PRÁCTICA 4.ECUACIONES DIFERENCIALES ORDINARIAS. OTROS PROBLEMAS NUMÉRICOS

1 INTRODUCCIÓN.

El alumno debe ir realizando todos los EJEMPLOS que aparecen en esta práctica e ir anotando los resultados en la hoja que se le ha entregado. Además de estos ejemplos, hay planteados unos EJERCICIOS, que, a diferencia de lo que ocurre con los EJEMPLOS, no vienen desarrollados en el lenguaje Maxima, sino que tendrá que ser el alumno el que tenga que introducirlos para obtener el resultado.

2 ECUACIONES DIFERENCIALES.

En esta última parte de la práctica 6, vamos a ver, inicialmente, como podemos resolver con Maxima los ejemplos más típicos de ecuaciones diferenciales ordinarias (edo)m especialmente las de primer orden y algunas de 2do orden.

2.1 Introducción a las EDO

En primer lugar veremos como podemos comprobar que una función determinada es solución explícita de una EDO. Para ello, solo tenemos que sustituir la función en la ec. diferencial, usando el comando ratsim() como vemos a continuación:

```
--> /*EJEMPLO N° 1: Probamos que la función f(x)=2/(2+x)
    es solución de la edo
    xy'+y-y^2=0*/

    y(x):=2/(2+x)$
    ratsimp(x*diff(y(x),x)+y(x)-y(x)^2);
```

```
--> /*EJEMPLO N° 2: Comprobamos que la familia uniparamétrica de
    funciones y=1+c*e^(-x), es solución de la edo de primer orden
    y'+y=1*/

    kill(y)$
    y(x):=1+c*%e^(-x)$
    is(ratsimp(diff(y(x),x)+y(x)=1));
```

Para representar gráficamente algunas soluciones particulares, podemos hacer una lista dando algunos valores a la constante c.Por ejemplo, generamos una lista que llamamos soluciones con valores de c = -3, -2, 0, 1, 2, 3:

```
--> /*EJEMPLO N° 2 (cont.):*/

    soluciones:makelist(1+c*\%e^(-x),c,-3,3);
```

Y podemos representarlas en un sólo gráfico, con la opción plot2d:

```
--> /*EJEMPLO Nº 2 (cont.):*/

    plot2d(soluciones,[x,-3,3]);
```

```
--> /*EJEMPLO Nº 3: Comprobamos que y(t)=c1*cos(4*t)+c2*sin(4*t)
    es solución explícita de la edo de segundo orden
    y''+16y=0*/

    kill(y)$
    y(t):=c1*cos(4*t)+c2*sin(4*t)$
    is(ratsimp(diff(y(t),t,2)+16*y(t))=0);
```

También podemos hacer lo mismo en el caso en que la solución venga dada en forma implícita:

```
--> /*EJEMPLO Nº 4: La ecuación x^2+y^2=25, siendo y función
    implícita de x, es solución implícita de la ecuación
    y*y'+x=0*/

    kill(y,x)$
    depends(y,x)$
    diff(x^2+y^2=25,x)$
    solve(%,diff(y,x));
```

```
--> /*EJEMPLO Nº 4 (cont):*/

    ec:y*'diff(y,x)+x=0$
    solve(ec,diff(y,x));
```

```
--> /*EJEMPLO Nº 5: Comprobamos que e^(x*y)+y=x-1 es solución
    implícita de la ecuación
    y'=(exp(-x*y)-y)/(exp(-x*y)+x)*/

    kill(x,y)$
    depends(y,x)$
    ec:%e^(x*y)+y=x-1$
    diff(ec,x);
    solve(%,diff(y,x));
    ec2:-(y*%e^(x*y)-1)/(x*%e^(x*y)+1)=
    (%e^(-x*y)-y)/(%e^(-x*y)+x);
    ratsimp(%);
```

También podemos comprobar que se cumple un PVI (problema de Cauchy): Por ejemplo, el PVI dado por y'=(3/2)*y^(2/3) y(0)=1 tiene una única solución, puesto que @f/@y no se anula en el punto (x,y)=(0,1). Veamos que la función y=((x+2)^3)/8 es la única solución de este PVI:

```
--> /*EJEMPLO Nº 6:*/

    kill(all)$
    y(x):=(x+2)^3/8$
    is(diff(y(x),x)=3/2*y(x)^(2/3));
```

```
--> /*EJEMPLO Nº 6 (cont): Y además verifica la
    condición inicial*/

    is(y(0)=1);
```

```
--> /*EJEMPLO Nº 7: Comprobamos que la función
    y=2^(1/2)*sin(2*x)-1/2 es solución del PVI
    y''+4y=-2, y(pi/8)=1/2, y'(pi/8)=2*/

    kill(all)$
    y(x):=sqrt(2)*sin(2*x)-1/2$
    is(ratsimp(diff(y(x),x,2)+4*y(x))=-2);
```

```
--> /*EJEMPLO Nº 7 (cont): Y además verifica las
    condiciones iniciales*/

    is(y(%pi/8)=1/2);
    diff(y(x),x)$
    %,x:%pi/8;
```

2.2 EDO en variables separadas (integración directa)

Maxima se puede utilizar no sólo para resolver ciertas EDO's, sino también para aplicar los métodos vistos en clase y así comprobar la resolución paso a paso. Por ejemplo, si queremos resolver la ecuación
y'=log(x)+1-x^2+(x+2)^(1/2)
estamos en el caso más sencillo, pues se puede resolver por integración directa:

```
--> /*EJEMPLO Nº 8:*/

    integrate(log(x)+1-x^2+sqrt(x+2),x);
```

Notemos que la constante de integración no parece (así que cuando sea necesaria habrá que incorporarla a mano).

Maxima tiene varias órdenes para la resolución directa de EDO's, una de ellas es "ode2". Si resuelves la ecuación anterior con ode2 observarás que en este caso no hay, esencialmente, ninguna diferencia:

```
--> /*EJEMPLO Nº 9:*/

    ode2('diff(y,x)=log(x)+1-x^2+sqrt(x+2),y,x);
```

En este caso sí que aparece el parámetro; si te dan alguna condición inicial podemos calcularlo mediante la orden "ic1": Por ejemplo, si en el caso anterior exigimos que y(1)=1:

```
--> ic1(%,x=1,y=1);
```

Supongamos ahora que queremos resolver la EDO x*y'+y=1. Ahora no es posible resolverla integrando directamente (ya que no es posible despejar y' solamente en función de x), pero sí es posible transformar la EDO en una de variables separadas (haciendo las operaciones necesarias), de manera que se obtiene
dy/(1-y)=dx/x,
por lo que podremos integrar ambos miembros e igualarlos para obtener la solución buscada.

```
--> /*EJEMPLO N° 10:*/

    integrate(1/(1-y),y);
    integrate(1/x,x);
```

A continuación podemos escribir la solución igualando ambos resultados y añadiendo la constante de integración en uno de los miembros (la llamaremos "C"). Al usar solve() de la siguiente manera, Maxima despejará la función y si es posible:

```
--> /*EJEMPLO N° 10 (cont.):*/

    solve(integrate(1/(1-y),y)=integrate(1/x,x)+C,y);
```

Para terminar observemos que escribir -e^(-C) es (casi) lo mismo que escribir C, puesto que se trata de una constante al fin y al cabo; pero el programa no hace esa simplificación por sí solo. En este caso funciona mejor la resolución directa con ode2:

```
--> /*EJEMPLO N° 10 (cont.):*/

    ode2(x*'diff(y,x)+y=1,y,x);
```

```
--> /*EJEMPLO N° 11: Para resolver y*y'+x=0 lo haremos directamente
    con ode2:*/

    ode2(y*'diff(y,x)+x=0,y,x);
    solve(%,y);
```

EJERCICIO N° 1:
a) Comprueba que las soluciones que hemos obtenido en los anteriores ejemplos son correctas sustituyendo en la ecuación diferencial.

b) Resuelve mediante separación de variables la ecuación
x^2*y'-y=1
Resuelvela después usando ode2.

c) Resuelve la siguiente edo usando ode2:
y'=(3x^2+4x+2)/(2y-2)
Tomando entonces como constante el valor 3, representa las soluciones en [-2,2] usando plot2d.

2.3 EDO's homogéneas y exactas

También podemos usar el programa para comprobar si una ecuación es homógenea o exacta, aunque en este caso tendremos que darle a Maxima las operaciones que queremos que nos calcule. Por ejemplo, para ver si una EDO es homogénea, tendremos que escribirla en la forma M(x,y)dx+N(x,y)dy=0, y será HOMOGÉNEA si se verifica que M(tx,ty)=t^n*M(x,y) y N(tx,ty)=t^n*N(x,y). Por tanto, los dos miembros de estas igualdades son las que tendríamos que calcular usando Maxima (notemos que posiblemente sea más sencillo hacerlo a mano).
De igual forma, sabemos que la EDO será EXACTA si se verifica que @M/@y=@N/@x, por lo que tendremos que pedirle a Maxima que nos calcule ambos miembros para ver si son iguales.

EJERCICIO N° 2:
En las siguientes EDO'S decir si son homogéneas, exactas, ambas cosas o ninguna:
a) (y^2+y*x)dx+(2*x*y+(x^2)/2)dy=0. (Sol: exacta y homogénea)

b) y/x+y'log(x)=0. (Sol: exacta y no homogénea)

c) x*exp(y/x)dx+ydy=0. (Sol: no exacta y sí homogénea)

Para resolver estas ecuaciones podemos seguir (con ayuda de Maxima) las indicaciones de la teoría (por ejemplo, para el caso de las EDO homogéna realizar el cambio y=x*u(x), que nos da una edo en variables separadas x y u), aunque es más sencillo usar la indicación ode2 (aunque normalmente nos dará siempre una solución en forma implícita)

EJERCICIO N° 3:
Resolver, con ayuda de ode2, las tres EDO's del ejercicio anterior.

2.4 Ecuaciones lineales de 1er orden

En lugar de buscarle un factor integrante que dependa de x (que sabemos que siempre lo admiten) es más sencillo usar la fórmula vista en la teoría: Para una EDO lineal de 1er orden y'+f(x)y=g(x) su solución viene dada por
y=exp(-integral(f(x)))*(Integral(g(x)*exp(integral(f(x)))+cte)
aunque estas ecuaciones también podemos resolverlas usando ode2:

```
--> /*EJEMPLO N° 12: Podemos resolver y'-(sin(x))*y=sin(x)
    usando ode2:*/

    ode2('diff(y,x)-(sin(x))*y-sin(x)=0,y,x);
```

EJERCICIO N° 4:
Resolver la EDO anterior usando la fórmula general del comentario anterior.

2.5 Ecuaciones lineales de orden superior con coeficientes constantes

Vamos a resolver EDO's lineales de orden superior y con coeficientes constantes. Veremos solamente las ecuaciones homogéneas.

Ejemplo 13. Resolver la ecuación homogénea
y^(5)-7*y^(4)+14*y'''+14*y''-47*y-39 = 0

Sabemos que se trata tan sólo de escribir el polinomio característico correspondiente a sustituir las derivadas de y por potencias de cierta variable 'm' y hallar las raíces correspondientes:
En este caso tenemos el par de conjugadas m=3+-2i y las reales m=3 y m = -1, pero no está claro si estas raíces reales son simples o dobles. Para ello habrá que ir un poco más allá y pedir a Maxima que nos factorice el polinomio. Si lo hacemos, veremos que m=-1 es doble y m=3 es simple. Por lo tanto, la solución de la ecuación es
y = C1*%e^(3*x)+C2*%e^(-x)+C3*x*%e^(-x)+C4*%e^(3*x)*cos(2*x)+
+C5*%e^(3*x)*sin(2*x)

Actualmente Maxima está muy limitado para resolver por sí sólo ecuaciones diferenciales de orden superior. Así que nos contentaremos con comprobar la solución obtenida.

EJERCICIO N° 5:
a) Resuelve la ecuación homogénea y'''-3*y'+2*y=0 y comprueba la solución.

b) Idem para y'''-y''-4*y = 0 y comprueba la solución.

2.6 Sistemas de EDO's lineales

La función 'desolve' resuelve sistemas de EDO's lineales utilizando la transformada de Laplace. La dependencia funcional respecto de una variable independiente debe indicarse explícitamente, tanto en las variables como en las derivadas. Por ejemplo, para la primera derivada, a diferencia de la forma 'diff(y,x) (usada en los apartados anteriores) debe usarse la forma 'diff(y(x),x).

```
--> /*EJEMPLO N° 14: He aquí la solución de la ecuación diferencial
    y''(x)+y(x)=2*x*/

    desolve('diff(y(x),x,2)+y(x)=2*x,y(x));
```

```
--> /*EJEMPLO N° 15: Para resolver el sistema x'(t)=y(t); y'(t)=x(t)*/

    desolve(['diff(x(t),t)=y(t),'diff(y(t),t)=x(t)],[x(t),y(t)]);
```

```
--> /*EJEMPLO N° 15 (cont.): Podemos añadir condiciones iniciales al
    sistema anterior x(0)=1; y(0)=0*/

    atvalue(x(t),t=0,1);
    atvalue(y(t),t=0,0);
```

```
--> /*EJEMPLO N° 15 (cont.): Podemos volver a resolver el sistema con
    las condiciones iniciales dadas*/

    desolve(['diff(x(t),t)=y(t),'diff(y(t),t)=x(t)],[x(t),y(t)]);
```

```
--> /*EJEMPLO N° 16: He aquí la solución de la ecuación diferencial
    y'''(x)-3*y''+3*y'- y(x)=e^x*x^2 con las condiciones iniciales
    y(0)=1; y'(0)=0; y''(0)=-2*/

    eq:'diff(y(x),x,3)-3*'diff(y(x),x,2)+3*'diff(y(x),x)-y(x)=%e^x*x^2;
    atvalue('diff(y(x),x,2),x=0,-2)$
    atvalue('diff(y(x),x),x=0,0)$
    atvalue(y(x),x=0,1)$
    desolve([eq],[y(x)]);
```

3 INTEGRACIÓN NUMÉRICA.

Aunque ya hemos visto que Maxima nos permite calcular de manera numérica (aproximada) el valor de casi todas las integrales definidas sin más que utilizar el comando Análisis->Integrar (y marcar "integración definida" e "integración numérica"), es conveniente que el alumno conozca algunos métodos de cálculo numérico de integrales definidas (como son la regla de los trapecios o la regla de Simpson), al igual que era aconsejable conocer métodos de resolución numérica de ecuaciones (como vimos en la Práctica 3). Además estos procedimientos no da tiempo de verlos en las clases teóricas, por lo que no está de mas que les dediquemos un poco de tiempo en las clases prácticas. Lo mismo ocurrirá con otro típico problema de Cálculo Numérico como es el de Interpolación (y que veremos brevemente en la última sección de esta práctica).

3.1 Regla de los trapecios (compuesta)

Cuando se quiere calcular el valor de la integral definida de una función f(x) que no tiene primitiva, es imposible utilizar un método exacto para calcular la misma (como, por ejemplo, la regla de Barrow), sino que tendremos que acudir a un método aproximado (numérico). Lo mismo ocurre cuando en lugar de conocer la expresión de f(x), solo disponemos de sus valores en determinados puntos, y pretendemos calcular, por ejemplo, el área que dicha función encierra en un determinado intervalo.

La regla de los trapecios consiste, básicamente, en lo siguiente:

Se divide el intervalo [a,b] en n subintervalos de igual longitud. Supongamos que (x0,y0) y (x1,y1) son los dos primeros puntos de esta partición. El proceso consiste en aproximar el área de f(x) entre estos dos puntos, por el área del trapecio formado por los lados paralelos de longitudes y0 e y1, y los lados no paralelos de longitudes x1-x0 y la recta que une los puntos (x0,y0) y (x1,y1). En definitiva, lo que hacemos es sustituir el arco de curva que une los puntos (x0,y0) y (x1,y1) por la recta que une ambos puntos. Esto lo haremos para todos los intervalos de la partición. Evidentemente, el valor que obtendremos al sustituir el área que pretendemos calcular por la suma de las áreas de diferentes trapecios, será un valor aproximado, por lo que también tendremos que dar una cota del error que se comete cuando realizamos esta aproximación.

Veamos el proceso con un ejemplo (el desarrollo teórico de este método puede verse en cualquier libro de Cálculo Numérico o buscando en Google):

```
--> /*EJEMPLO 17: Vamos a calcular mediante este proceso la integral
    de f(x)=e^x/x en el intervalo [1,3], dividiendo el intervalo en
    10 subintervalos*/

    n:10$
    a:1$
    b:3$
    A:makelist(2,i,1,n+1)$
    f(x):=exp(x)/x$
    p:makelist(a+i*((b-a)/n),i,0,n)$
    q:makelist(f(p[i]),i,1,n+1)$
    h:(b-a)/n$
    T:float((h/2)*((A.q)-q[1]-q[n+1]));
```

Y este el el valor para la integral que nos da el método que hemos programado.

NOTA: Observamos del ejemplo anterior, que para aplicar este método a cualquier otra función en otro intervalo, sólo es necesario cambiar en dicho ejemplo los valores de a, b, n y f(x).

No obstante, Maxima dispone de un comando de integración numérica "quad_qag" que nos da mucha más información que lo que hemos programado en el anterior ejemplo:

```
--> /*EJEMPLO 17 (Cont.): Vamos a calcular mediante este proceso
    la integral de f(x)=e^x/x en el intervalo [1,3], mediante el
    comando quad_qag*/

    quad_qag(exp(x)/x,x,1,3,1);
```

NOTA: El 1 que se coloca detrás de los extremos 1 y 3 del intervalo de la integral es una opción elegida desde el 1 al 6 según el método numérico de cuadratura empleado. De los 4 números que aparecen en el resultado, el primero es el valor aproximado de la integral (comparar con el obtenido para T), el segundo es el error absoluto estimado de la integración, el tercero es el número de evaluaciones del integrando, y el cuarto, un código de error que puede valer 0, 1, 2, 3 ó 6. Cuando el valor es 0, significa que no ha habido ningún problema en el proceso.

EJERCICIO Nº 6: Calcular de manera aproximada el valor de la integral de f(x)=(1+x^3)^(1/2) en [-1,2]. Hacerla por el método de los trapecios, dividiendo el intervalo en 10 y 50 subintervalos.
Comprobar el resultado con el obtenido al aplicar "quad_qag" y también con el que resulta de utilizar en el menú superior "Análisis->Integrar".

3.2 Regla de Simpson (compuesta)

La regla de Simpson consiste, básicamente, en lo siguiente:
Se divide el intervalo [a,b] en n subintervalos de igual longitud. Supongamos que (x0,y0), (x1,y1) y (x2,y2) son los tres primeros puntos de esta partición. El proceso consiste en aproximar el lazo de curva de f(x) entre estos tres puntos, por el polinomio de grado 2 que pasa por ellos. De esta forma, el área de f(x) en [x0,x2] se aproximará por el área de dicho polinomio de segundo grado (que tiene integral inmediata) en dicho intervalo. Para que siempre sea posible agrupar los puntos de la partición del intervalo en grupos de 3 (para en cada grupo calcular el correspondiente polinomio de 2° grado), es necesario que el n sea PAR, lo cual no le quita generalidad al problema.

```
--> /*EJEMPLO 18: Vamos a calcular mediante este proceso la integral
    de f(x)=e^x/x en el intervalo [1,3], dividiendo el intervalo en
    10 subintervalos*/

    n:10$
    a:1$
    b: 3$
    A:makelist((-1)^i+3,i,1,n+1)$
    f(x):= exp(x)/x$
    p:makelist(a+i*((b-a)/n),i,0,n)$
    q:makelist(f(p[i]),i,1,n+1)$
    h:(b-a)/n$
    S:float((h/3)*(A.q-q[1]-q[n+1]));
```

EJERCICIO N° 7: Calcular de manera aproximada el valor de la integral de f(x)=(1+x^3)^(1/2) en [-1,2]. Hacerla por el método de Simpson, dividiendo el intervalo en 10 y 50 subintervalos.
Comprobar el resultado con el obtenido al aplicar "quad_qag" y también con el que resulta de utilizar en el menú superior Análisis->Integrar

4 POLINOMIO DE INTERPOLACIÓN DE LAGRANGE.

Por interpolación se entiende a la obtención de nuevos puntos a partir del conocimiento de un conjunto finito de puntos.
A veces disponemos de un cierto número de puntos obtenidos por muestreo o a partir de un experimento y queremos construir una función que los ajuste. Otro problema estrechamente ligado con el de la interpolación es el de la aproximación de una función complicada por una más simple: Si tenemos una función cuyo cálculo resulta costoso, podemos partir de un cierto número de sus valores e interpolar dichos datos construyendo una función más simple. En general, por supuesto, no obtendremos los mismos valores evaluando la función obtenida que si evaluásemos la función original, si bien dependiendo de las características del problema y del método de interpolación usado, la ganancia en eficiencia puede compensar el error cometido.

En este problema se trata de partir de n+1 puntos (xk,yk), y obtener una función f que verifique f(xk)=yk para k=0,1,...,n.
El caso más simple consiste en tomar f como un polinomio que pase por todos los puntos dados. Y dentro de esta interpolación polinómica, el procedimiento más sencillo es el de la interpolación de Lagrange:

En este caso, el polinomio viene dado por
L(x)=y0*L0(x)+y1*L1(x)+...+yk*Lk(x)+...+yn*Ln(x)
siendo Lk(x) los llamados polinomios de Lagrange, definidos por
Lk(x)=(((x-x0)(x-x1)...(x-x(k-1))(x-x(k+1))...(x-xn))/
((xk-x0)(xk-x1)...(xk-x(k-1))(xk-x(k+1))...(xk-xn)))

Como siempre, lo vemos con unos ejemplos:

```
--> /*EJEMPLO 19: Vamos a calcular mediante este proceso el
    polinomio interpolador que pasa por los puntos
    P0=(0,0), P1=(1,1),P2=(3,2)*/

    /*En primer lugar definimos los polinomios de Lagrange:*/
    La0(x):=(((x-1)*(x-3))/((0-1)*(0-3)))$
    La1(x):=(((x-0)*(x-3))/((1-0)*(1-3)))$
    La2(x):=(((x-0)*(x-1))/((3-0)*(3-1)))$

    /*y finalmente el polinomio interpolador de Lagrange vendrá
    dado por:*/
    La(x):=0*La0(x)+1*La1(x)+2*La2(x);
```

```
--> /*EJEMPLO 19 (Cont.): Podemos simplifcar el anterior polinomio
    usando ratsimp:*/

     ratsimp(La(x));
```

```
--> /*EJEMPLO 19 (Cont.): Y podemos comprobar que este polinomio
    efectivamente pasa por los puntos dados sin más que hacer:*/
    La(0);
    La(1);
    La(3);
```

```
--> /*EJEMPLO 20: Vamos a calcular mediante este proceso el
    polinomio interpolador que pasa por los puntos
    (0,2),(1,1) y (2,0)*/

    Lb0(x):=(((x-1)*(x-2))/((0-1)*(0-2)))$
    Lb1(x):=(((x-0)*(x-2))/((1-0)*(1-2)))$
    Lb2(x):=(((x-0)*(x-1))/((2-0)*(2-1)))$

    /*y el polinomio interpolador de Lagrange es:*/
    Lb(x):=2*Lb0(x)+1*Lb1(x)+0*Lb2(x);
    ratsimp(Lb(x));
```

NOTA: En este caso el polinomio es de grado menor que dos por estar los tres puntos alineados.

```
--> /*EJEMPLO 21: Vamos a calcular mediante este proceso el
    polinomio interpolador que pasa por los puntos
    (1,-3),(3,-1),(-2,0) y (4,2)*/

    Lc0(x):=(((x-3)*(x+2)*(x-4))/((1-3)*(1+2)*(1-4)))$
    Lc1(x):=(((x-1*(x+2)*(x-4))/((3-1)*(3+2)*(3-4))))$
    Lc2(x):=(((x-1)*(x-3)*(x-4))/((-2-1)*(-2-3)*(-2-4)))$
    Lc3(x):=(((x-1)*(x-3)*(x+2))/((4-1)*(4-3)*(4+2)))$

    Lc(x):=-3*Lc0(x)-Lc1(x)+0*Lc2(x)+2*Lc3(x);
    ratsimp(Lc(x));
```

```
--> /*EJEMPLO 22: Hallemos el polinomio interpolador de la función
    f(x)=(x³-x²+6)/(1+x²) que pasa por los puntos (-1,f(-1)),
    (0,f(0)),(1,f(1)),(2,f(2))*/

    /*En primer lugar definimos la función f(x) y calculamos sus
    valores en los puntos pedidos*/

    f(x):=(x^3-x^2+6)/(1+x^2);
    f(-1);f(0);f(1);f(2);

    /*es decir, se trata de calcular el polinomio de interpolación
    que pasa por (-1,2),(0,6),(1,3),(2,2). Para ello actuamos como
    en los ejemplos anteriores:*/

    Ld0(x):=(((x-0)*(x-1)*(x-2))/((-1-0)*(-1-1)*(-1-2)))$
    Ld1(x):=((x+1)*(x-1)*(x-2))/((0+1*(0-1)*(0-2)))$
    Ld2(x):=(((x+1)*(x-0)*(x-2))/((1+1)*(1-0)*(1-2)))$
    Ld3(x):=(((x+1)*(x-0)*(x-1))/((2+1)*(2-0)*(2-1)))$

    Ld(x):=2*Ld0(x)+6*Ld1(x)+3*Ld2(x)+2*Ld3(x)$
    ratsimp(Ld(x));
```

```
--> /*EJEMPLO 22 (Cont.): Y podemos comprobar que este polinomio
    efectivamente pasa por (-1,2),(0,6),(1,3),(2,2):*/
    Ld(-1);Ld(0);Ld(1);Ld(2);
```

EJERCICIO 8: Calcular el polinomio interpolador de Lagrange que pasa por los puntos (3,1),(2,3),(-1,2),(-2,0)..

EJERCICIO 9: Calcular el polinomio interpolador que pasa por los puntos (3,f(3)),(-2,f(-2)),(-1,f(-1)),(0,f(0)), donde f(x) = e^x.

EJERCICIO 10: Se sabe que una función f pasa por los puntos (0,5), (3,-2),(1,-1),(4,0). Utilizar el polinomio interpolador de Lagrange que pasa por dichos puntos para estimar el valor de la función en x=2 .