# Tutoría 3
# Física Computacional I
# Grado en Física

Javier Carrasco Serrano, javcarrasco@madrid.uned.es

# Repaso funciones

- 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*/     A(2, 3);
V(R, H) := %pi*R^2*H; /*volumen*/               V(2, 3);
```

# Repaso funciones

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

“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`)` ) ;

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

- PEC Maxima!!!

# Repaso funciones

- 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) ) ;

# Tema 4: Cálculo con funciones de una variable

04

# Algunos comandos básicas

Maxima permite realizar operaciones básicas de cálculo. Algunas de ellas son:

- Límites: *limit (función, variable, límite variable);*
- Límites laterales: *limit (función, variable, límite variable, plus/minus);*

Tip: uso del operador ‘ (apóstrofe) al inicio de una expresión hace que ésta no se ejecute. Se puede utilizar para que el resultado quede más legible si se escribe dos veces la expresión, una con apóstrofe y otra sin apóstrofe.

*infinity* → infinito, pero sin determinar el signo.

- Derivada: *diff(función, variable);*
- Derivada de orden mayor a uno: *diff(función, variable, orden);*
- Derivada parcial: *diff(función, variable_1, orden_1, … , variable_n, orden_n);*

# Algunos comandos básicas

Maxima permite realizar operaciones básicas de cálculo. Algunas de ellas son:

- Integral: *integrate(funcion, variable);*
- Integral definida: *integrate(funcion, variable, lím_inferior, lím_superior);*
- Cálculo numérico de una integral: *quad_qag(funcion,variable, lim_inf, lim_sup, algoritmo)* → aproxima una integral que no tiene solución mediante métodos numéricos cuadráticos. Algoritmo elige el método para aproximar la integral (entero del 1 al 6) → salida da valor aproximado, estimación error, veces que se evalúa la función, y un índice de error (0 si no hay errores).

- Desarrollo serie de Taylor: *taylor(funcion, variable, punto_entorno, orden);*

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

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Algunas funciones que es útil desarrollar: exponenciales, logarítmicas, trigonométricas, geométricas, hiperbólicas...

Calcular los desarrollos de Taylor dadas una función, el orden hasta el que se quiere llegar, y el punto en torno al cual se desarrolla la función.

Adobe Acrobat Document

Camino ineficiente →

```
taylorlistineficiente(f, x, x0, orden) := makelist( taylor(f, x, x0, i),
    i, 0, orden, 1);
```

Camino eficiente → utilizar un bucle

```
for i : 1 thru n step 1 do
```

→ añadir registros a una lista → *append*

# 4.2. Problemas resueltos

Algunos ejercicios resueltos de probabilidad:

Adobe Acrobat Document

Adobe Acrobat Document

# Tema 5: Visualización

05

# Algunos commandos básicos

Maxima visualizar funciones en dos y tres dimensiones gracias al paquete *gplot* (se instala al instalar wxMaxima). Algunos de los comandos son:

- Gráficas en 2D: *plot2d (función, [variable, lim_inf, lim_sup], opciones);*

¿qué *opciones* tenemos? → ayuda wxMaxima! → limitar lo que se muestra del eje y, título, modificar escalas ejes, etiquetas variables…

Inconveniente: saca cada gráfica en una ventana diferente.

Solución si se quieren sacar en el notebook → función *wxplot2d(…).*

Solución alternativa para sacar dos funciones en una gráfica: librería *draw* contiene las funciones *draw2d* y *draw3d.* También se puede con la función plot2d si se le pasa una lista de funciones y dominios.

- Gráficas en 3D: *plot2d (función, [variable, lim_inf, lim_sup], opciones);*

→ *wxplot3d(…)*

## 5.1. Problemas resueltos

Con la ayuda de la función describe(plot2d) para conocer comandos de diseño si fuera necesario, utilizar la función *plot2d* para pintar una función y sus desarrollos de Taylor hasta un orden dado (input).

Adobe Acrobat Document

# Tema 3: Aplicaciones de Maxima en Álgebra

05

- 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]
```

- Autovalores y autovectores:

$$(A - \lambda I) \cdot v_\lambda = 0$$

es decir, $A \cdot v_\lambda = \lambda v_\lambda$. Aplicamos la matriz $A$ sobre un vector arbitrario $(x, y, z)$

```
(%i5) condicion: matrizD . [x, y, z]$
```

e imponemos que se cumpla para cada uno de los 3 autovalores que hemos encontrado:

```
(%i6) condicion1 : subst(-1, lambda, condicion)$
(%i7) condicion2 : subst(+1, lambda, condicion)$
(%i8) condicion3 : subst( 0, lambda, condicion)$
```

Resolviendo cada una de estas condiciones obtenemos los autovectores correspondientes a cada uno de los autovalores

```
(%i9) solve([condicion1[1, 1] = 0, condicion1[2, 1] = 0, condicion1[3, 1]
= 0], [x, y, z]);
```

- 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

$$v_{+1} = (1/\sqrt{2}, 1/\sqrt{2}, 0)$$

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

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

$$v_0 = (0, 0, 1)$$

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

# 3.3. Ejercicio resuelto

Adobe Acrobat
Document

# Gracias!