Apéndice A

Apéndice

A.1 Introducción a los CAS

Los sistemas algebraicos computacionales o sistemas de álgebra computacional (CAS: *Computer Algebra System*) son sistemas o calculadoras avanzadas, que permiten realizar operaciones de manera simbólica. Esto significa que el computador puede efectuar operaciones con ecuaciones y fórmulas simbólicamente, es decir, $a + b = c$ se interpreta como la suma de variables y no como la suma de números previamente asignados.

Estos sistemas permiten operar de manera exacta con símbolos que representan objetos matemáticos tales como:

- Números (Enteros, racionales, reales, complejos...)
- Polinómios, Funciones Racionales, Sistemas de Ecuaciones.
- Grupos, Anillos, Algebras . . .

A diferencia de los sistemas tradicionales de computación numérica:

- FORTRAN, Basic, C, C++, Java $=>$ Precisión fija (Punto Flotante)

Otra característica principal radica en el hecho de que son interactivos (interpretados o ejecutados al momento de proveer una instrucción).

Los CAS se pueden clasificar en dos grandes grupos:[1]

- Sistemas de Propósito Especial (Creados para hacer cálculos en un área específica): FORM, GAP, CAMAL, SHEEP, STENSOR, LiE, KANT.
- Sistemas de Propósito General (¡Especies de navajas suizas!): Axiom, Derive, Reduce, Maple, MatLab, Mathematica, Maxima, MuPAD. Recientemente, lenguajes como Python comienzan a incorporar bibliotecas que permiten generar formas de cálculo simbólico[2] que ofrecen una perspectiva interesante para integrar ambientes algebráicos-numéricos-visuales.

Los CAS modernos de propósito general son ambientes completamente integrados de computación para la investigación y la educación conformados por:

- Interfaz gráfica (worksheet) o ambiente interactivo:
- Procesador de texto, de fórmulas y de gráficas.
- Con salidas en Latex, RTF, HTML, FORTRAN y C; o hyperlinks a otros documentos.
- Manuales en línea.

[1] https://en.wikipedia.org/wiki/List_of_computer_algebra_systems

[2] http://docs.sympy.org/

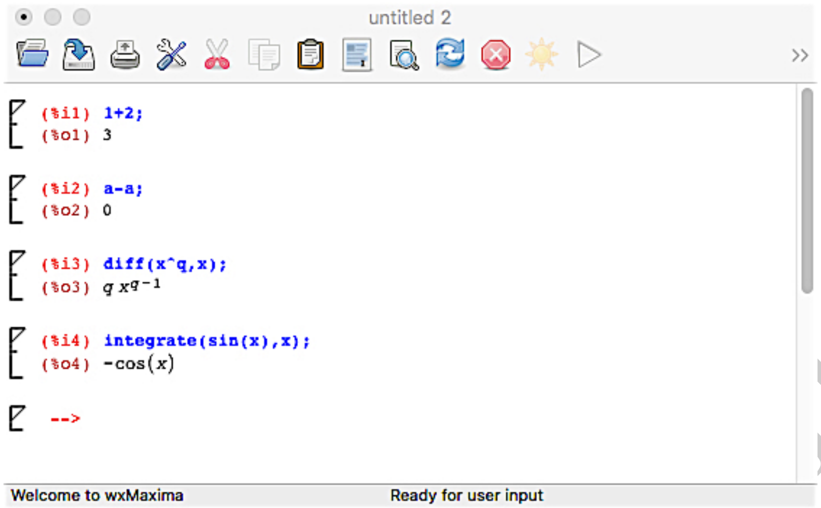


Figura A.1: Ventana gráfica de vxMaxima

- Enlaces a otros programas y bibliotecas
- Capacidades para cálculo numérico
- Capacidades para visualización, con salidas gráficas en diferentes formatos: PostScript, GIF,JPG, . . .
- Pensado para usuarios no especializados en computación

La principal ventaja de estos programas radica en la enorme capacidad para realizar cálculos algebraicos largos y tediosos. Por ejemplo, se puede demostrar que la función:

$$f = \frac{\operatorname{sen}\left(\frac{nz\sqrt{x^2+y^2+z^2}}{\sqrt{y^2+z^2}}\right)}{\sqrt{x^2+y^2+z^2}}$$

es solución de la ecuación diferencial:

$$\frac{\partial^4 f}{\partial x^4} + \frac{\partial^4 f}{\partial y^2 x^2} + \frac{\partial^4 f}{\partial z^2 x^2} + n^2\left[\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right] = 0$$

y la realización de éste cálculo le puede tomar a un PC estándar un tiempo de CPU relativamente corto:

```
tiempo de cpu = 1,065 seg
```

Un ejemplo de un CAS es **Maxima** que básicamente consta de una hoja de trabajo (worksheet) que es una interfaz tipo procesador de textos. Las hojas de trabajo constan de dos modos básicos de funcionamiento: el modo texto y el modo cálculo. **Maxima** opera en la celda de cálculos de la manera siguiente:

```
Input (Instrucción de entrada) ;
Output (respuesta del programa)
```

Existe la posibilidad de introducir textos de la misma manera que en un procesador de textos estándar y de generar gráficas en 2D y 3D a través de los respectivos comandos.

La interacción con la hoja de trabajo se hace a través de lo que denominamos la celda del Input, que aparece señalado por un aviso de espera o PROMPT

```
[ -->
```

El código fuente de **Maxima**, código de software libre, puede ser compilado sobre diferentes sistemas operativos: Windows, Linux, MacOS X y Android, y puede obtenerse en: `http://sourceforge.net/projects/wxmaxima` o en `http://andrejv.github.io/wxmaxima`, con la respectiva documentación.

Utilizaremos la versión gráfica **wxMaxima** para los fines pedagógicos del desarrollo de estas notas. También existe una versión que funciona sólo en modo texto para ser ejecutada en una consola o terminal.

En la Figura A.1 se puede apreciar el despliegue de una hoja de cálculo con algunos instrucciones sencillas, notemos que cada instrucción (en azul) termina con un punto y coma, de esta manera se le dice al programa la finalización del comando a ejecutar. Luego de escribir la instrucción y presionar la tecla Enter ↰ el Output aparecerá a continuación en color negro.

A.2 Maxima: Sintaxis básica

Es necesario familiarizarse con los comandos básicos del programa, para ello iremos desarrollando algunos cálculos sencillos a manera de conocer la filosofía de cómo funciona **Maxima**, y durante el transcurso de este curso iremos haciendo un despliegue de las aplicaciones del programa para cálculos más específicos.

(%i1) 3!+3^5;
(%o1) 249

Veamos la diferencia entre el igual $=$ y los dos puntos :

(%i2) a=b+c;
(%o2) c+b

(%i3) a;
(%o3) a

(%i4) a:b+c;
(%o4) c+b

(%i5) a;
(%o5) c+b

Al usar "$=$"no asignamos a la variable lo que está del lado derecho mientras que con ":"si le asignamos el objeto a la nueva variable.

Los cálculos se pueden hacer tanto con números enteros como en Punto Flotante:

(%i6) 15 + 5^(50);
(%o6) 88817841970012523233890533447265640

(%i7) 15.0 + 5^(50);
(%o7) $8{,}881784197001253 \times 10^{34}$

Pero el énfasis radica en los cálculos exactos:

(%i8) cos(%pi/12)^2 + `log`(2/3+5)/7;
(%o8) $\cos^2\left(\frac{\pi}{12}\right) + \frac{\log\left(\frac{17}{3}\right)}{7}$

y si queremos el valor numérico podemos escribir

(%i9) float(%);
(%o9) 1,180812852661949

Aquí hemos hecho uso de varios símbolos nuevos. Las constantes matemáticas en **Maxima** se escriben de la siguiente manera:

- La unidad imaginaria i: $\%i$
- El número π: $\%pi$.
- El símbolo ∞: inf.
- El número e: $\%e$.

En cuanto a los logaritmos:

- $e^x =$ exp(x).
- log(x): la función logaritmo en base e
- log(x)/log(b) el logaritmo de x en base b.

Para las funciones trigonométricas es lo estándar: sin(x), asin(x), cos(x), acos(x), tan(x), atan(x), csc(x), sec(x), cot(x).

También hemos utilizado $\%$ para introducir la última salida en la siguiente instrucción, veamos nuevamente como funciona, pero ahora pondremos al final del comando el símbolo $ en lugar de ; para decirle al programa que ejecute la instrucción sin escribir la salida.

(%i10) 3^ 12$

(%i11) %;
(%o11) 531441

Veamos otro ejemplo:

(%i12) alpha;
(%o12) α

(%i13) % + sqrt(beta);
(%o13) $\sqrt{\beta}+\alpha$

(%i14) %o12 + 2*gamma^ 2;
(%o14) $2\,\gamma^2+\alpha$

Es importante tener cuidado a la hora de poner los paréntesis en las expresiones:

(%i15) 1+2*3^ 2;
(%o15) 19

(%i16) (1+2)*3^ 2;
(%o16) 27

El volumen de un cilindro $V = \pi(\text{radio})^2 \times \text{altura}$

(%i17) radio: 5$

(%i18) altura: 50$

(%i19) area: %pi*radio^ 2;
(%o19) $25\,\pi$

(%i20) volumen: area*altura;
(%o20) $1250\,\pi$

Para limpiar la memoria del programa de todas las variables utilizadas se puede usar el comando **kill**(all) (Existen otras opciones que veremos más adelante). Esta es una manera de reiniciar la hoja de trabajo.

(%i21) volumen;
(%o21) $1250\,\pi$

(%i22) kill(all);
(%o22) $done$

(%i23) volumen;
(%o23) $volumen$

A.3 Cálculos elementales

Se puede definir, evaluar y derivar funciones abstractas utilizando $:=$ como se muestra a continuación

(%i1) f(x,y):=exp(x^2+y^2)/(x-y);
(%o1) $f(x,y) := \dfrac{exp\left(y^2+x^2\right)}{x-y}$

(%i2) f(2,3);
(%o2) $-e^{13}$

(%i3) %,numer;
(%o3) $-442413{,}3920089205$

(%i4) f(alpha^(1/2),beta^(1/2));
(%o4) $\dfrac{e^{\beta+\alpha}}{\sqrt{\alpha}-\sqrt{\beta}}$

Derivando respecto a x y y;

(%i5) diff(f(x,y),x) + diff(f(x,y),y);
(%o5) $\dfrac{2\,y\,e^{y^2+x^2}}{x-y}+\dfrac{2\,x\,e^{y^2+x^2}}{x-y}$

Aquí es bueno acotar que una expresión NO es una función:

(%i6) f(x):=3*sin(x+1)+2*sqrt(x);
(%o6) $f(x) := 3\,\sin(x+1)+2\,\sqrt{x}$

(%i7) F:3*sin(x+1)+2*sqrt(x);
(%o7) $3\,\sin(x+1)+2\,\sqrt{x}$

(%i8) f(8); F(8);
(%o8) $3\,\sin 9+2^{\frac{5}{2}}$
(%o9) $\left(3\,\sin(x+1)+2\,\sqrt{x}\right)(8)$

Por lo tanto, F es únicamente una expresión que no puede ser evaluada como la función f. Pero se le puede dar la vuelta para evaluarla con **ev**()

(%i10) ev(F,x=8);
(%o10) $3\,\sin 9+2^{\frac{5}{2}}$

Consideremos los siguientes cálculos básicos:

(%i11) kill(all)$
(%i1) sigma(x):=2*x/sqrt(x^2+1);
(%o1) $\sigma(x) := \dfrac{2\,x}{\sqrt{x^2+1}}$

La primera derivada:

(%i2) diff(sigma(x),x);

(%o2) $\dfrac{2}{\sqrt{x^2+1}} - \dfrac{2\,x^2}{(x^2+1)^{\frac{3}{2}}}$

La cuarta derivada:

(%i3) diff(sigma(x),x,4);

(%o3) $\dfrac{90\,x}{(x^2+1)^{\frac{5}{2}}} - \dfrac{300\,x^3}{(x^2+1)^{\frac{7}{2}}} + \dfrac{210\,x^5}{(x^2+1)^{\frac{9}{2}}}$

Si queremos reutilizar la derivada para definir una nueva función, en este caso la función derivada, lo podemos hacer utilizando dos apóstrofos ” (No es la doble comilla)

(%i4) dsigma(x):=” %o2;

(%o4) $dsigma\,(x) := \dfrac{2}{\sqrt{x^2+1}} - \dfrac{2\,x^2}{(x^2+1)^{\frac{3}{2}}}$

(%i5) dsigma(2);

(%o5) $\dfrac{2}{5^{\frac{3}{2}}}$

(%i6) integrate(sigma(x),x);

(%o6) $2\sqrt{x^2+1}$

La misma integral, pero definida para x entre 0 y 1.

(%i7) integrate(sigma(x),x,0,1);

(%o7) $2\left(\sqrt{2}-1\right)$

Límites:

(%i8) limit(sigma(x),x,1/2);

(%o8) $\dfrac{2}{\sqrt{5}}$

(%i9) limit(sigma(x),x,inf);

(%o9) 2

Sumatorias:

(%i10) sum(sigma(i),i,0,6);

(%o10) $\dfrac{12}{\sqrt{37}} + \dfrac{10}{\sqrt{26}} + \dfrac{8}{\sqrt{17}} + \dfrac{6}{\sqrt{10}} + \dfrac{4}{\sqrt{5}} + \sqrt{2}$

Podemos calcular series de Taylor, digamos, alrededor de $x = 1$ y hasta orden 4.

(%i11) taylor(sigma(x),x,1,4);

(%o11) $\dfrac{-1}{128}\left(5\,2^{\frac{1}{2}}\right)(x-1)^4 + \dfrac{1}{16}\left(3\,2^{\frac{1}{2}}\right)(x-1)^3 + \dfrac{-1}{8}\left(3\,2^{\frac{1}{2}}\right)(x-1)^2 + \dfrac{1}{2}\,2^{\frac{1}{2}}\,(x-1) + 2^{\frac{1}{2}} + \cdots$

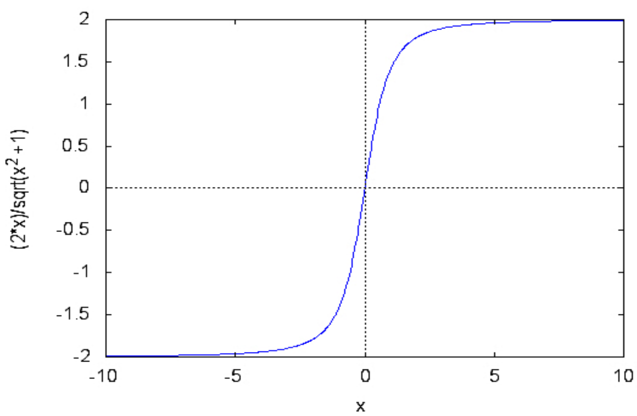
Al rededor de $x = 0$ es más simple todo:

(%i12) taylor(sigma(x),x,0,6);

(%o12) $\frac{3}{4}\,x^5 + (-1)\;x^3 + 2\,x + \cdots$

Y por supuesto, también podemos hacer una gráfica de la función. Para ello utilizaremos el comando **wxplot2d** que nos generará un gráfico incrustado dentro de la misma hoja de trabajo.

(%i13) wxplot2d(sigma(x),[x,-10,10]);
(%o13)



Los cálculos anteriores se pueden repetir para que queden de una manera más elegante usando una camilla, esto hará que no se efectúe la evaluación de las operaciones.

(%i14) 'diff(sigma(x),x)=diff(sigma(x),x);

(%o14) $\frac{d}{d\,x}\left(\frac{2\,x}{\sqrt{x^2+1}}\right) = \frac{2}{\sqrt{x^2+1}} - \frac{2\,x^2}{(x^2+1)^{\frac{3}{2}}}$

(%i15) 'integrate(sigma(x),x,0,1)=integrate(sigma(x),x,0,1);

(%o15) $2\int_0^1 \frac{x}{\sqrt{x^2+1}}\,dx = 2\left(\sqrt{2}-1\right)$

(%i16) 'limit(sigma(x),x,inf)=limit(sigma(x),x,inf);

(%o16) $2\left(\lim_{x\to\infty}\frac{x}{\sqrt{x^2+1}}\right) = 2$

(%i17) 'sum(sigma(i),i,0,6)=sum(sigma(i),i,0,6);

(%o17) $2\sum_{i=0}^{6}\frac{i}{\sqrt{i^2+1}} = \frac{12}{\sqrt{37}} + \frac{10}{\sqrt{26}} + \frac{8}{\sqrt{17}} + \frac{6}{\sqrt{10}} + \frac{4}{\sqrt{5}} + \sqrt{2}$

Anteriormente mencionamos que uno de las ventajas de los programas de manipulación simbólica es la gran capacidad de llevar a cabo cálculos largos y tediosos, veamos entonces como se hace para demostrar que la función antes mencionada:

(%i18) f(x,y,z):=sin(n*z*sqrt(x^2+y^2+z^2)/sqrt(y^2+z^2))/sqrt(x^2+y^2+z^2);

(%o18) $f\,(x,y,z) := \frac{\sin\left(\frac{n\,z\,\sqrt{z^2+y^2+x^2}}{\sqrt{z^2+y^2}}\right)}{\sqrt{z^2+y^2+x^2}}$

es solución de la ecuación diferencial

$$\frac{\partial^4 f}{\partial x^4}+\frac{\partial^4 f}{\partial y^2 x^2}+\frac{\partial^4 f}{\partial z^2 x^2}+n^2\left[\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}\right]=0$$

(%i19) diff(f(x,y,z),x,4)+diff(diff(f(x,y,z),x,2),y,2)+diff(diff(f(x,y,z),x,2),z,
2)+n^2*(diff(f(x,y,z),x,2)+diff(f(x,y,z),y,2));

(%o19) « Expression too long to display! »

Aquí **Maxima** no hace un despliegue en la pantalla de los cálculos porque la expresión matemática es muy larga. Existen opciones para que muestre en pantalla los que nos interese que iremos viendo más adelante.

Necesitamos entonces decirle al programa que la expresión anterior sea simplificada, es decir, que minimice la expresión a su valor más simple. Para simplificar expresiones que contienen radicales, exponenciales o logaritmos es conveniente utilizar el comando **ratsimp**. También existe la opción **fullratsimp**

(%i20) ratsimp(%);

(%o20) 0

En la mayoría de los casos **Maxima** no factoriza ni desarrolla automáticamente las expresiones, por lo tanto, debemos indicarle al programa que haga las respectivas simplificaciones. Veamos un ejemplo con polinomios:

(%i21) kill(all)$

(%i1) p:(x+2)*(x-1);

(%o1) $(x-1)\,(x+2)$

(%i2) q:(x-3)^2;

(%o2) $(x-3)^2$

(%i3) p-q;

(%o3) $(x-1)\,(x+2)-(x-3)^2$

(%i4) expand(p-q);

(%o4) $7\,x-11$

(%i5) expand(p/q);

(%o5) $\frac{x^2}{x^2-6\,x+9}+\frac{x}{x^2-6\,x+9}-\frac{2}{x^2-6\,x+9}$

Si queremos dividir usando fracciones simples podemos hacer lo siguiente:

(%i6) partfrac(p/q,x);

(%o6) $\frac{7}{x-3}+\frac{10}{(x-3)^2}+1$

Las funciones **logexpand** y **radexpand** permiten controlar si queremos simplificar logaritmos y radicales cuando contienen productos. Veamos:

(%i7) log(x*y);

(%o7) $\log(x\,y)$

(%i8) sqrt(x*y);

(%o8) $\sqrt{x\,y}$

(%i9) sqrt(x^2);

(%o9) $|x|$

(%i10) radexpand:all$ logexpand:all$

(%i11) log(x*y); sqrt(x*y); sqrt(x^2);
(%o11) $\log y+\log x$

(%o12) $\sqrt{x}\,\sqrt{y}$

(%o13) x

Lo inverso a la expansión de expresiones es la factorización:

(%i14) factor(200);
(%o14) $2^3\,5^2$

(%i15) factor(x^2+x-2);
(%o15) $(x-1)\,(x+2)$

(%i16) p:x^3-1;
(%o16) x^3-1

(%i17) factor(%);
(%o17) $(x-1)\,\left(x^2+x+1\right)$

La evaluación de expresiones se realiza de la manera siguiente

(%i18) ev(p,x=8);
(%o18) 511

O también

(%i19) p,x=%pi;
(%o19) π^3-1

(%i20) ev(x+(x+y)^2-3*(x+y)^3,x+y=t);
(%o20) $x-3\,t^3+t^2$

En esta guía rápida de **Maxima**, veamos el uso de uno de los comandos más comunes de estos programas, y que tiene que ver con la solución de ecuaciones.

(%i21) ecu: 3*x^2+2*x+x^3-x^2=2*x^2;
(%o21) $x^3+2\,x^2+2\,x=2\,x^2$

(%i22) sol: solve(ecu,x);
(%o22) $\left[x=-\sqrt{2}\,\%i, x=\sqrt{2}\,\%i, x=0\right]$

Recordemos que $\%i$ es la notación para el imaginario i.
Si necesitamos aislar una de las soluciones usamos el comando **rhs** (de right-hand side):

(%i23) rhs(part(sol,1)); rhs(part(sol,2));
(%o23) $-\sqrt{2}\,\%i$

(%o24) $\sqrt{2}\,\%i$

Para un sistema de ecuaciones, digamos, dos ecuaciones con dos incógnitas:

(%i25) ecu1: x^2+y^2=1;
(%o25) $y^2 + x^2 = 1$

(%i26) ecu2:(x-2)^2+(y-1)^2=4;
(%o26) $(y-1)^2 + (x-2)^2 = 4$

(%i27) solve([ecu1,ecu2],[x,y]);
(%o27) $\left[\left[x = \frac{4}{5}, y = -\frac{3}{5}\right], [x = 0, y = 1]\right]$

Cuando el sistema no tiene solución **Maxima** responde de la siguiente manera

(%i28) solve([x+y=0,x+y=1],[x,y]);
(%o28) [ ]

En el caso de un sistema que tiene más incógnitas que ecuaciones el programa utiliza los símbolos $\%r_1$, $\%r_2$... para indicar los parámetros arbitrarios

(%i29) solve([x+y+z=9,x-y=2*z],[x,y,z]);
(%o29) $\left[\left[x = \frac{\%r_1 + 9}{2}, y = -\frac{3\,\%r_1 - 9}{2}, z = \%r_1\right]\right]$

Cuando el sistema esta conformado por ecuaciones lineales podemos usar el comando **linsolve**, en lugar de **solve**. En este caso el algoritmo es más eficiente en cuanto al uso de los recursos del computador.

(%i30) ecus:[x+y+z+w=1,x-y+z-w=-2,x+y-w=0];
(%o30) $[z + y + x + w = 1, z - y + x - w = -2, y + x - w = 0]$

(%i31) linsolve(ecus,[x,y,z]);
(%o31) $\left[x = \frac{4\,w - 3}{2}, y = -\frac{2\,w - 3}{2}, z = 1 - 2\,w\right]$

En el caso de polinómios de orden superior el resultado estará dado de forma aproximada:

(%i32) ec: x^7+x^5-x^3+x-2;
(%o32) $x^7 + x^5 - x^3 + x - 2$

(%i33) allroots(ec);
(%o33) $[x = 0{,}766414088337633\,\%i + 0{,}5507199727230275\,,$
$x = 0{,}5507199727230275 - 0{,}766414088337633\,\%i\,,$
$x = 0{,}4922671445862202\,\%i - 0{,}9637112977011089\,,$
$x = -0{,}4922671445862202\,\%i - 0{,}9637112977011089\,,$
$x = 1{,}381985877916414\,\%i - 0{,}08700867502191806\,,$
$x = -1{,}381985877916414\,\%i - 0{,}08700867502191806\,,$
$x = 0{,}9999999999999988\,]$

(%i34) realroots(ec);
(%o34) $[x = 1]$

Otro tipo de ecuaciones a resolver son las ecuaciones diferenciales. Veamos como funciona **ode2** con la ecuaciones diferenciales ordinarias

(%i35) ecd: (2*x+1)*'diff(y,x)+y*(x-1)=0;

(%o35) $(2\,x+1)\,\left(\frac{d}{d\,x}\,y\right)+(x-1)\,y=0$

(%i36) ode2(ecd,y,x);

(%o36) $y=\%c\,e^{\frac{3\,\log(2\,x+1)}{4}-\frac{x}{2}}$

Por ser una ecuación diferencial de primer orden debe aparecer una constante en la solución. La constante aquí es denotada por "$\%c$".

(%i37) ecd2:'diff(y,x,2)-3*'diff(y,x)+2*y=x;

(%o37) $\frac{d^2}{d\,x^2}\,y-3\,\left(\frac{d}{d\,x}\,y\right)+2\,y=x$

(%i38) ode2(ecd2,y,x);

(%o38) $y=\%k_1\,e^{2\,x}+\%k_2\,e^{x}+\frac{2\,x+3}{4}$

Para el caso de que se tengan condiciones iniciales utilizamos **ic2** para indicar las condiciones

(%i39) ic2(%o38,x=0,y=0,diff(y,x)=1);

(%o39) $y=\frac{5\,e^{2\,x}}{4}-2\,e^{x}+\frac{2\,x+3}{4}$

Y para valores de contorno **bc2**

(%i40) bc2(%o38,x=0,y=0,x=1,y=0);

(%o40) $y=\frac{(3\,e-5)\,e^{2\,x}}{4\,e^2-4\,e}-\frac{\left(3\,e^2-5\right)\,e^{x}}{4\,e^2-4\,e}+\frac{2\,x+3}{4}$

Existe el comando **desolve** para resolver también ecuaciones o sistemas de ecuaciones diferenciales ordinarias lineales utilizando transformadas de Laplace. Trabaja de manera parecida a **ode2** pero se necesita especificar la dependencia de las funciones con las variables independientes.

(%i41) ecd3:'diff(y(x),x,2)-y(x)=2*x;

(%o41) $\frac{d^2}{d\,x^2}\,y\,(x)-y\,(x)=2\,x$

(%i42) desolve(ecd3,[y(x)]);

(%o42) $y\,(x)=\frac{e^{x}\,\left(\left.\frac{d}{d\,x}\,y\,(x)\right|_{x=0}+y\,(0)+2\right)}{2}-\frac{e^{-x}\,\left(\left.\frac{d}{d\,x}\,y\,(x)\right|_{x=0}-y\,(0)+2\right)}{2}-2\,x$

Si tenemos condiciones iniciales en $x=0$ entonces podemos escribir:

(%i43) atvalue(y(x),x=0,1); atvalue(diff(y(x),x),x=0,2);

(%o43) 1

(%o44) 2

(%i45) desolve(ecd3,[y(x)]);

(%o45) $y\,(x)=\frac{5\,e^{x}}{2}-\frac{3\,e^{-x}}{2}-2\,x$

Si **desolve** no encuentra una solución, entonces devuelve “false".
Veamos un ejemplo de un sistema de ecuaciones diferenciales

(%i46) ecd1: 'diff(f(x),x)='diff(g(x),x)+sin(x);

(%o46) $\frac{d}{d\,x}\,f\left(x\right)=\frac{d}{d\,x}\,g\left(x\right)+\sin x$

(%i47) ecd2: 'diff(g(x),x,2)='diff(f(x),x)-cos(x);

(%o47) $\frac{d^2}{d\,x^2}\,g\left(x\right)=\frac{d}{d\,x}\,f\left(x\right)-\cos x$

(%i48) atvalue('diff(g(x),x),x=0,a)$ atvalue(f(x),x=0,b)$ atvalue(g(x),x=0,c)$

(%i51) desolve([ecd1, ecd2], [f(x),g(x)]);

(%o51) $\left[f\left(x\right)=a\,e^{x}+b-a,g\left(x\right)=\cos x+a\,e^{x}+c-a-1\right]$

(%i52) kill(all)$

Operaciones básicas con matrices:

(%i1) A:matrix([1,2,3],[4,5,6],[7,8,9]); B:matrix([9,8,7],[6,5,4],[3,2,1]);

(%o1) $\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$

(%o2) $\begin{pmatrix}9&8&7\\6&5&4\\3&2&1\end{pmatrix}$

(%i3) A+B;

(%o3) $\begin{pmatrix}10&10&10\\10&10&10\\10&10&10\end{pmatrix}$

El producto ordinario de matrices:

(%i4) A.B;

(%o4) $\begin{pmatrix}30&24&18\\84&69&54\\138&114&90\end{pmatrix}$

El producto elemento a elemento

(%i5) A*B;

(%o5) $\begin{pmatrix}9&16&21\\24&25&24\\21&16&9\end{pmatrix}$

El cociente elemento a elemento

(%i6) A/B;

(%o6) $\begin{pmatrix}\frac{1}{9}&\frac{1}{4}&\frac{3}{7}\\\frac{2}{3}&1&\frac{3}{2}\\\frac{7}{3}&4&9\end{pmatrix}$

El producto por un escalar:

(%i7) n*A;

(%o7) $\begin{pmatrix} n & 2\,n & 3\,n \\ 4\,n & 5\,n & 6\,n \\ 7\,n & 8\,n & 9\,n \end{pmatrix}$

Podemos generar matrices de muchas maneras

(%i8) a[i,j]:=iˆ 2 + jˆ 2 $

(%i9) A:genmatrix(a,4,4);

(%o9) $\begin{pmatrix} 2 & 5 & 10 & 17 \\ 5 & 8 & 13 & 20 \\ 10 & 13 & 18 & 25 \\ 17 & 20 & 25 & 32 \end{pmatrix}$

También de manera interactiva:

(%i10) n:3$

(%i11) M:entermatrix(n,n)$

```
Is the matrix  1. Diagonal  2. Symmetric  3. Antisymmetric  4. General
Answer 1, 2, 3 or 4 : 1;
Row 1 Column 1: (x+y)^n$
Row 2 Column 2: (x-y)^(n+1)$
Row 3 Column 3: (x.y)^(n-1)$
Matrix entered.
```

(%i12) M;

(%o12) $\begin{pmatrix} (y+x)^3 & 0 & 0 \\ 0 & (x-y)^4 & 0 \\ 0 & 0 & (x \cdot y)^2 \end{pmatrix}$

La matriz identidad de tamaño n x n usamos el comando **ident(n)**, como mostramos a continuación

(%i13) ident(4);

(%o13) $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

A.4 Bibliotecas

No todos los comandos están disponibles en la memoria cuando el programa es iniciado. Sólo los comandos estándar son cargados automáticamente. Pero podemos contar con funciones adicionales cargando al programa los diferentes paquetes, módulos o librerías que dispone **Maxima**. Por ejemplo, **vect** nos permite introducir vectores y operar con ellos.

El paquete **vect** debe entonces ser previamente cargado y se hace de la manera siguiente:

(%i1) load(vect)$

Las operaciones básicas de suma, resta, multiplicación y multiplicación por escalares de vectores las podemos ver a continuación, pero primero debemos saber como introducir los vectores al programa.

Por ejemplo, para los vectores $\mathbf{a} = 2\mathbf{i} + 4\mathbf{j} + 6\mathbf{k}$, $\mathbf{b} = 5\mathbf{i} + 7\mathbf{j} + 9\mathbf{k}$ y $\mathbf{c} = \mathbf{i} + 3\mathbf{j}$, tenemos:

(%i2) a:[2,4,6];
(%o2) $[2, 4, 6]$

(%i3) b:[5,7,9];
(%o3) $[5, 7, 9]$

(%i4) c:[1,3,0];
(%o4) $[1, 3, 0]$

(%i5) a+b+c;
(%o5) $[8, 14, 15]$

(%i6) 3*a+5*b-c;
(%o6) $[30, 44, 63]$

Si queremos calcular el módulo de los vectores podemos hacerlo definiendo una función: la función módulo, como mostramos a continuación:

(%i7) modulo(a):=sqrt(a.a);
(%o7) $\text{modulo}(a) := \sqrt{a.a}$

(%i8) modulo(a); modulo(b); modulo(c);
(%o8) $2\sqrt{14}$
(%o9) $\sqrt{155}$
(%10) $\sqrt{10}$

Para el producto escalar usamos un punto

(%i11) a.b;
(%11) 92

Mientras que para el producto vectorial debemos usar la tilde $\sim$ y escribir lo siguiente

(%i12) express(a˜ b);
(%o12) $[-6, 12, -6]$

(%i13) express(b˜ a);
(%o13) $[6, -12, 6]$

De manera que el producto triple lo podemos calcular asi:

(%i14) c.(express(a˜ b));
(%o14) 30

Otra librería que podemos explorar es **plotdf** que nos permite realizar gráficos del tipo $y' = f(x, y)$ y hacernos una idea de cómo es la solución de ésta ecuación diferencial.

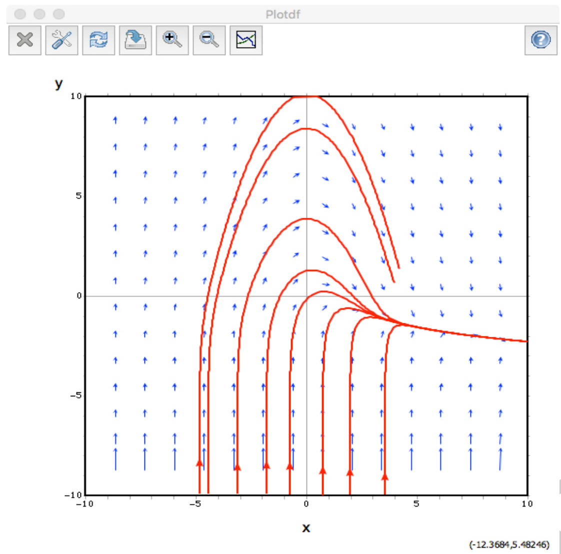


Figura A.2: Curvas integrales para $y' = -x + e^{-y}$

(%i1) load(plotdf)$

La librería **plotdf** nos permite estudiar los campos de direcciones y las curvas integrales a través del estudio de las pendientes. Veamos la siguiente ecuación diferencial

$$y' = -x + e^{-y}$$

su solución, de manera gráfica, es decir los campos de direcciones lo podemos obtener escribiendo el siguiente comando.

(%i2) plotdf(-x+exp(-y));

La gráfica que resulta puede verse en la Figura A.2 y cada curva integral (curvas en rojo) se obtiene haciendo un “click"sobre un punto de la gráfica, esto generará la curva integral que pasa por ese punto.

Por otra parte, existen varias opciones para el comando **plotdf**. Supongamos que queremos la trayectoria que pase por el punto específico $(2, 3)$. Para tal fin escribimos

(%i3) plotdf(-x+exp(-y),[trajectory_ at,2,3]);

Y la gráfica obtenida puede verse en la Figura A.3.

También nos podemos encontrar con que la ecuación diferencial depende de algún parámetro, digamos κ. Por ejemplo

$$y' = -x + \kappa e^{-y}$$

y nos gustaría obtener una gráfica para algún valor del parámetro en partícular, digamos $\kappa = -0{,}5$. Entonces podemos escribir

(%i4) plotdf(-x+exp(-kappa*y),[parameters,"kappa=-0.5"]);

O, si queremos recorrer, en una misma figura, los diferentes valores del parámetro usamos la opción **sliders**, como se muestra a continuación

(%i1) plotdf(-x+exp(-kappa*y),[parameters,"kappa=-0.5"],[sliders,"kappa=-3:3"]);

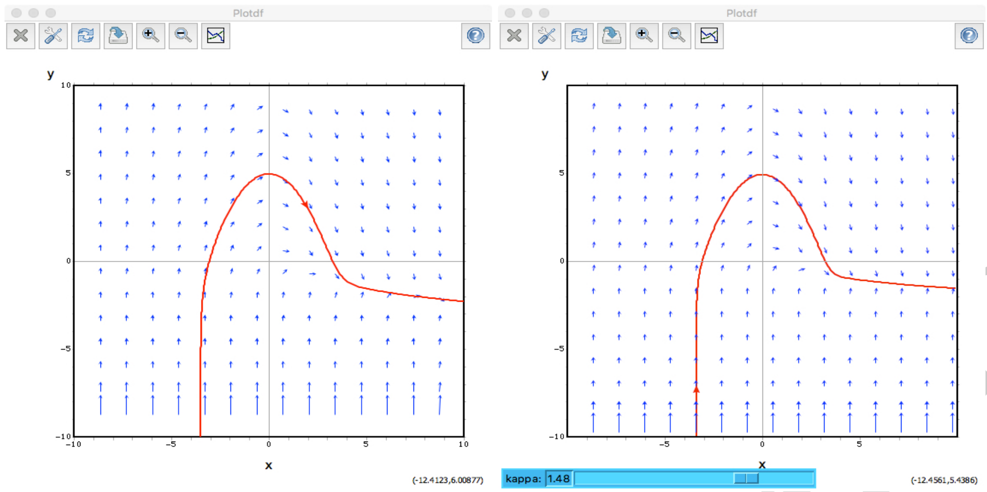


Figura A.3: A la izquierda la curva integral para $y' = -x + e^{-y}$ y que pasa por el punto $(2, 3)$ y la derecha para diferentes valores de κ.

La gráfica que se obtiene se muestra en la Figura A.3 (derecha), y como se puede ver, en la parte inferior aparece un botón deslizante y el valor del parámetro κ. Al deslizar el botón, estaremos cambiando el valor del parámetro que en este caso variará entre -3 y 3, podremos apreciar entonces como los campos de direcciones y las curvas integrales seleccionadas cambian.

A.5 Maxima en modo texto

Es posible utilizar **Maxima** en un computador que funcione bajo alguno de los diferentes versiones de sistemas operativos tipo UNIX, como por ejemplo Linux, esto lo podemos hacer cuando no queremos utilizar el ambiente gráfico. Podemos recurrir al ambiente de texto escribiendo el comando **maxima** en un terminal de nuestro computador, esto hará que entremos en un ambiente de cálculo que funcionará exclusivamente en modo texto y aparecerá, luego de una bienvenida, el aviso de espera o prompt. Para finalizar una sesión en **Maxima** se utiliza el comando **quit()**. Esta posibilidad que ofrece el programa es muy conveniente a la hora de realizar grandes cálculos ya que podemos dejar el proceso en modo “background” y utilizar el computador en otra actividad.

Al entrar en este modo al programa tendremos un mensaje como el que se muestra a continuación y donde aprovecharemos de hacer un par de cálculos a modo de ejemplo.

```
Obatala\% maxima
Maxima 5.36.1 http://maxima.sourceforge.net
using Lisp SBCL 1.2.10
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(\%i1) integrate( tan(x), x );
(\%o1)                              log(sec(x))
```

```
(\%i2) float(sqrt(\%pi));
(\%o2)                          1.772453850905516
(\%i3) quit();
```

Sobre UNIX podemos utilizar los archivos de entradas y salidas estándar para leer e imprimir información en el terminal: $<$, $>$, $|$.

```
Obatala\% maxima < archimax.txt
```

Con esta instrucción **Maxima** ejecutará todos los comandos que se encuentran en el archivo de texto archimax.txt e irá mostrando los resultados en pantalla

En la siguiente instrucción **Maxima** ejecutará todos los comandos que se encuentran en el archivo de texto archimax.txt pero escribirá los resultados en el archivo de salida llamado archimax.out

```
Obatala\% maxima < archimax.txt > archimax.out
```

También se puede hacer que todos los comandos del archivo sean ejecutados para luego ser enviados al terminal pero paginados.

```
Obatala\% maxima < archimax.txt | more
```

Maxima puede ser detenido temporalmente con el comando "Control Z" de manera que para poner procesos en "background" se procede de la manera usual:

```
Obatala\% maxima  < archimax.txt > archimax.out
^Z
Suspended
Obatala\%> bg
[2] maxima  < archimax.txt > archimax.out &

Obatala\%
```

O si lo preferimos, y de manera equivalente, podemos escribir la instrucción pero ponemos al final &

```
Obatala\% maxima  < archimax.txt > archimax.out &
[1] 5114
```

A.6 Invocando la ayuda

El ambiente **wxMaxima** permite acceder al manual de ayuda fácilmente visible en la barra de herramientas, parte superior de la ventana. Pero también si conocemos el comando podemos escribir, por ejemplo:

(%i1) describe(diff);

```
 0: diff  (Functions and Variables for Differentiation)
 1: diff <1>  (Functions and Variables for Differentiation)
 2: diff <2>  (Functions and Variables for itensor)
Enter space-separated numbers, 'all' or 'none':
```

Al seleccionar una de las opciones, por ejemplo si escribimos 1 después de los dos puntos, aparecerá la descripción completa del comando:

```
 -- Function: diff
          diff (<expr>, <x_1>, <n_1>, ..., <x_m>, <n_m>)
          diff (<expr>, <x>, <n>)
          diff (<expr>, <x>)
          diff (<expr>)
     Returns the derivative or differential of <expr> with respect to
     some or all variables in <expr>.
     'diff (<expr>, <x>, <n>)' returns the <n>'th derivative of <expr>
     with respect to <x>.
     'diff (<expr>, <x_1>, <n_1>, ..., <x_m>, <n_m>)' returns the mixed
     partial derivative of <expr> with respect to <x_1>, ..., <x_m>.  It
     is equivalent to 'diff (... (diff (<expr>, <x_m>, <n_m>) ...),
     <x_1>, <n_1>)'.
     'diff (<expr>, <x>)' returns the first derivative of <expr> with
     respect to the variable <x>.
     'diff (<expr>)' returns the total differential of <expr>, that is,
     the sum of the derivatives of <expr> with respect to each its
     variables times the differential 'del' of each variable.  No
     further simplification of 'del' is offered.
     The noun form of 'diff' is required in some contexts, such as
     stating a differential equation.  In these cases, 'diff' may be
     quoted (as ''diff') to yield the noun form instead of carrying out
     the differentiation.....

     There are also some inexact matches for `diff'.
     Try `?? diff' to see them.
```

También podemos utilizar:

(%i2) apropos("diff");

```
[covdiff,diff,maxtaydiff,pdiff_diff_var_names,pdiff_prime_limit...]
```

O pedirle al programa algunos ejemplos

(%i3) example(diff);

```
(%i4) kill(f,g,h,x,y)
(%o4) done
(%i5) diff(sin(x)+x^3+2*x^2,x)
(%o5) cos(x)+3*x^2+4*x
(%i6) diff(sin(x)*cos(x),x)
(%o6) cos(x)^2-sin(x)^2
```

```
(%i7) diff(sin(x)*cos(x),x,2)
(%o7) -4*cos(x)*sin(x)
    .
    .
    .
```

Las ayudas completas de **Maxima** se pueden consultar en: http://maxima.sourceforge.net/es/

A.7 Ejercicios

1. Calcule:
   (a). los 70 primeros decimales del número e
   (b). el arco coseno hiperbólico de 1
   (c). la expansión de $\sin(2\arctan(x))$
2. Para la siguiente función
$$f(x) = \frac{(x+1)^3}{\sqrt{x^2-1}}$$
   Calcule
   (a). $\frac{\partial f(x)}{\partial x}$, $\frac{\partial f(x)}{\partial x^2}$
   (b).
$$\int f(x)dx\,, \qquad \int_2^4 f(x)dx$$
   (c).
$$\lim_{x\to\infty} f(x)\,, \qquad \lim_{x\to-\infty} f(x)\,, \qquad \lim_{x\to 0} f(x)$$
   (d). Haga un gráfico de $f(x)$ para valores de $x \in [-5,5]$.
3. Encuentre las raíces de
$$p = x^7 + x^5 + 2x + x$$
4. Resuelva la siguiente ecuación diferencial
$$x\frac{dy(x)}{dx} = y(x)\ln(xy(x)) - y(x);$$
   y haga una gráfica del campo de direcciones que muestre algunas curvas integrales.
5. Resuelva la siguiente ecuación diferencial
$$\frac{d^2y(t)}{d^2t} - \left(1-y(t)^2\right)\frac{dy(t)}{dt} + y(t) = 0\,, \quad \text{con:} \;\; y(0) = 0\,, \;\; \frac{dy(t)}{dt} = -0{,}1$$
6. Realice los ejercicios de la sección 1.2.4 con la ayuda de la librería **vect** de **Maxima**.
7. En un archivo de texto escriba las siguientes instrucciones que tienen que ver con operaciones de números complejos:

```
z_1=1+2%i;
z_2=3+4%i;
z_1+z_2;
z_1*z_2;
expand(%);
z_1/z_2;
```

```
rectform(%);
polarform(%);
```

guarde el archivo con el nombre pruebamax.txt y en un terminal de su computador, y en el mismo directorio donde está el archivo escriba:

```
localhost%  maxima < pruebamax.txt > pruebamax.out &
```

y verifique que en el archivo pruebamax.out se haya realizado los cálculos.

Índice alfabético