PRÁCTICA 1

Introducción a *MAXIMA*

1. Breve historia

Maxima es un programa para realizar cálculos matemáticos simbólicos y numéricos que es capaz de manejar expresiones algebraicas, derivar, integrar o realizar gráficos.

Los orígenes de Maxima están en el año 1967 en el Laboratorio de Inteligencia Artificial del Instituto Tecnológico de Massachussets. Poco a poco fue perdiendo popularidad y uso debido a la competencia de otros programas como *Maple* o *Mathematica*, pero en 1998 William Schelte, de la Universidad de Texas consiguió que una versión de Máxima se distribuyese bajo licencia GNU-GPL (Licencia Pública General). Actualmente el programa es mantenido y mejorado por varias personas de distintos países ayudados por otros muchos interesados en Máxima.

Por todo esto, tanto el código fuente como los manuales son de libre acceso a través de la web del proyecto (http://maxima.sourceforge.net).

En un principio Maxima era sólo un potente motor de cálculo pero su interfaz gráfica era como una consola de texto. Con el tiempo han ido apareciendo diversos entornos de ejecución como el que nosotros usaremos *wxMaxima*, que permite al usuario ejecutar Maxima de forma indirecta e interactuar con el programa.

La interfaz *wxMaxima* tiene versiones tanto para Linux como para Windows. Integra elementos específicos para la navegación de ayuda, introducción de matrices, creación de gráficas, cálculo de límites, derivadas, integrales, etc.

Los presentes apuntes han sido confeccionados con el manual de J. Rafael Rodriguez Galván, Universidad de Cádiz.

2. El entorno *wxMaxima*

Se comienza con una pantalla de bienvenida:

wxMaxima 0.8.0 http://wxmaxima.sourceforge.net
Maxima 5.17.0 http://maxima.sourceforge.net
Using Lisp GNU Common Lisp (GCL) GCL 2.6.8 (aka GCL)
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.

Ya está a la espera de que hagamos la primera entrada que se llamará (% i1) a la que responderá con la primera salida (% o1).

La ventana de *wxMaxima* está dividida en varias secciones:

1.- Barra de menús.
2.- Barra de iconos.
3.- Área de trabajo de entradas y salidas.
4.- Área de botones de rápido acceso.

Pinchamos en el área de trabajo y ya podemos empezar: Al acabar una línea, si queremos ejecutarla deberemos apretar Mayúsculas+Intro. El " ; " , al final de nuestra entrada, aparecerá automáti-camente.

Prueba las siguientes entradas:

(%i1) 34+56; (No hace falta que pongas " ; ", sólo Mayús+Intro)
(%o1) 90

Asignamos un valor a una variable:

(%i2) x:34.22;y:12;x*y+3;
(%o2) 34.22
(%o3) 12
(%o4) 413.64

Si no queremos que aparezca en pantalla los resultados de algunas operaciones utilizaremos el "$" para acabar la línea en lugar de " ; ".

(%i11) x:345$ (El paso de línea sólo con Intro)
y:22$
x/y-7;
(%o13) 191/22

Observa que Maxima trabaja con aritmética racional y nos devuelve una fracción. Si deseamos una expresión numérica (por defecto con 16 cifras decimales) hemos de añadir a la entrada *"**,numer**"*. Prueba:

(%i14) x:345$
y:22$
x/y-7, numer;
(%o16) 8.681818181818182

O bien podías escribir *"**%, numer**"*, que devolverá el resultado numérico de la salida (output) anterior. En este sentido se puede hacer referencia a la salida 23 escribiendo "**% o 23**" o a la entrada 15 escribiendo "**% i 15**".

Maxima también trabaja con expresiones simbólicas:

(%i17) a+b+a/b;
(%o17) b+a/b+a

Y podemos usar la orden " **ratsimp**" para simplificar expresiones racionales o usar el botón rápido "**simplificar**", que por defecto lo aplicará a la última salida:

(%i18) ratsimp(%);
(%o18) (b^2+a*b+a)/b

Hemos de tener cuidado, cuando empleemos letras, que alguna variable no tenga ya un valor asignado anteriormente. Si deseamos borrarlo:

(%i19) kill(x,y);
(%o19) done

También se podía haber empleado el menú superior Maxima -> "Borrar variable".

Ejercicio: Ahora vamos a realizar varios ejercicios con el fin de ilustrar el funcionamiento de wxMaxima:

a) Descomponer el valor de 11! En producto de factores primos.
   Orden "**factor**"
b) Factorizar el polinomio $x^4 - 1$.
   Orden "**factor**"
c) Volver a multiplicar los factores obtenidos en b).
   Orden "**expand(%)**"
d) Simplificar la fracción algebraica: $(x^6 - 1) / (x^2 + x + 1)$
   Orden "**ratsimp**"
e) Resolver la ecuación: $(x^6 - 1) / (x^2 + x + 1) = 0$
   Orden "**solve(ecuación=0)**"

También se podía haber utilizado la barra de menús: "Simplificar" -> "Factorizar expresión", "Simplificar" -> "Expandir expresión" y los botones rápidos "Simplificar" y "Resolver". Este último nos abrirá un cuadro en el que rellenar la ecuación y las variables incógnita.

Si con la orden "*solve*" en lugar de una ecuación introducimos un polinomio, este será tomado como ecuación homogénea, es decir "polinomio=0".

Vemos que wxMaxima pone entre corchetes "[…]" las ecuaciones, las variables y las soluciones que Maxima trata como listas de datos.

Utilizando el menú "Ecuaciones" -> "Resolver sistema lineal" se resuelven ecuaciones lineales. Prueba con el sistema: $x + y = 0$, $2x + 3y = -1$. Primero se abre un cuadro en el que te piden el número de ecuaciones y luego otro para que escribas una ecuación en cada línea y otra línea para que pongas las incógnitas. ¡Cuidado, debes poner "2*x" para expresar "2x".

3. Trabajo con números

3.1. Enteros

En wxMaxima se trabaja con números enteros de cualquier longitud. Prueba:

(%i20) 100!;
(%o20)933262154439441526816992388562[98 digits]9168640000000...

Maxima nos dice que hay una zona intermedia [98 digits] que no aparece en la pantalla. Si deseamos verlos, activamos el menú "Maxima" -> "Cambiar pantalla 2D" y elegimos el algoritmo "ascii".

(%i21) set_display('ascii)$

(%i22) 100!;
(%o22)3326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000

Podemos utilizar la función "**is**" para comprobar si una proposición lógica es cierta o falsa. Prueba:

(%i23) is(36<4!);
(%o23) false

3.2. Racionales

Con los números se pueden emplear las operaciones +, -, *, / y ^siempre teniendo en cuenta la prioridad de las operaciones y el uso de los paréntesis. Prueba:

4 - 1/2
(4 - 1)/2
1/2 + 4
1/(2+4)
1+5*6+7
(1+5)*6+7
1+5*(6+7)
(1+5)*(6+7)

3.3. Reales

Hay algunos números reales predefinidos que se deben emplear así: "%pi", "%e" y otros irracionales como las raíces cuadradas "sqrt" de números reales. Prueba:

(%i24) (%pi+sqrt(2))/%e+1;
- 1
(%o24) %e (%pi + sqrt(2)) + 1

La respuesta se puede representar:

-En coma flotante con precisión fija: orden "flota" o bien ",numer" o bien en el menú "Numérico" -> "A real(float)". Prueba con el resultado anterior.

(%i25) float(%);
(%o25) 2.675987444813811

Nota: Si en una operación interviene un número en representación decimal el resultado saldrá en ese formato. Prueba:

(%i26) 34/5+25.0;
(%o26) 31.8

-En coma flotante de precisión no fija: La precisión se fija con la variable "fpprec" (*float point precision*, por defecto es 16) o en el menú "Numérico" -> "Establecer precision".

En este formato para representar un número real podemos utilizar la función "bfloat" o ir al menú "Numérico" -> "A real grande (bigfloat)". Prueba:

```
(%i27) fpprec:60$
       bfloat(%e);
(%o28)
2.7182818284590452353602874713526624977572470936999595749669 7b0
```

Los caracteres finales "bN" significan "multiplicar por 10^N. Prueba:

```
(%i29) 100+sin(1b0);
(%o29)
1.0084147098480789650665250232163029899962256306079837106567 3b2
```

3.4. Complejos

Algunas de las órdenes que se emplean para trabajar con números complejos son las que aparecen en los siguientes ejemplos:

c : 2/3 – 4*%i;	definimos el complejo c
realpart(c);	parte real de c
imagpart(c);	parte imaginaria de c
exp(c);	complejo c en forma exponencial
demoivre(c);	expresión trigonométrica o binómica de c

3.5. Potencias y logaritmos

Para expresar una potencia se usa el signo "^". Para potencias de base "e" podemos emplear la función "exp". Ejemplos:

```
(%i30) 1/3+(1/3)^2+(1/3)^3;
                       13
(%o30)                 --
                       27
(%i31) is(6^5=3^5*4^5);
(%o31)                 false

(%i32) is((m/n)^4=m^4/n^4);
(%o32)                 true

(%i33) exp(4)-%e^4;
(%o33)                  0
```

Fíjate que es lo mismo: asignar a la variable r el valor de raíz cuadrada de 2 así

r:sqrt(2) que **r: 2^(1/2)**

Es importante, si se desea simplificar expresiones con potencias o radicales emplear el menú "Simplificar" o alguno de los botones rápidos. Puedes probar:

(%i34) expresion:a/b+sqrt(a/b);

(%i36) ratsimp(expresion); simplificar expresión

(%i37) radcan(expresion); simplificar radicales

Ejercicio: Prueba ahora las mismas órdenes con **(a^b)^c.**

Para el cálculo de logaritmos tenemos la función "logaritmo neperiano" que se expresa "log"

(%i38) x:%e^6$
log(x);
(%o39) 6

Para calcular logaritmos en cualquier otra base debemos definir la siguiente función: "logaritmos en base b", así: definimos la función "logb" empleando ":="

logb(x,b) := log(x) / log(b)$ no devolverá respuesta.
logb(512,2); logaritmo de 512 en base 2.
bfloat(%); resultado decimal, o bien float(%)

Ejercicio: Es interesante que te fijes en algunos comandos del menú "Simplificar" para el trabajo con logaritmos como "Expandir logaritmos" o "Contraer logaritmos". Pruébalos con la expresión **log(a/b).**

3.6. Divisibilidad

La orden factor(N) devuelve la descomposición del número N en producto de factores primos. Prueba a descomponer 21600.

(%i40) factor(21600);
5 3 2
(%o40) 2 3 5

Para averiguar si un número es primo la orden "primep":

(%i41) primep(17);
(%o41) true

4. Trabajo con listas

Una lista es una serie de expresiones separadas por comas y encerradas entre corchetes. Como ejemplo:

(%i45) lista:[1,3*x-2,Juan];
(%o45) [1, 3 x - 2, Juan]

Para acceder a un elemento de una lista: al primero, segundo, ..., último:

```
(%i46) first(lista);
(%o46)                    1

(%i47) second(lista);
(%o47)                  3 x - 2

(%i48) last(lista);
(%o48)                  Juan
```

Obteniéndose los mismos resultados que así:

```
lista[1]
lista[2]
lista[3]
```

Algunas operaciones aritméticas pueden actuar sobre listas:

```
(%i50) l1:[1,3,5,7]$
       l2:[-1,-3,-5,-7]$
       l1+l2;
(%o52)                  [0, 0, 0, 0]
```

También tiene sentido:

```
(%i53) sqrt(l1);
(%o53)          [1, sqrt(3), sqrt(5), sqrt(7)]
```

Pero no calcula ni devuelve nada interesante si hacemos:

```
(%i54) cos(l1);
(%o54)                cos([1, 3, 5, 7])

(%i55) log(l1);
(%o55)                log([1, 3, 5, 7])
```

Se puede eliminar un elemento de una lista con la orden “delete”, o añadir un elemento al principio con “cons”, o unir dos listas con “append”. Hazlo tu:

```
s1 : delete(1, lista);
s2 : cons(%e, lista);
append(s1,s2);
```

Otras funciones interesantes para el manejo de listas:
makelist(término general, variable índice, rango inf., rango super.). Por ejemplo:

```
(%i57) makelist(3^k,k,0,4);
(%o57)                [1, 3, 9, 27, 81]
```

También podemos emplearla así:

```
(%i58) f(x):=3*x+1$
       l:makelist(f(k),k,0,3);
(%o59)                [1, 4, 7, 10]
```

Podemos emplear la orden "apply" a todos los elementos de una lista:

apply(min,l); devolverá el mínimo de l: 1
apply(max,l); devolverá el máximo de l: 10
apply("+",l); devolverá la suma de todos los elementos: 22

O aplicar una función a todos los elementos de una lista con la orden "map", o en el menú "Algebra" -> "Corresponder a lista":

```
(%i64) z:makelist(k*%pi/4,k,0,8)$
       map(sin,z);
```

Podemos definir polinomios

```
(%i66) p:x^2-4$
       q:2*x^3+x^2-3*x-1;
```

Y luego operar con ellos:

```
       3*q + 2*p;
       p*q
```

Ejercicio: *Se puede calcular el m.c.d y el m.c.m. de dos polinomios en el menú* "Análisis" -> "Máximo común divisor" *y* "Mínimo común múltiplo" *o con las funciones* "gcd" y "lcm". *Puedes probarlas con los polinomios* p : x^3 – 7x + 6 y q : x^2 + 3x – 4. *Para que el mínimo común múltiplo funcione se debe cargar primero una función mediante* **load("functs")**.

La división entre polinomios devuelve una lista con el cociente y el resto de la división, pruébalo así:

```
(%i69) divide(q,p);
(%o69)              [2 x + 1, 5 x + 3]
```

Para dar valores a un polinomio existen varias posibilidades:

a) Sustituir la variable por un valor deseado, por ejemplo:

```
(%i70) p,x=1;
(%o70)                    - 3
```

b) Mediante la función "subst" o con el menú "Simplificar" -> "Sustituir"

```
(%i72) subst(1,x,p);
```

(%o72) - 3

c) Definiendo una función polinómica

```
(%i73) f(x):=x^2-4$
       f(1);
(%o74)                    - 3
```

También la orden "map" tiene su uso aquí:

```
(%i75) map(f,[-3,1,2,5]);
(%o75)                  [5, - 3, 0, 21]
```

Para calcular los ceros o raíces de un polinomio emplearemos la orden "solve", resultando idénticas las siguientes órdenes:

```
solve(p);
solve(p,x);
solve(p=0,x);
```

Si un polinomio tiene raíces reales y complejas, con la orden "allroots" obtendremos todas y con la orden "realroots" tendremos sólo las raíces reales, así:

```
p : x^3 + x;

(%i77) allroots(p);
(%o77)          [x = 0.0, x = 1.0 %i, x = - 1.0 %i]

(%i78) realroots(p);
(%o78)                  [x = 0]
```

Otras órdenes para manejar polinomios se observan en los siguientes ejemplos:

```
p : x^3 + x$
factor(p);                  salida: x(x^2 + 1)
expand((x + b)^3);          salida:  x^3 + 3bx^2 + 3b^2x + b^3
solve(%,x);                 salida:  x = -b
```

Las órdenes hasta ahora conocidas para polinomios siguen siendo válidas para trabajar con fracciones algebraicas.

6. Ecuaciones, sistemas e inecuaciones

Se define una ecuación con el signo "=" y se le puede asignar un símbolo o nombre:

```
(%i80) ec:x+3=(2*x^2+x-3)/(x-1);
```

Para acceder al primer o segundo miembro "lhs" y "rhs" ("left" y "right hand side") o bien las funciones "first" y "second", así es lo mismo:

first(ec); que **lhs(ec);**
second(ec); que **rhs(ec);**

```
(%i81) is(rhs(ec)=second(ec));
(%o81)                         true
```

Para resolver una ecuación empleaos la orden "solve" y si no es posible resolverla así nos ayudaremos de otros métodos para obtener una solución numérica aproximada. Ejemplo:

```
(%i82) ec:exp(x)=x$
       solve(ec);
                                    x
(%o83)                       [x = %e  ]
```

Los sistemas de ecuaciones se escriben como listas de ecuaciones (entre corchetes) y si son lineales se podrán resolver mediante la orden "solve" o mediante el menú "Ecuaciones" -> "Resolver sistemas lineales". Si hay varias variables hemos de escribir aquellas en las que esperamos la solución. Así:

```
(%i20) sis:[x+y=c,2*x-y=2*c];      c es un parámetro
(%o20) [y+x=c,2*x-y=2*c]

(%i21) solve(sis, [x,y]);
(%o21) [[x=c,y=0]]
```

En sistemas indeterminados la solución depende de un parámetro que en este caso será %r1, veamos:

```
(%i23) solve([x-y+3*z=3,2*x+y-z=2,3*x+2*z=5]);
Dependent equations eliminated:  (1)
(%o23) [[z=-(3*%r1-5)/2,y=-(7*%r1-9)/2,x=%r1]]
```

Cuando se ha de resolver sistema no lineales, intentará dar una resolución simbólica, pero a veces no es posible y entra en funcionamiento (sin que nos demos cuenta) la orden "algys" dando una solución numérica o incluso sin obtener la solución como en estos casos:

```
(%i2) solve([x^4-y^2=1,x^2+y=2]);
(%o2) [[y=3/4,x=-sqrt(5)/2],[y=3/4,x=sqrt(5)/2]]

(%i3) solve([x^6-y^3=1,x^2+y^2=2]);
(%o3)[[y=0.89565802760013,x=-
1.094439117929051],[y=0.89565802760013,x=1.094439117929051],[y=0.2447509434
5397*%i-1.119432090612319,x=-0.29025378661936*%i- ……..

(%i4) solve([sin(x)+y=0,x+y^2=1]);
```

`algsys' cannot solve - system too complicated.
-- an error. To debug this try debugmode(true);

Para resolver inecuaciones primero resolvemos la correspondiente ecuación y luego estudiamos el signo por intervalos. Hazlo para resolver la inecuación

$x^2 – 8x + 15 < 0$

```
(%i5) p(x):=x^2-8*x+15;
(%o5) p(x):=x^2-8*x+15

(%i7) solve(p(x)=0);
(%o7) [x=3,x=5]

(%i8) p([1,4,7]);
(%o8) [8,-1,8]
```

Del último resultado se deduce que la solución es el intervalo **(3, 5)**.

7. Sucesiones y límites

Las sucesiones se definen mediante su término general a[n] utilizando el operador ":=".

```
(%i1) a[n]:=n/(3*n-1);
(%o1) a[n]:=n/(3*n-1)

(%i2) a[5];
(%o2) 5/14

(%i3) makelist(a[i],i,1,12);
(%o3) [1/2,2/5,3/8,4/11,5/14,6/17,7/20,8/23,9/26,10/29,11/32,12/35]
```

En las definiciones por recurrencia se emplea ":" para asignar los primeros valores de la sucesión.

```
(%i4) b[1]:2;
(%o4) 2

(%i5) b[n]:=3*b[n-1]+1;
(%o5) b[n]:=3*b[n-1]+1

(%i6) makelist(b[h],h,1,12);
(%o6) [2,7,22,67,202,607,1822,5467,16402,49207,147622,442867]
```

Ejercicio: Escribe, por el método de recurrencia los 10 primeros términos de la sucesión de Fibonacci.

El límite de una sucesión se calcula con la función "limit" o en el menú "Análisis" -> "Calcular límite". Se abrirá un cuadro para rellenar en el que escribiremos

la expresión de la sucesión, el nombre de la variable y el valor hacia el que esta tiende (puede ser +infinito o –infinito) aunque para sucesiones sea +infinito. Ejemplo:

(%i12) a[n]:=(3*n-sqrt(9*n^2-3))/n;
(%o12) a[n]:=(3*n-sqrt(9*n^2-3))/n

(%i13) limit(a[n], n, inf);
(%o13) 0

Ejercicio: Calcula el límite de las sucesiones $(-1)^n$ *y* $n(-1)^n$.

8. Funciones. Definición y límites.

Maxima cuenta con las funciones habituales ya definidas como son: “sin”, “cos”, “tan”, “asin”, “acos”, “atan” y también “csc”, “sec”, “cot” (cosecante, secante, cotangente), “abs” (valor absoluto), “binomial” (**binomial(m,n)** es el número combinatorio m sobre n)

Una función se define **f(x) := x^2 - 3*x + 2**, es decir con “:=”. Hazlo y calcula después f(2), f(-1/2) y f(3.45) y observa los distintos resultados.

(%i14) f(x):=x^2-3*x+2;
(%o14) f(x):=x^2-3*x+2

(%i15) f(2);
(%o15) 0

(%i16) f(-1/2);
(%o16) 15/4

(%i17) f(3.45);
(%o17) 3.5525

Para calcular el **límite** ve al menú “Análisis” -> “Calcular Límite”, se abrirá un cuadro para rellenar que es el mismo que para sucesiones pero en este caso es posible usar puntos “especiales” como “pi”, “e”, “infinito” o “-infinito” y puedes calcular la dirección, por “izquierda”, “derecha” o “ambos lados”.

(%i19) limit(f(x),x,-3);
(%o19) 20

Vamos a calcular límites laterales con los añadidos “plus” (por la derecha) y “minus” (por la izquierda).

(%i22) limit(1/(x-3),x,3);
(%o22) infinity

(%i23) limit(1/(x-3),x,3,plus);
(%o23) inf

(%i24) limit(1/(x-3),x,3,minus);
(%o24) –inf

También se pueden definir funciones compuestas:

(%i25) f(x):=1-1/x;
(%o25) f(x):=1-1/x
(%i26) g(x):=sin(x);
(%o26) g(x):=sin(x)

(%i27) h(x):=f(g(x));
(%o27) h(x):=f(g(x))

(%i28) h(x);
(%o28) 1-1/sin(x)

O una función definida a trozos: Mediante la orden "if".

(%i30) f(x):=if x<=0 then cos(x) else log(x+2);
(%o30) f(x):=if x<=0 then cos(x) else log(x+2)

(%i32) [f(-%pi/2),f(-%pi/4),f(0),f(2.0)];
(%o32) [0,1/sqrt(2),1,1.386294361119891]

Ejercicio: Expresa la siguiente función definida a trozos y estudia su continuidad en 0 y en 2:

$$f(x)=2x-3 \text{ si } x<0,\ x^2-1 \text{ si } 0<=x<2,\ 3 \text{ si } x>2$$

9. Gráficas de funciones de una variable (coordenadas cartesianas)

Para dibujar la gráfica de una función empleamos la orden "wxplot2D" o con el botón de acceso rápido "Gráficos 2D" o en el menú "Plot" -> "Gráficos 2D". Hemos de expresar la función o lista de funciones a representar, y otra lista con la variable independiente y el rango de esta. Prueba con estas órdenes:

(%i33) wxplot2d(cos(3*x), [x,-2*%pi,2*%pi])$

(%i39) wxplot2d([x^2,sqrt(x)], [x,0,1])$

10. **Derivadas e integrales**

Calculamos derivadas usando la función "diff" (o con "Análisis" -> "Derivar").

(%i49) f(x):=a*x*%e^(4*x);
(%o49) f(x):=a*x*%e^(4*x)

(%i50) diff(f(x),x,1);

```
(%o50) 4*a*x*%e^(4*x)+a*%e^(4*x)

(%i51) diff(f(x),x,2),a=-1 /*segunda derivada para a=-1 */;
(%o51) -16*x*%e^(4*x)-8*%e^(4*x)
```

Para integrales lo hacemos con la orden "integrate" (o con "Análisis" -> "Integrar") con o sin extremos según que la integral sea indefinida o definida, respectivamente.

```
(%i52) g(x):=x/(x^2+4);
(%o52) g(x):=x/(x^2+4)

(%i53) diff(g(x),x);
(%o53) 1/(x^2+4)-(2*x^2)/(x^2+4)^2

(%i54) ratsimp(%);
(%o54) -(x^2-4)/(x^4+8*x^2+16)

(%i55) integrate(%, x);
(%o55) x/(x^2+4)

(%i56) integrate(5*x*sin(2*x), x, 0, %pi/2);
(%o56) (5*%pi)/4
```

También haremos desarrollos de Taylor mediante la función "taylor" que tiene como argumentos, la función, la variable, el centro y el orden.

```
(%i57) h:(%e^x-1)/x;
(%o57) (%e^x-1)/x

(%i58) taylor(h,x,0,4);
(%o58) 1+x/2+x^2/6+x^3/24+x^4/120+...
```

11. Ejercicios finales (para entregar en la próxima sesión)

1.- Escribe el número "pi" con 100 cifras.

2.- Expresa el número complejo $\frac{1}{2}+\frac{3}{4}i$ *en forma exponencial y trigonométrica.*

3.- ¿Cómo calcularías el $\log_3 100$ *?*

4.- Descompón el número 2475 en producto de sus factores primos.

5.- Expresa como suma de sus fracciones simples la fracción: $\frac{x^2+x-1}{x^3-2x^2+x}$.

6.-Halla el dominio de definición de la función $f(x)=\frac{x^2+x-1}{x^3-2x^2+x}$ *y sus límites cuando x->1 y cuando x->0.*

7.- Confecciona una lista en la que aparezcan los 20 primeros términos de la sucesión recurrente $a_1=1,\ \ a_2=2,\ \ a_n=2a_{n-2}-a_{n-1}$.