"El saber de mis hijos
hará mi grandeza"

Universidad De Sonora

División de Ciencias Exactas y Naturales

Licenciatura En Física

Física Computacional I

Reporte de Actividad 9

"Sistema de Álgebra Computacional Maxima"

César Omar Ramírez Álvarez

Profr. Carlos Lizárraga Celaya

Hermosillo, Sonora Abril 28 de 2018

Introducción

El presente informe es producto de la novena práctica de la materia de Física Computacional I, en esta ocasión se utiliza un Sistema de Álgebra Computacional denominado "Maxima". Un sistema algebraico computacional o sistema de álgebra computacional (CAS, del inglés computer algebra system) es un programa de ordenador o calculadora avanzada que facilita el cálculo simbólico. La principal diferencia entre un CAS y una calculadora tradicional es la habilidad del primero para trabajar con ecuaciones y fórmulas simbólicamente, en lugar de numéricamente. Es decir, una expresión como $a + b$ es interpretada siempre como "la suma de dos variables", y no como "la suma de dos números" (con valores asignados).

A lo largo del reporte trataremos de recrear un breve tutorial, recorriento las funciones más importantes que tiene Maxima para que después de conocerlo, logre ser una herramienta aprovechable como apoyo en nuestros cursos durante la licenciatura.

Contextualizando

El sistema de álgebra computacional Maxima es un motor de cálculo simbólico escrito en lenguaje Lisp publicado bajo licencia GNU GPL.

Cuenta con un amplio conjunto de funciones para hacer manipulación simbólica de polinomios, matrices, funciones racionales, integración, derivación, manejo de gráficos en 2D y 3D, manejo de números de coma flotante muy grandes, expansión en series de potencias y de Fourier, entre otras funcionalidades. Además tiene un depurador a nivel de fuente para el código de Maxima.

A continuación iniciaremos con un breve tutorial de algunas funciones básicas e importantes que tiene este sistema.

Operaciones Básicas

Operaciones Aritméticas en Modo Exacto

El sistema wxMaxima utiliza la notación estándar para escribir las operaciones matemáticas básicas: suma [+], resta [-], producto [*], cociente [/] y potencia [^]. Tras introducir una expresión, basta pulsar *INTRO* para que Maxima lo simplifique.

```
(%i2)   (2+3*(a²)³)/12;
(%o1)   (3a⁶+2)/12

(%i2)   sqrt(12)+25^8;
(%o2)   2√3+152587890625
```

Maxima trabaja en aritmética exacta. Si el resultado de un cálculo es un número con muchos dígitos, en principio puede no mostrarlos todos. Pero podemos conseguir que los muestre con el menú *Maxima-Cambiar pantalla 2D*, sin más que elegir *ascii*.

```
(%i3)   136!;
(%o3)   365904288195254865768972722051[173 dígitos]000000000000000000000000000000

(%i4)   set_display('ascii)$

(%i5)   136!;
(%o5) 365904288195254865768972722051989334542861729518157261561238151014051895\
82089242773975735166401987907830880230417721361838572072126683305250473185240 2\
76110436058808776620558462614334914409164842205184000000000000000000000000000\
00000
```

Obtener el Resultado Aproximado de una Operación

Para obtener el resultado aproximado, a la hora de evaluar la expresión en un valor determinado, añadiremos la directiva *numer*.

```
(%i7)   1/5+sqrt(7);
(%o7)   √7+1/5

(%i8)   1/5+sqrt(7),numer;
(%o8)   2.845751311064591
```

Funciones Básicas

Introducir y Manejar Funciones

Definir una Función

Se puede definir la función con una o varias variables.

```
(%i9)   f(x):=2*x;
(%o9)   f(x):=2 x

(%i10)  g(x,y):=2*x^2+5*y^4;
(%o10)  g(x,y):=2 x²+5 y⁴
```

Es fundamental identificar las variables y usar := para definirla.
Para definir una función a trozos se utiliza: *If condición then sentencia1 else sentencia2*

Evaluar una Función

Una vez definida la función, para evaluarla en $x = a$ bastará con ejecutar $f(a)$. A continuación, vamos a evaluar las funciones definidas anteriormente.

```
(%i12)  f(2);
(%o12)  4

(%i13)  g(2,1);
(%o13)  13

(%i14)  h(5),numer;
(%o14)  -0.9589242746631385
```

Tabla de Valores de una Función

Para obtener con wxMaxima varios valores de una expresión se utiliza la función *makelist*, cuya sintaxis es *makelist (expresión, variable, inicio, fin)*.

Al ejecutar la instrucción anterior, se evalúa la expresión para los distintos valores de la variable, desde inicio hasta fin, con paso de longitud 1.

```
(%i15)  f(x):=5*x+sin(x);
(%o15)  f(x):=5 x+sin(x)

(%i16)  makelist(f(x),x,0,10),numer;
(%o16)  [0.0,5.841470984807897,10.90929742682568,15.1411
40.98935824662338,45.41211848524175,49.45597888911063]
```

Resolver Ecuaciones

El comando básico para resolver ecuaciones de todo tipo es *solve*.
En algunos casos el sistema no sabe muy bien cómo resolver la ecuación y la devuelve sin cambios:

```
(%i17)  solve(4*x^2+x=5,x);
(%o17)  [x=-5/4,x=1]

(%i18)  solve(sqrt(x-4)+x=6,x);
(%o18)  [x=6-sqrt(x-4)]
```

En estos casos, se puede "ayudar" al programa tal y como lo haríamos a mano. En este caso, dejando en un miembro la raíz ($\sqrt{x-4} = 6 - x$) y elevando al cuadrado:

```
(%i18)  solve(sqrt(x-4)+x=6,x);
(%o18)  [x=6-sqrt(x-4)]

(%i19)  solve(x-4=(6-x)^2,x);
(%o19)  [x=5,x=8]
```

Las ecuaciones se pueden etiquetar en Maxima. Esto facilita mucho la labor de saber si el resultado que obtenemos con *solve* es efectivamenteuna solución o no, cosa que podemos hacer con *subst*.

```
(%i23)  eq1:2*x+y=5;
(eq1)   y+2 x=5

(%i24)  eq2:2*x+5*y=10;
(eq2)   5 y+2 x=10

(%i25)  solve([eq1,eq2],[x,y]);
(%o25)  [[x=15/8,y=5/4]]
```

En primer lugar, definimos una ecuación, y posteriormente la resolvemos. Por último, aplicamos el comando mencionado para saber si la solución obtenida es la correcta o no.

```
(%i19)  solve(x-4=(6-x)^2,x);
(%o19)  [x=5,x=8]

(%i20)  eq1:x^3-x^2-x+1=0;
(eq1)   x^3-x^2-x+1=0

(%i21)  solve(eq1,x);
(%o21)  [x=-1,x=1]

(%i22)  subst(X=-1,eq1);
(%o22)  x^3-x^2-x+1=0
```

Resolución Aproximada de Ecuaciones

Cuando Maxima no puede calcular la solución exacta, devuelve la propia ecuación. Podemos buscar una solución aproximada en un determinado intervalo utilizando la opción de menú *Ecuaciones / Calcular raíz*.

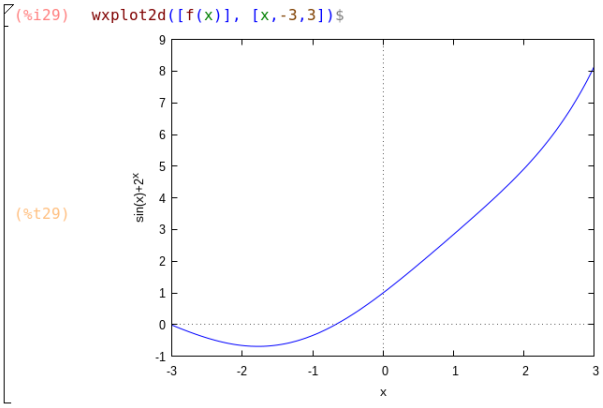
```
(%i26)  solve(x=cos(x),x);
(%o26)  [x=cos(x)]

(%i27)  find_root(x=cos(x), x, -1, 1);
(%o27)  0.7390851332151607

(%i28)  f(x):=2^x+sin(x);
(%o28)  f(x):=2^x+sin(x)
```

Representaciones Gráficas

Funciones de una variable. (Gráficas en 2D)

Se puede representar gráficamente una función, previamente definida, usando el botón correspondiente del menú de comandos, para ello se marca con el ratón la expresión a representar (no la asignación), es decir si queremos representar una función que tenemos definida en la forma $f(x) := 2^x + \sin(x)$, marcaríamos solo $2^x + \sin(x)$, o solo $f(x)$.

Posteriormente, pinchamos en el menú de *Matemáticas generales* el botón *Gráfico 2D.* Aparece una pantalla para introducir el rango y el formato deseado y al pulsar *Aceptar* aparece el gráfico.



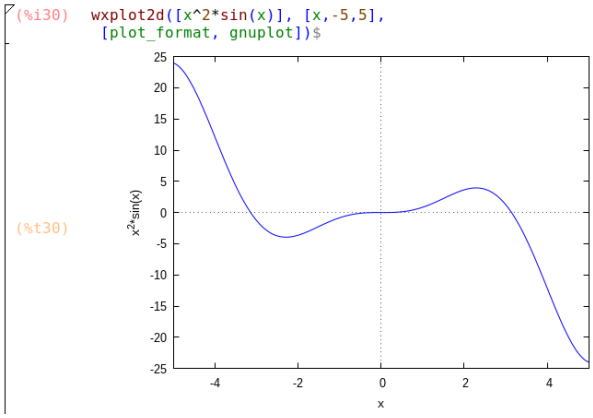
Si queremos que el gráfico se abra en otra pestaña, tiene la ventaja de que te indica las coordenadas del punto marcado por el cursor en cada momento, deberemos elegir el formato: *openmath* o bien *gnuplot*. En este caso, es recomendable cerrar la ventana gráfica al terminar, para seguir trabajando.

Otra forma de crear un gráfico 2D es introduciendo la expresión o el nombre de la función manualmente en la interfaz de los gráficos. Para ello, elegimos la opción de menú *Gráficos / Gráficos 2D*, se abre la ventana de diálogo en la que podemos introducir todos los datos, incluida la expresión a representar.

Trabajar en Modo Gráfico

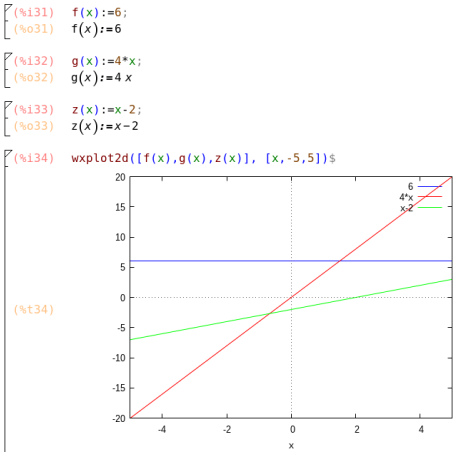
Para interactuar con un gráfico se debe elegir el formato: *gnuplot*.

En la ventana gráfica, además del menú de comandos propios de este tipo de ventanas, tenemos en la parte inferior las coordenadas del punto en que se encuentrael cursor. En la zona lateral izquierda, aparece la función representada.



Representar Varias Funciones Simultaneamente

Una vez definidas las funciones, por ejemplo, $f(x)$, $g(x)$, y $z(x)$, abrimos el menú de *Gráficos 2D* y las introducimos separadas por comas. Al pulsar *Aceptar* aparece la pantalla con todas las funciones pintadas.



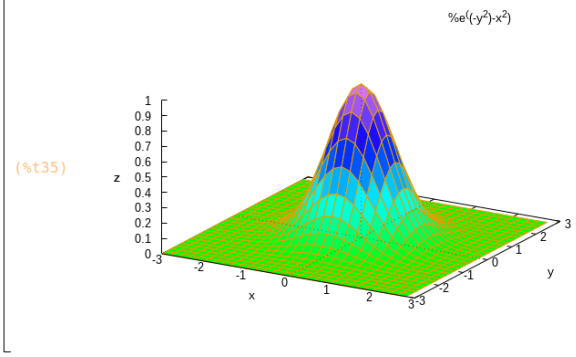
En la parte superior de la primera imagen se muestrala correspondencia entre funciones y colores de la gráfica, y abajo las coordenadas en donde esté situado el cursor.

Funciones de Dos Variables. (Gráficas en 3D)

La gráfica de una función de dos variables es una superficie, que se puede representar con el botón *Gráficos 3D*, introduciendo los datos en la ventana de diálogo, de modo análago a como se hace con unafunción de una variable:

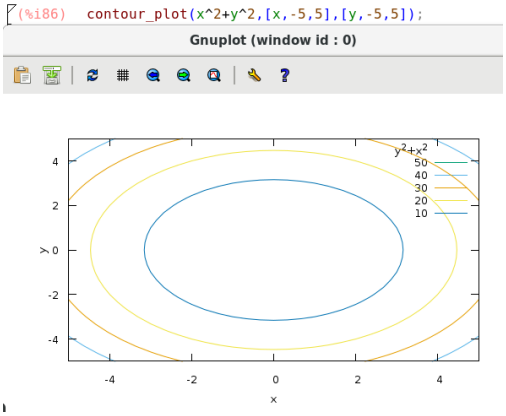
Al pulsar Aceptar, aparecerá en la hoja de trabajo la instrucción y el dibujo de la superficie:

```
(%i35)  wxplot3d(%e^(-(x^2+y^2)), [x,-3,3], [y,-3,3],
         [gnuplot_pm3d,true])$
```



También se pueden representar las curvas de nivel de una superficie utilizando la función *contour_plot*:

```
(%i86)  contour_plot(x^2+y^2,[x,-5,5],[y,-5,5]);
```



Límites, Derivadas e Integrales

Límites

El comando para hallar límites, *limit*, es uno de los más sencillos de usar. Para calcular $\lim_{x\to 0} f(x)$ se usa *limit* $(f(x), x, a)$

```
(%i37)  f(x):=(x+a)*(x^2+b*x+c);
(%o37)  f(x):=(x+a)(x^2+b x+c)

(%i38)  limit(f(x),x,1);
(%o38)  (a+1) c+(a+1) b+a+1

(%i39)  limit(f(x),x,k);
(%o39)  k^3+(b+a) k^2+(c+a b) k+a c
```

Maxima identifica límites finitos e infinitos. También puede calcular límites laterales, con las directivas *minus* (límites por la izquierda) y *plus* (límites por la derecha).

```
(%i40)  limit(1/x,x,0,minus);
(%o40)  -∞

(%i41)  limit(1/x,x,0,plus);
(%o41)  ∞
```

Otra forma, de calcular un límite sería marcar la expresión deseada, y seleccionar el botón *Límite* de la barra de herramientas. Indicamos la variable (x), el punto (x_0) y la dirección (Izquierda, Derecha o ambos), y pulsamos aceptar.

Derivadas

La instrucción para derivar respecto a una variable es *diff (función, variable)*.

```
(%i42)  g(x,y):=sin(x*y);
(%o42)  g(x,y):=sin(x y)

(%i43)  diff(g(x,y),x);
(%o43)  y cos(x y)

(%i44)  diff(g(x,y),y);
(%o44)  x cos(x y)
```

Se pueden calcular derivadas segundas, terceras, etc., sin más que indicar el orden de derivación a continuación de la variable.

```
(%i44)  diff(g(x,y),y);
(%o44)  x cos(x y)

(%i45)  diff(g(x,y),x,2);
(%o45)  -y^2 sin(x y)

(%i46)  diff(g(x,y),x,4);
(%o46)  y^4 sin(x y)
```

Otra manera, sería marcar la expresión que queremos derivar, seleccionar el botón *Derivar* de la barra de herramientas, y en la ventana de diálogo, indicamos la variable (x) y el orden de la derivada que queremos calcular.

Integrales

La principal orden de Maxima para calcular integrales es *integrate*. Nos va a permitir calcular integrales, tanto definidas como indefinidas, con mucha comodidad.

Comencemos por integrales indefinidas.

```
(%i48)  integrate(x*sin(x),x);
(%o48)  sin(x)-x cos(x)

(%i49)  integrate(1/(x^4-1),x);
                  log(x+1)   atan(x)   log(x-1)
(%o49)  - ------- - ------ + -------
             4         2         4
```

Para calcular integrales definidas sólo tenemos que añadir los extremos del intervalo de integración.

```
(%i53)  integrate(x*sin(x),x,%pi/2,%pi);
(%o53)  π-1

(%i54)  integrate(1/(x^2-1),x,2,5);
                    log(6)   log(4)   log(3)
(%o54)  -           ------ + ------ + ------
                      2        2        2
```

Polinomio de Taylor

Para calcular el Polinomio de Taylor de una función $f(x)$, de orden n en torno al punto x_0, se utiliza la instrucción taylor *(funcion, variable, punto,orden-del-polinomio)*.

Al usar el comando *taylor* Maxima devuelve el polinomio de Taylor seguido de puntos suspensivos. Para que éstos no aparezcan bastará con utilizar la sentencia *taytorat*. El inconveniente de usar taytorat es que reduce a común denominador las fracciones. Si quiere recuperarse, por ejemplo para obtener una regla general, se puede usar *expand (expresión)*.

```
(%i55)  f(x):=sin(x);
(%o55)  f(x):=sin(x)

(%i56)  taylor(f(x),x,0,8);
                      3    5    7
                     x    x    x
(%o56)/T/  x - -- + --- - ---- + ...
                     6   120  5040

(%i57)  taytorat(taylor(f(x),x,0,8));
                      7      5      3
                     x  - 42 x  + 840 x  - 5040 x
(%o57)/R/  - ------------------------------------
                            5040

(%i58)  expand(taytorat(taylor(f(x),x,0,8)));
                      7    5    3
                     x    x    x
(%o58)  - ---- + --- - -- + x
                    5040  120   6
```

También se puede obtener el polinomio de Taylor, y la serie de Taylor, mediante la opción de menú *Análisis / Calcular serie*:

Ecuaciones Diferenciales

Para obtener la solución general de una Ecuación diferencial (de primer o segundo orden), utilizaremos la opción de menú *Ecuaciones / Resolver EDO*.

Un detalle importante, que hay que tener en cuenta al introducir una ecuación diferencial es quepara escribir la derivada hay que poner un acento grave en la izquierda de diff.

Si existen condiciones iniciales, una vez obtenida la solución general se podrá obtener la solución particular utilizando las opciones *Ecuaciones / Problema de valor inicial (1)* o *Ecuaciones / Problema de valor inicial (2)* según sea de primer o segundo orden respectivamente.

```
(%i60)  ode2('diff(y,x)=y*x, y, x);
                          x^2
(%o60)  y=%c %e^(--)
                          2

(%i61)  ic1(y=%c*%e^(x^2/2), x=0, y=1);
                          x^2
(%o61)  y=%e^(--)
                          2

(%i62)  ode2('diff(y,x,2)+2*'diff(y,x)+y=0, y, x);
(%o62)  y=(%k2 x+%k1) %e^-x

(%i63)  ic2(y=(%k2*x+%k1)*%e^(-x), x=0, y=1, 'diff(y,x)=2);
(%o63)  y=(3 x+1) %e^-x
```

Sucesiones y Series de Números Reales

Definir una Sucesión en Modo Explícito

Podemos definir una sucesión, conociendo su término general, de igual forma que lo haríamos con una función. Por ejemplo, si, $a_n = \dfrac{1}{3^n}$ introducimos en la línea de comandos la instrucción: $a(n) := 1/3^n$.

Generar Términos de una Sucesión

Para generar términos de la sucesión a(n),previamente definida,podemos utilizarla instrucción m*akelist(a(n), n,n_inicio,n_fin)*.

Si queremos el resultado en modo aproximado, basta añadir al final la sentencia *numer*.

```
(%i64)  a(n):=1/3^n;
                          1
(%o64)  a(n):= --
                          n
                         3

(%i65)  makelist(a(n),n,1,10);
(%o65)  [1/3, 1/9, 1/27, 1/81, 1/243, 1/729, 1/2187, 1/6561, 1/19683, 1/59049]
```

Series

Con Maxima es posible obtenerla suma de n sumandos o el valor numérico de la suma de algunas series, por ejemplo series geométricas. Para ello, disponemos de los comandos:

sum(expr,n,m,p): suma *expr* usando n como variable, desde el valor m al p (que puede ser infinito). Si no puede sumarla, la expresa como un sumatorio.

nusum(expr,n,m,p): como sum pero emplea otro algoritmo más eficaz en expresiones racionales.

load(simplify_ sum): carga el paquete symply_sum, el más potente de Maxima para sumar series.

simplify_ sum(serie): calcula la suma exacta de la serie o indica si la serie es divergente.

También podemos acceder a los comandos *sum* y *nusum* desde el menú *Análisis / Calcular Suma.*

```
(%i67)  sum((-1)^(n+1)/n,n,1,20);
(%o67)  155685007/232792560

(%i68)  sum((-1)^(n+1)/n,n,1,20),numer;
(%o68)  0.6687714031754279

(%i69)  sum((-1)^(n+1)/n,n,1,inf);
(%o69)  sum((-1)^(n+1)/n,n,1,inf)
```

Matrices

Definir una Matriz

Se pueden definir matrices de diferentes formas:

a) Utilizando las distintas opciones del menú *Álgebra*: introducir matriz, generar matriz a partir de expresión,...
b) Declarando sus elementos mediante listas, una para cada fila, con la instrucción *matrix* ([$a11, a12, a13$,...],[$a21$,$a22$,$a23$,...],...[$an1$,$an2$,...])
c) Introduciendo interactivamente bajo demanda sus elementoscon la instrucción *entermatrix (NúmeroFilas, NúmeroColumnas)*
d) Mediante una fórmula que define el elemento genérico de la matriz: *a[i,j] := Fórmula de i y j$ genmatrix (a,NúmeroFilas,NúmeroColumnas)*

```
(%i72)  a:matrix([3,3,3],[1,3,5],[4,2,7]);
                                [ 3  3  3 ]
(a)                             [         ]
                                [ 1  3  5 ]
                                [         ]
                                [ 4  2  7 ]

(%i73)  genmatrix(b,2,2);
                                [ b1,1  b1,2 ]
(%o73)                          [            ]
                                [ b2,1  b2,2 ]

(%i74)  b[i,j]:=i^2+j^2;
                                    2    2
(%o74)                          bi,j := i  + j
```

Operaciones con Matrices

Pueden realizarse diferentes operaciones con matricesusando los siguientes operadores:

+ suma de dos matrices

- diferencia de dos matrices

. producto ordinario de dos matrices

* multiplicación de dos matrices, elemento a elemento, y también multiplicar por un número fijo todos los elementos

/ división de dos matrices, elemento a elemento

^ ^ elevar una matriz a una potencia

^ elevar cada uno de los elementos de una matriz a una potencia

```
(%i77)  a:matrix([1,2,3],[3,2,1]); b:matrix([2,1,2],[2,3,1]);
                                [ 1  2  3 ]
(a)                             [         ]
                                [ 3  2  1 ]
                                [ 2  1  2 ]
(b)                             [         ]
                                [ 2  3  1 ]

(%i79)  print("la suma es")$ a+b;
la suma es
                                [ 3  3  5 ]
(%o79)                          [         ]
                                [ 5  5  2 ]

(%i81)  print("el producto elemento a elemento")$ a*b;
el producto elemento a elemento
                                [ 2  2  6 ]
(%o81)                          [         ]
                                [ 6  6  1 ]
```

Conclusión

Los Sistemas Algebraicos Computacionales como el caso de Maxima son buenas herramientas que nos facilitan el hacer cáculos matemáticos, que si le entramos "a pie" es muy poca optimización del tiempo. La verdad, me asombra las poderosas funciones que trae consigo este software, que quiza no fueron mencionadas todas pero tarte de ser centrarme en lo más básico para aprender a utilizarlo rápidamente cuando se me ofrezca.

Es de muy buen aprendizaje aprender a usar este tipo de herramientas que nos facilitan cálculos y pues con la realización de este tutorial y la investigación antes de, puedo decir que ya se varias de las funciones con las que cuenta, y recalco que me servirá a lo largo de la licenciatura. Me gustó mucho la parte de graficación en 3D, por lo que en el final del documento de práctica agregue tres gráficas.

Bibliografía

- Maxima. (2018). Recuperado desde: https://es.wikipedia.org/wiki/Maxima
- Primero Pasos en Maxima (2018). Recuperado desde: http://maxima.sourceforge.net/docs/tutorial/es/max.pdf
- Mini-Manual wxMaxima (2018). Recuperado desde: http://giematic.etsisi.upm.es/images/pdf/TUTORIAL%20WXMAXIMA.pdf
- wxMaxima Tutorials. (2018). Recuperado desde: http://www.scotchildress.com/wxmaxima/
- A. Maxima Tutorial. (2018). Recuperado desde: https://def.fe.up.pt/dynamics/maxima_tutorial.html

Apéndice

1. **¿Cuál fue tu primera impresión de wxmaxima?**
   Primeramente no sabia nada acerca de su funcionamiento y como es que era que operaba, llegué a pensar que era como tipo programación, pero con la investigación y las lecturas de los tutoriales llegué a la conclusión que es una herramienta muy útil para resolver cálculos matemáticos.

2. **¿Crees que esta herramienta puede ser útil en otros de tus cursos?**
   Claro que sí, sobre todo en todas del área de físico-matemáticas ya nos puede apoyar para una rápida solución de los problemas

3. **¿Qué se te dificultó mas en esta actividad?**
   En un principio, descargue una versión mal del programa y no me funcinó, pero en general estuve sin dificultades

4. **¿Se te hizo compleja esta actividad? ¿Cómo la mejorarías?**
   No estuvo completa, y me gustó la modalidad del tutorial, ya que cada quién se extiende hasta donde quiera conocer.