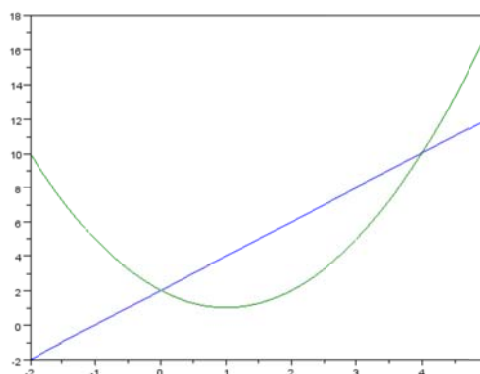


Figura 1.13 Representación gráfica en Scilab de las funciones $f(x)=2x+2$ y $g(x)=(x-1)^2+1$ del ejemplo 1.

A continuación se muestran las cuatro sentencias tecleadas y ejecutadas en Maxima para resolver el problema planteado en el ejemplo 1. Las sentencias se diferencian del resto porque están representadas como *inputs* de Maxima. La primera y la segunda sentencia sirven para asignar a las variables fx y gx respectivamente las expresiones que definen las dos funciones del ejemplo. En la tercera, que sirve para determinar los puntos de

correspondiente al punto de corte, se hace uso de la función *solve*. Y en la cuarta, que sirve para calcular el área como la integral definida, se hace uso de la función *integrate*. En Maxima también podemos generar una representación gráfica similar a la generada en Scilab. Por ejemplo, encadenando una quinta sentencia como la mostrada en la figura 1.14.

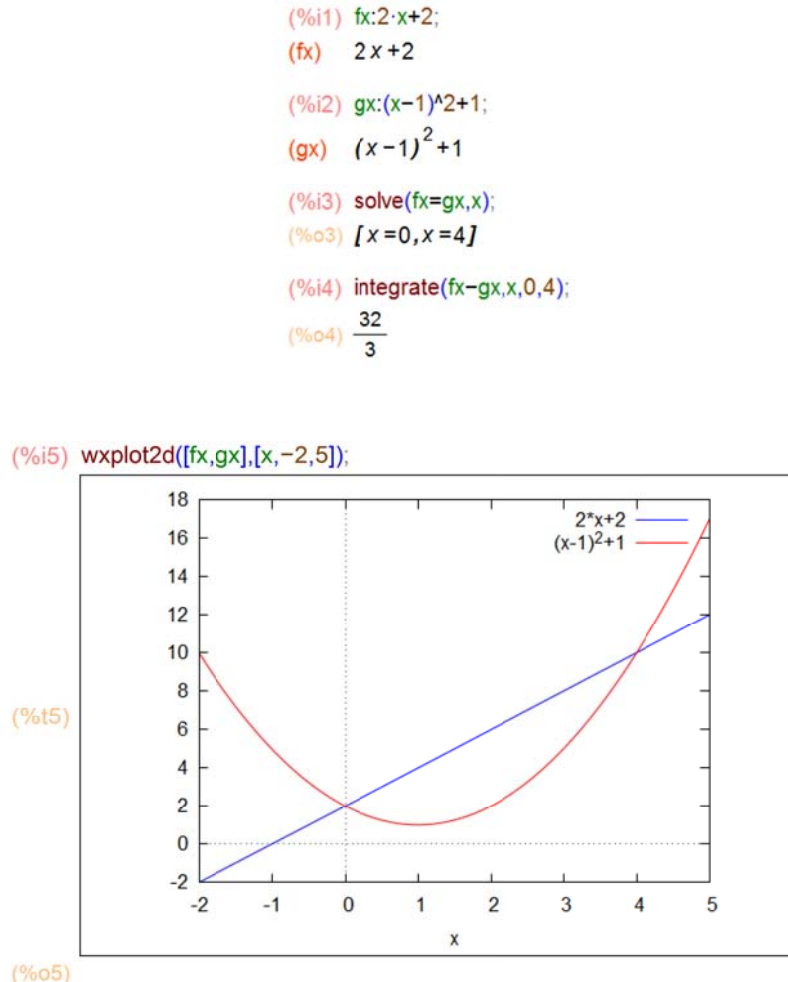


Figura 1.14 Representación gráfica en wxMaxima de las funciones del ejemplo 1.

El problema planteado en este ejemplo se ha podido resolver en los dos entornos. En Scilab hemos necesitado el apoyo gráfico, mientras que en Maxima ese apoyo no hubiera hecho falta y sólo hemos acudido a él para comprobar el resultado.

1.4.2 Soluciones al ejemplo 2

Este tipo de lanzamiento recibe el nombre de tiro parabólico porque el proyectil siempre describe una parábola de eje vertical en el plano XY. La demostración se puede hacer eliminando la variable t , el tiempo, de las ecuaciones (1.1) y (1.2)

$$x(t) = v_o (\cos \alpha) t \Rightarrow t = \frac{x}{v_o \cos \alpha}$$

$$y = v_o \operatorname{sen} \alpha \frac{x}{v_o \cos \alpha} - \frac{1}{2} g \left(\frac{x}{v_o \cos \alpha} \right)^2$$

$$y = (\operatorname{tg} \alpha) x - \frac{g}{2 v_o^2 \cos^2 \alpha} x^2 \quad (1.3)$$

Pues se llega a una función $y(x)$ de la forma $y = a x^2 + bx + c$, que representa una parábola cuyo eje es vertical. Resolviendo (1.3) para $y=0$ se obtiene el punto del eje X donde el proyectil impacta sobre el suelo. Este valor, que nos da el alcance del tiro, se representa de la forma siguiente:

$$d = \frac{2 v_o^2 \cos \alpha \operatorname{sen} \alpha}{g} = \frac{v_o^2 \operatorname{sen} 2\alpha}{g} \quad (1.4)$$

Luego está claro que, para una velocidad inicial dada, el máximo alcance se obtiene con un ángulo $\alpha=45^\circ$. Por tanto, para ángulos menores y para ángulos mayores a 45° se obtienen menores alcances y concretamente se demuestra que para ángulos complementarios se consigue el mismo alcance. El tiro rasante, el que se hace con un ángulo $\alpha_1 < 45^\circ$, tendrá un alcance d_1 dado por

$$d_1 = \frac{v_o^2 \operatorname{sen} 2\alpha_1}{g}$$

y el tiro por elevación, con ángulo $\alpha_2 = 90^\circ - \alpha_1 > 45^\circ$, tendrá un alcance d_2 dado por

$$d_2 = \frac{v_o^2 \operatorname{sen} 2\alpha_2}{g} = \frac{v_o^2 \operatorname{sen} 2(90^\circ - \alpha_1)}{g} = \frac{v_o^2 \operatorname{sen} (180^\circ - 2\alpha_1)}{g} = \frac{v_o^2 \operatorname{sen} 2\alpha_1}{g} = d_1$$

Por tanto si representamos las trayectorias descritas por los tiros rasantes y por elevación tendremos dos parábolas como las representadas en la figura 1.15. Se confirma que el proyectil impacta en la misma posición pero lógicamente tarda menos tiempo en hacerlo cuando el tiro es rasante. Para demostrarlo sustituimos el valor x por d en la expresión de t , y denominamos a ese valor el instante de impacto con el suelo:

$$t_{\text{impacto}} = \frac{d}{v_o \cos \alpha} = \frac{2 v_o^2 \cos \alpha \operatorname{sen} \alpha}{g v_o \cos \alpha} = \frac{2 v_o \operatorname{sen} \alpha}{g} \quad (1.5)$$

Queda demostrado que el instante de impacto depende directamente del seno del ángulo de tiro, cuanto mayor es el ángulo más tarda el proyectil en impactar con el suelo.

Otra característica del tiro parabólico es la altura máxima que alcanza el proyectil. Si aplicamos la propiedad de simetría de la trayectoria está claro que la altura máxima se presenta cuando $x=d/2$, o visto de otra forma cuando ha transcurrido la mitad del tiempo. Por tanto sustituyendo t por $t_{\text{impacto}}/2$ en la expresión de $y(t)$ se obtiene:

$$y_{\max} = v_o (\sin \alpha) \frac{v_o \sin \alpha}{g} - \frac{1}{2} g \left(\frac{v_o \sin \alpha}{g} \right)^2 = \frac{v_o^2 \sin^2 \alpha}{2g} \quad (1.6)$$

Con esta expresión queda demostrado que la altura máxima que alcanza el proyectil depende cuadráticamente del seno del ángulo de tiro, cuanto mayor es el ángulo mayor es la altura máxima alcanzada por el proyectil.

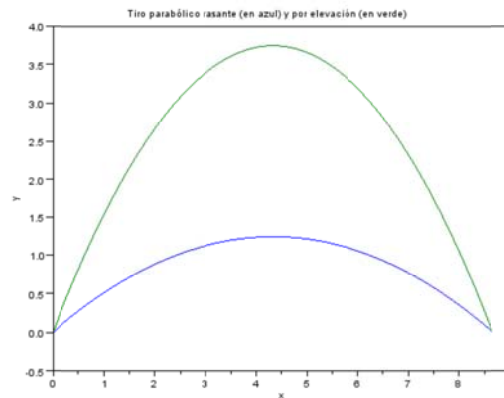


Figura 1.15 Trayectorias de los tiros parabólico rasante y por elevación del ejemplo 2.

Para facilitar el estudio del tiro parabólico en Scilab hemos decidido utilizar el concepto de función. Tanto la x , que representa la distancia recorrida en la horizontal, como la y , que representa la altura instantánea, son variables dependientes. Con las dos primeras sentencias definimos las coordenadas del tiro parabólico.

```
function x=distancia(vo, alfa, t),x=vo*cos(alfa)*t,endfunction
function y=altura(vo, alfa, g, t),y=vo*sin(alfa)*t-g*t^2/2,endfunction
```

A continuación asignamos valores concretos para tratar de reproducir las trayectorias del tiro rasante y del tiro por elevación. Pero aún así no tenemos seguridad de qué rango hay que utilizar en la otra variable independiente, el tiempo.

```
vo=10;
g=10;
alfa1=%pi/6;
alfa2=%pi/3;
```

En primer lugar decidimos utilizar el rango [0 1.5], de manera que tecleando las sentencias conseguimos trazar las trayectorias de la figura 1.16.

```
t=0:0.01:1.5;
x1=distancia(vo,alfa1,t);
y1=altura(vo,alfa1,g,t);
x2=distancia(vo,alfa2,t);
y2=altura(vo,alfa2,g,t);

plot(x1,y1,x2,y2)
```

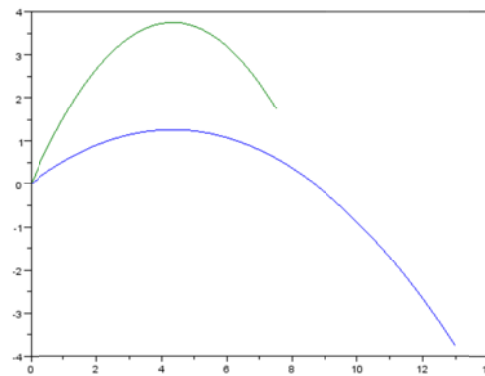


Figura 1.16 Representación de los tiros parabólicos del ejemplo 2 utilizando los mismos valores entre 0 y 1.5 para la variable tiempo.

El trazo azul nos indica que el proyectil por tiro rasante ya habría llegado en ese tiempo al suelo, y el trazo en verde nos indica que el proyectil por tiro en elevación no habría llegado todavía, de ahí que sea preferible generar una nueva gráfica con valores de tiempo distintos para cada trayectoria. Esto es lo que se ha hecho con las siguientes instrucciones, donde hemos aprovechado los conocimientos del desarrollo analítico para fijar el valor máximo de $t1$ y de $t2$ y donde hemos incorporado las etiquetas a los ejes y un título a la gráfica. La gráfica resultante ya se utilizó anteriormente en la figura 1.15 para complementar a la solución mediante lápiz y papel.

```
t1=0:0.01:1;
x1=distancia(vo,alfa1,t1);
y1=altura(vo,alfa1,g,t1);
t2=0:0.01:sqrt(3);
x2=distancia(vo,alfa2,t2);
y2=altura(vo,alfa2,g,t2);

plot(x1,y1,x2,y2)

title('Tiro parabólico rasante (en verde) y por elevación (en azul)')
xlabel('x')
ylabel('y')
```

Respecto al uso de Maxima en este ejemplo, las dos primeras sentencias han servido para representar el tiro parabólico como una función $y(x)$. La tercera sentencia nos permite determinar los valores de x en los que $y=0$. El resultado coincide con el obtenido en la solución analítica. Las siguientes sentencias se han utilizado para asignar valores concretos al tiro por rasante y al tiro por elevación. Y la última para representar ambas trayectorias en el mismo gráfico, en ventana aparte, como muestra la figura 1.17.

```
(%i1) t:x/(vo*cos(alfa));
(t)      x
      cos(alfa) vo

(%i2) y:vo*sin(alfa)*t-g*t^2/2;
(y)      sin(alfa) x      g x^2
      cos(alfa)      2 cos(alfa)^2 vo^2

(%i3) solve(y,x);
(%o3) [x = -\frac{2 \cos(alfa) \sin(alfa) vo^2}{g}, x = 0]

(%i7) vo:10;g:10;alfa1:%pi/3;alfa2:%pi/6;
(vo) 10
(g) 10
(alfa1) \frac{\pi}{3}
(alfa2) \frac{\pi}{6}

(%i8) d:(2*cos(alfa1)*sin(alfa1)*vo^2)/g;
(d) 5\sqrt{3}

(%i9) y1:(sin(alfa1)*x)/cos(alfa1)-(g*x^2)/(2*cos(alfa1)^2*vo^2);
(y1) \sqrt{3}x - \frac{x^2}{5}

(%i10) y2:(sin(alfa2)*x)/cos(alfa2)-(g*x^2)/(2*cos(alfa2)^2*vo^2);
(y2) \frac{x}{\sqrt{3}} - \frac{x^2}{15}

(%i11) plot2d([y1,y2],[x,0,d])$
```

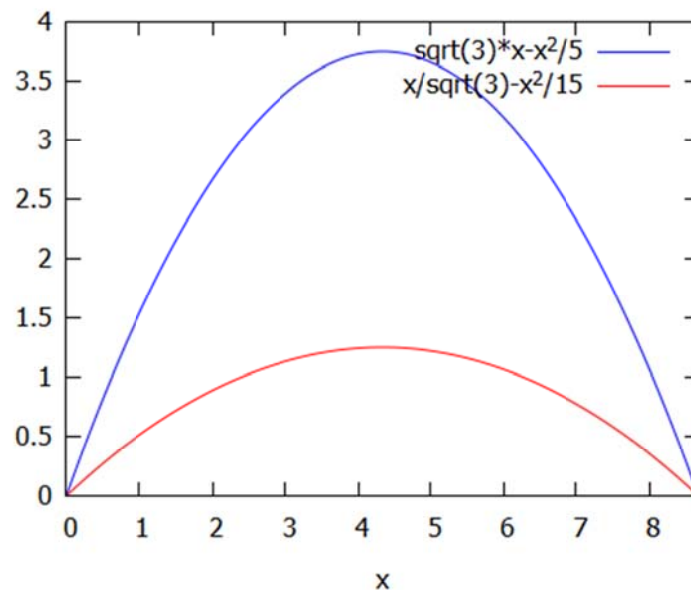


Figura 1.17 Trayectorias en wxMaxima de los tiros parabólicos rasante y por elevación del ejemplo 2.

El problema planteado en este ejemplo obligaba a utilizar la representación gráfica. Pero mientras en Scilab hemos tenido que probar varios vectores de tiempo para las trayectorias, en Maxima teníamos desde el primer momento un conocimiento exacto del tiempo necesario en cada trayectoria.

1.4.3 Soluciones al ejemplo 3

El valor máximo del producto $a b c$ de tres números naturales a, b, c se obtendrá cuando los tres números tengan sus valores máximos posibles. En este caso concreto los números no pueden ser cualesquiera sino que tienen que sumar 15, entonces podíamos pensar que la solución al problema es $a=b=c=5$, pero como además el enunciado nos dice que los números tienen que ser distintos entre sí. Podemos afirmar que el resultado es $a=5+1=6$, $b=5$ y $c=5-1=4$.

La solución también se podría haber afrontado de otra forma, comenzando por la búsqueda exhaustiva de las ternas de números naturales distintos entre sí que suman 15, para luego seleccionar aquella que da lugar al producto máximo entre sus elementos. A continuación se indican las 12 posibles ternas y en el orden en el que se han determinado

Ternas_naturales_que_suman_15 = {(1,2,12), (1,3,11), (1,4,10), (1,5,9), (1,6,8), (2,3,10), (2,4,9), (2,5,8), (2,6,7), (3,4,8), (3,5,7), (4,5,6)}

Como era de esperar están ordenadas en orden creciente del producto de sus elementos, como muestran los siguientes valores

Producto_elementos_ternas_naturales_que_suman_15 = {24, 33, 40, 45, 48, 60, 72, 80, 84, 96, 105, 120}

Y por tanto, la solución como era de esperar es la terna (4,5,6).

La solución más fácil en Scilab puede consistir en anidar tres *bucles for* para generar todas las posibles ternas de números naturales comprendidos entre 1 y 12, pero quedarse sólo con aquellas cuyos elementos sean distintos entre sí y sumen 15. Para luego elegir de entre ellas la terna cuyo producto de elementos sea máximo. Las siguientes sentencias valen para ello.

```
i=0;
for a=1:1:12
    for b=1:1:12
        for c=1:1:12
            if (a+b+c)==15 then
                if a<>b & a<>c & b<>c then
                    i=i+1;
                    ternassuman15(i,:)= [a b c];
                end
            end
        end
    end
end
y=prod(ternassuman15,2)
[maxy,i]=max(y)
resultado=ternassuman15(i,:)
disp(resultado)
```

En la búsqueda anterior de la solución se han generado $12^3=1728$ posibles ternas, aunque únicamente se han asignado 72 ternas para evaluar el producto de sus elementos, aquellas que cumplen la condición de que sus elementos son distintos entre sí y suman 15, pero aún así estamos por encima de las únicas 12 ternas posibles. Ello se debe a que con esas sentencias hemos generado ternas que únicamente se diferencian de otras por la posición de los términos. Basta incorporar otras condiciones adicionales en los bucles para que se generen un menor número de ternas, por ejemplo podemos forzar a que todas las ternas cumplan que $b > a$ y que $c = 15 - a - b$. Las sentencias que permiten ese tipo de búsqueda son las siguientes:

```
i=0;
for a=1:1:12
    for b=a+1:1:13-a
        c=15-a-b
        if c<>a & c<>b
            i=i+1;
            ternassuman15(i,:)=[a b c];
        end
    end
end
y=prod(ternassuman15,2)
[maxy,i]=max(y)
resultado=ternassuman15(i,:)
disp(resultado)
```

El número de ternas asignadas en este caso se reduce a 31, todavía por encima de las 12 posibles. Otra posible solución algo mejor que la anterior, porque se ejecutaría en menos tiempo y requeriría menos almacenamiento de variables, pasaría por no guardar memoria de todas las ternas que suman 15 sino únicamente de una terna, aquella cuyo producto sea mayor que el de la última terna generada. Esto se consigue con las siguientes sentencias, donde además comprobamos que únicamente se asignan 11 resultados intermedios hasta llegar al resultado definitivo. En definitiva únicamente se exploran las 12 ternas posibles y en el mismo orden en el que se determinaron para la solución mediante lápiz y papel.

```
i=0;
producto=1;
for a=1:1:12
    for b=a+1:1:13-a
        c=15-a-b
        if c<>a & c<>b & a*b*c>producto
            producto=a*b*c
            i=i+1;
            ternassuman15(i,:)=[a b c];
            resultado=[a b c]
        end
    end
end
disp(ternassuman15)
disp(producto)
disp(resultado)
```

Con el siguiente resultado en la consola

1.	2.	12.
1.	3.	11.
1.	4.	10.

1.	5.	9.
1.	6.	8.
2.	3.	10.
2.	4.	9.
2.	5.	8.
2.	6.	7.
3.	4.	8.
3.	5.	7.
4.	5.	6.

120.

4.	5.	6.
----	----	----

Puesto que el problema planteado en este ejemplo se ha abordado de forma numérica por búsqueda exhaustiva, no hay gran diferencia entre utilizar Scilab o Maxima. En Maxima nos vamos a limitar a reproducir la última de las soluciones, presentando las ternas y el producto resultante según se van generando, con el siguiente resultado en la ventana principal de wxMaxima.

```
(%i1) producto:1;
(producto) 1

(%i2) for a:1 thru 12 do
for b:a+1 thru 13-a do
(c:15-a-b, if c#a and c#b and a-b-c>producto then
(producto:a-b-c, display([a,b,c],producto)));
[a,b,c]=[1,2,12]
producto=24
[a,b,c]=[1,3,11]
producto=33
[a,b,c]=[1,4,10]
producto=40
[a,b,c]=[1,5,9]
producto=45
[a,b,c]=[1,6,8]
producto=48
[a,b,c]=[2,3,10]
producto=60
[a,b,c]=[2,4,9]
producto=72
[a,b,c]=[2,5,8]
producto=80
[a,b,c]=[2,6,7]
producto=84
[a,b,c]=[3,4,8]
producto=96
[a,b,c]=[3,5,7]
producto=105
[a,b,c]=[4,5,6]
producto=120
(%o2) done
```


1.4 LECTURA COMPLEMENTARIA

El estudiante debe complementar este tema con la lectura de:

Apartados 1, 2 y 3 del documento **Introduction to Scilab**

Apartados 1 y 2 del documento **Primeros pasos en Maxima**

Sin olvidar la ayuda disponible en Scilab y en wxMaxima.

También es conveniente que consulte los términos “List of computer algebra systems”, “Sistema algebraico computacional” y “Análisis numérico o cálculo numérico” en la correspondientes versiones de la enciclopedia libre [Wikipedia](#). Donde encontrará referencias y enlaces que le permitirán ampliar sus conocimientos sobre el tema.