Tutoría 2
Física Computacional I
Grado en Física

Javier Carrasco Serrano

Física Computacional I, Las Tablas

Tema 2: Introducción a Maxima

02

2.1. ¿Por qué Maxima?

- http://maxima.sourceforge.net/
- Software online: http://maxima-online.org/
- Editor, compilador y gráficos: wxMaxima.
- Cálculo simbólico y numérico; abierto, gratuito.
- Cálculo simbólico: segunda derivada de $f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$.
- Cálculo numérico: $\int_0^{100} \frac{x^2}{3} dx$.
- EDOs, EDPs → Métodos Matemáticos I (segunda parte de la asignatura), Análisis Matemático II
- Otros gratuitos: wolframalpha; otros de pago: matlab, maple, mathematica.

2.2. Sintaxis de Maxima

- Ayuda → Ayuda de Maxima Ayuda → Ejemplo
- 3 maneras de ejecutar:
  - Terminal de comandos (cmd, prompt).
  - **wxMaxima.**
  - XMAxima.

- Sintaxis elemental:

(%i1) → input % → expresión anterior
(%o1) → output
Shift+Enter o Ctrl+Enter → ejecuta sentencia
Flechas arriba y abajo → uso del historial
; → separador explícito de sentencias
$ → separador implícito de sentencias (no saca output por pantalla)
. → separador decimal (anglosajón)
, → separador elementos de una lista
: → asignación
= → igualdad de ecuaciones
kill(all) → reinicia memoria (elimina todas las asignaciones) o Maxima → Reiniciar Maxima
kill(variable) → elimina una asignación → error habitual no borrar las asignaciones!

2.2. Sintaxis de Maxima

- Operadores básicos:

+ (suma), - (resta), * (producto), / (división) n*y (también una expresión 2*x tiene que llevar *), ^ (potencia, pero teclear ^^ para evitar problemas), sqrt (square root, raíz cuadrada)

/* … */ → comentarios del código.

- Funciones básicas:

solve(ecuación, variable) → resuelve una ecuación: solve (x-2=0, x)

llamar a una entrada o salida → %i1; %o1;

‘’%1 → ejecutar una orden anterior (doble apóstrofe)

describe(función) → describe una función (ayuda)

save(“nombre”,variables, resolucion=instrucción) → guarda una sesión (en el cmd), ejemplo save(“prueba”,x,resolucion=%i1);

Quit() → salir de la sesión

Load(“nombre”) → carga una sesión salvada (en el cmd) → para poder cargar el directorio del promt y el del fichero guardado debe ser el mismo.

Resolucion → llamada a sentencia guardada

''resolucion → ejecución de sentencia guardada

2.3. Trabajando con wxMaxima

- Ventajas: texto + código, creación documentos (guarda y carga sesiones trabajo), ayuda (predicción al escribir), funciones en menús (aunque no las creadas por el usuario), ejemplos…
- Estructura de celdas: texto, código, gráficos. Conjunto de instrucciones. Celda → Insertar celda de entrada / texto / título…
- Guardar: formato .wxm (guarda los inputs de las celdas, pero hay que evaluarlas para calcular los outputs) o .wxmx (guarda inputs y outputs).
- Ecuaciones (Q) → permite utilizar algunas funciones sin necesidad de escribirlas (pide las entradas).
- Igual en los menús Álgebra, Análisis, Simplificar, Gráficos y Numérico.
- Simplificar: comando ratsimp(), sirve para simplificar una expresión matemática.
- Menú ayuda.
- Constantes maxima: %pi, %i, %e, %inf, %minf.
- MAXIMA DIFERENCIA ENTRE MAYÚSCULAS Y MINÚSCULAS
- Maxima no tiene en cuenta los espacios en blanco.

2.3. Trabajando con wxMaxima

- Ejemplo cálculo área lateral y volumen cilindro: (R:2,H:3) → carga varios input

```
(R : 2, H : 3)$ A : 2*%pi*R*(R + H); V :  %pi*R^2*H;
```

- Ejecutar expresiones simbólicas: numer → V, numer
  float(V); → Numérico → Establecer precisión (16 decimales defecto)
- <u>Funciones:</u> resolver para cualquier valor el área lateral y volumen del cilindro.
  subst → sustituye expresiones de una variable → subst (a,b,c)
  subst (3,H,subst(2,R,A));

```
A : 2*%pi*R*(R + H); /*área total*/
V :%pi*R^2*H; /*volumen*/
```

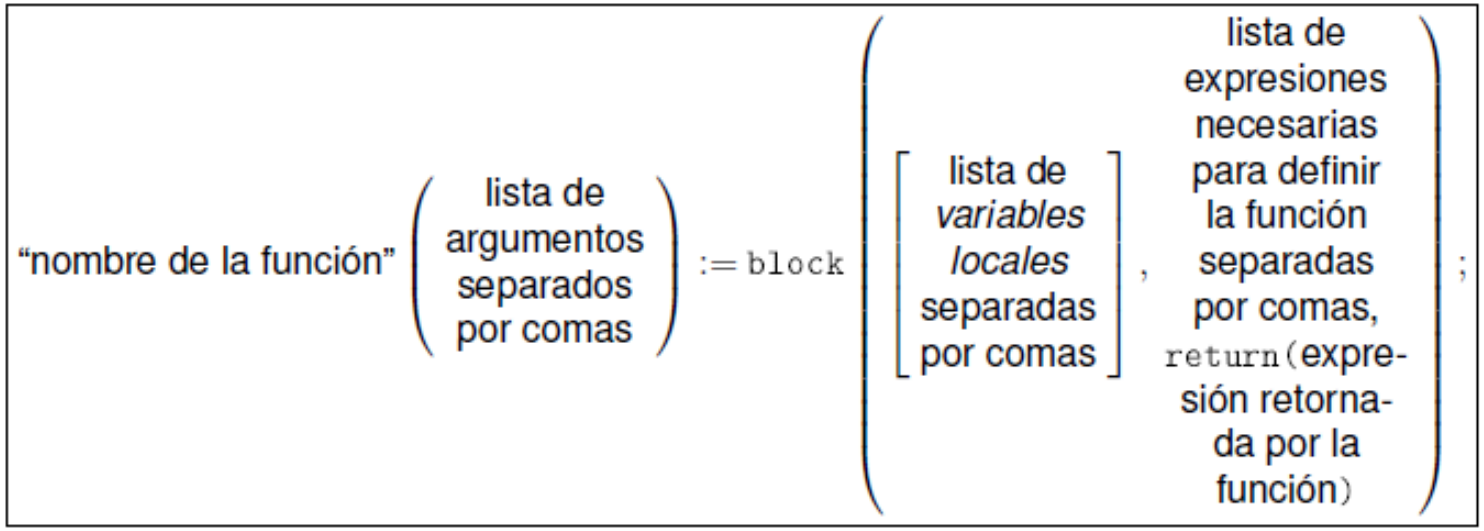
- Resulta poco práctico → mejor definir funciones → símbolo :=

```
A(R, H) :=  2*%pi*R*(R + H); /*área total*/
V(R, H) :=  %pi*R^2*H; /*volumen*/
```

```
A(2, 3);
V(2, 3);
```

2.3. Trabajando con wxMaxima

- Funciones complejas que necesitan evaluar varias expresiones intermedias → :=block




```
(%o1) A(R, H) := block( [Atapa, Alado],
                        Atapa :%pi*R^2,
                        Alado : 2*%pi*R*H,
                        return(2*Atapa + Alado) )$
```

- PEC Maxima!!!

2.4. Expresiones en Maxima

- Teoría de números: primep(n) → verifica si es primo, next_prime(n), gcd(n1,n2) → greater common divisor, ifactors(n) → factoriza n.
- Simplificar, expandir, factorizar: ratsimp() → simplifica expresiones racionales, trigsimp() → trigonométricas, factor() → factoriza, expand()→ desarrolla, rectform() → factorización compleja.
- Funciones trigonométricas: sin(), cos(), tg() → en radianes (0,2pi)

```
sen2x: trigexpand(sin(2*x));
sen3x: trigexpand(sin(3*x)); cos3x: trigexpand(cos(3*x));
trigsimp(trigexpand(%));
```

- Sustituciones: subst() → ejemplo funciones homogéneas

```
(%i1) (x^3 + y^3)/(x^2*y^2) + 2*x^2*y/(2*y^4 - x^3*y - 3*x^2*y^2);
(%i2) subst(lambda * x,x,%o1);
(%i3) subst(lambda*y,y,%o2);
(%i3) factor(%o3 /%o1);
```

2.5. Definición de funciones con Maxima

- Única expresión:

"nombre de la función" (lista de argumentos separados por comas) := expresión que define la función ;

```
(%i1) f(x) := x^2;
```

```
(%i1) f(a);
```

- Varias expresiones:

"nombre de la función" (lista de argumentos separados por comas) := block ([lista de *variables locales* separadas por comas] , lista de expresiones necesarias para definir la función separadas por comas, return(expresión retornada por la función)) ;

2.5. Definición de funciones con Maxima

Por ejemplo, la función `graficaderivadaEPS(f, x, a, b, filename)` calcula la primera (′) y segunda (″) derivada del primer argumento, `f`, respecto del segundo, `x`, a continuación muestra la gráfica de f, f' y f'' para valores de la variable `x` entre `a` y `b`, guarda dicha gráfica en formato `eps` en un archivo con nombre indicado en el último argumento `filename` (de tipo "cadena" o *string*), y finalmente devuelve como output una lista con los resultados obtenidos para f' y f'':

```
(%i1) graficaderivadaEPS(f, x, a, b, filename) := block( [aux],
          aux : [f, diff(f, x, 1), diff(f, x, 2)],
          plot2d( aux, [x, a, b],
                [gnuplot_term, ps], [gnuplot_out_file, filename] ),
          wxplot2d( aux, [x, a, b] ),
          return( [aux[2], aux[3]] )
          )$
```

para ver lo que hace esta función evaluamos, p. ej.

```
(%i2) graficaderivadaEPS(x^3, x, -1, 1, "mi-grafica.eps");
```

necesitamos para generar el nombre de un archivo en el disco. En la siguiente instrucción `wxplot2d( aux, [x, a, b] )` genera la misma gráfica de antes, pero esta vez la muestra por pantalla dentro de la sesión de trabajo de WXMAXIMA. Finalmente, por medio de `return( [aux[2], aux[3]] )` la función "retorna" una lista con las expresiones simbólicas obtenidas para f' y f'', en este caso $[3x^2, 6x]$.

Estudiaremos más adelante guardar ficheros y uso gráficas, no preocuparse por el momento.

2.5. Definición de funciones con Maxima

- Variables locales: pasos intermedios de funciones, sólo útiles una vez. Variables auxiliares que se utilizan sólo dentro de una función.
- Variables globales: cuando se utilizan de manera reiterada → x:1;

```
(%i1) graficaderivadaEPS(f, x, a, b, filename) := block( [aux],
          aux : [f, diff(f, x, 1), diff(f, x, 2)],
          plot2d( aux, [x, a, b],
                [gnuplot_term, ps], [gnuplot_out_file, filename] ),
          wxplot2d( aux, [x, a, b] ),
          return( [aux[2], aux[3]] )
          )$
```

Las funciones no hacen uso de variables globales!!!

2.6. Aprendiendo Maxima

- Guardar archivos desde sesiones de trabajo: archivo → exportar → fichero por lotes maxima (*.mac).
- Es más, para trabajar es mucho mejor guardar en ficheros de texto plano funciones etc, y cargarlas en máxima para su uso.
- Guarrería: copiar y pegar.
- Menos guarrería: archivo → fichero de lotes.
- Recomendable para cargar ficheros → bathcload (filename);

  Filename es el nombre con directorio del fichero:

  C:\Users\Documents\Javier\uned\Tutor\Fisica_Computadores_I\Tutorías\Tutoría 2 - 19 Febrero\ejemplo.mac

  Por defecto busca sólo en los directorios guardados en la variable file_search_máxima.

  Se pueden añadir → append(file_search_máxima, [“mi_directorio”]);

  En la práctica es más cómodo asignar el directorio a una variable global, y llamarla cada vez que hace falta:

```
path_mec : "/home/usuario/bib/Maxima/mecanica/";
```

(obsérvese el uso de comillas dobles, a fin de que la variable `path_mec` sea de tipo *string*) y posteriormente empleamos la función de *concatenación* `concat` para generar el path completo de los archivos a cargar

```
batchload( concat( path_mec, "ec-Newton.mc") );
batchload( concat( path_mec, "ecs-Euler-Lagrange.mc") );
```

Tema 3: Aplicaciones de Maxima en Álgebra

03

3.1. Operaciones elementales con vectores y matrices

- Vectores, matrices, matrices de rotaciones, cálculo de autovalores/autovectores, cambios de base, diagonalización.

https://es.wikipedia.org/wiki/Vector_propio_y_valor_propio

- Vectores:

  listas de elementos → a:[1,2,3,4];

  a[n] → extrae elemento n-ésimo, funciones sobre el vector se aplican a todos sus elementos (porque trabaja con listas, pero no son operaciones vectoriales correctas en muchos casos → a^2 !!!)

  Producto escalar → a·b; (símbolo con tecla mayúscula + 3)

- Matrices: `matrix([elementos fila 1],[elementos fila 2],...)`

  M: matrix([1,2,3],[4,5,6],[7,8,9]);

  M[i,j] → acceso al elemento con la posición (i,j)

  transpose() → matriz traspuesta

  producto de matrices y vectores por matrices igual que en vectores → · (shift + 3)

  entermatrix() → para crear matriz introduciendo elementos

  También se pueden definir como funciones de la posición:

  y genmatrix ();

```
(%i1) componentesH[i,j] := (i + j - 1)^(-1)$
```

3.1. Operaciones elementales con vectores y matrices

- Matrices:

  ident(), zeromatrix(), diagmatrix() → crea matriz identidad, nula o diagonal.

  determinant(M) → calcula determinante

  invert(M) → calcula inversa

  addarrow(), addcol → añade fila o columna

  ```
  M1: addrow(M, [0, 2, 0] );
  ```

  M^2 → matriz con todos los elementos al cuadrado

  M^^2 → matriz al cuadrado

- Rotación: transformación lineal (producto por matriz ortogonal → traspuesta igual a inversa).

$$A = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
(%i1) A(phi) := matrix(
          [cos(phi), sin(phi), 0],
          [-sin(phi), cos(phi), 0],
          [0, 0, 1] );
```

$$(\%o1)\quad A(\phi) := \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
(%i3) invert(A(phi)); transpose(A(phi));
```

3.1. Operaciones elementales con vectores y matrices

- Ángulos de Euler: orientación de cualquier sistema de referencia descompuesta en 3 ángulos.

  https://es.wikipedia.org/wiki/%C3%81ngulos_de_Euler

- Autovalores y autovectores:

$$\det(A - \lambda I) = 0$$

```
(%i1) A : matrix([0, 1, 0], [1, 0, 0], [0, 0, 0])$

(%i2) matrizD : A - lambda * ident(3); D : determinant(matrizD);
```

$$(\%o2)\ \begin{pmatrix} -\lambda & 1 & 0 \\ 1 & -\lambda & 0 \\ 0 & 0 & -\lambda \end{pmatrix}$$

$$(\%o3)\ \lambda - \lambda^3$$

```
(%i4) solve(D = 0, lambda);
(%o4) [lambda = -1, lambda = 1,lambda = 0]
```

3.1. Operaciones elementales con vectores y matrices

- Autovalores y autovectores:

$$(A - \lambda I) \cdot \boldsymbol{v}_\lambda = 0$$

es decir, $A \cdot \boldsymbol{v}_\lambda = \lambda \boldsymbol{v}$

```
(%i5) condicion
```

e imponemos que se
trado:

```
(%i6) condicion
(%i7) condicion
(%i8) condicion
```

Resolviendo cada ur
dientes a cada uno d

```
(%i9) solve([co
= 0], [x, y, z]);
```

Para el siguiente autovalor encontramos

```
(%i10) solve([condicion2[1, 1] = 0, condicion2[2, 1] = 0, condicion2[3,
1] = 0], [x, y, z]);
solve: dependent equations eliminated: (1)
(%o10) [[x=%r2,y=%r2,z=0]]
```

de donde deducimos que el autovector normalizado correspondiente al autovalor $+1$ es

$$\boldsymbol{v}_{+1} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$$

por tanto el subespacio para $\lambda = 1$ es la recta con vector director $\boldsymbol{v}_{+1} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$.

Finalmente para el tercer autovalor encontramos

```
(%i11) solve([condicion3[1, 1] = 0, condicion3[2, 1] = 0, condicion3[3,
1] = 0], [x, y, z]);
solve: dependent equations eliminated: (3)
(%o11) [[x=0,y=0,z=%r3]]
```

de modo que

$$\boldsymbol{v}_0 = (0, 0, 1)$$

3.1. Operaciones elementales con vectores y matrices

- Autovalores y autovectores:

Para el siguiente autovalor encontramos

```
(%i10) solve([condicion2[1, 1] = 0, condicion2[2, 1] = 0, condicion2[3,
1] = 0], [x, y, z]);
solve: dependent equations eliminated: (1)
(%o10) [[x=%r2,y=%r2,z=0]]
```

de donde deducimos que el autovector normalizado correspondiente al autovalor $+1$ es

$$\boldsymbol{v}_{+1} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$$

por tanto el subespacio para $\lambda = 1$ es la recta con vector director $\boldsymbol{v}_{+1} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$.

Finalmente para el tercer autovalor encontramos

```
(%i11) solve([condicion3[1, 1] = 0, condicion3[2, 1] = 0, condicion3[3,
1] = 0], [x, y, z]);
solve: dependent equations eliminated: (3)
(%o11) [[x=0,y=0,z=%r3]]
```

de modo que

$$\boldsymbol{v}_0 = (0, 0, 1)$$

- **COMANDOS MAXIMA: eigenvalues(M); eigenvectores(M);**

3.2. Cambios de Base

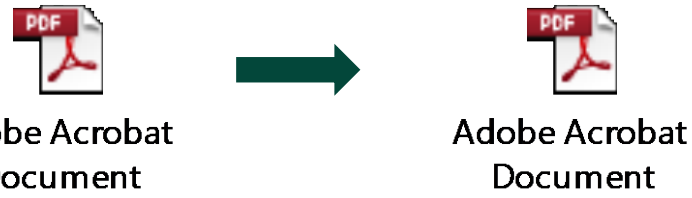
- Concepto vectores unitarios y base.
- Matriz de cambio de base → coordenadas respecto de las bases.

$$v_i = C_{ij}\hat{v}_j \qquad \hat{v}_i = (C^{-1})_{ij}v_j$$

- Elementos de una matriz respecto a otra base:

$$A_{ij} = C_{ik}\hat{A}_{kl}(C^{-1})_{lj} \qquad \hat{A}_{ij} = (C^{-1})_{ik}A_{kl}C_{lj}$$

3.3. Problemas



Cuando los elementos de una lista van a obedecer un cierto criterio de construcción, podemos utilizar la función `makelist` para construirla,

```
(%i23) s:makelist(2+k*2,k,0,10);
```

(%o23) [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]

donde le hemos indicado a Maxima que nos construya una lista con elementos de la forma `2+2*k`, de modo que `k` tome valores enteros de 0 a 10.

- Buscar función makelist() en Manual Maxima:

- Length() → número elementos de la lista.

Gracias!

Javier Carrasco Serrano

Física Computacional I, Las Tablas