PRACTICA 3. ANÁLISIS COMPLEJO.

1 Operaciones básicas con números complejos.

Veremos incialmente algunos comandos específicos para trabajar con números complejos. Recordemos que en Maxima la unidad imaginaria se representa por %i.

Podemos comenzar definiendo el número complejo z=3-2i:

```
--> z:3-2*%i;
```

Si necesitamos escribir la parte real de z usaremos

```
--> realpart(z);
```

mientras que para la parte imaginaria

```
--> imagpart(z);
```

Sea entonces w=3+2i . Podemos operar con ambos complejos, y así hacer:

```
--> w:-1+2*%i;
```

Notemos que a veces Maxima nos devuelve la parte imaginaria del complejo antes que la parte real. Para que siempre escriba primero la parte real seguida de la parte imaginaria, haremos:

```
--> powerdisp:true$
    w;
```

```
--> /*podemos calcular la suma o diferencia de ambos,
    que los introducimos como unas nuevas variables,
    que llamamos suma y resta*/

    suma:z+w;
    resta:z-w;
```

Podemos multiplicar complejos como se multiplican cualquier par de nros:

```
--> z*w;
```

Observamos que nos da una expresión literal, por lo que si queremos calcular su resultado haremos:

```
--> expand(%);
```

Lo mismo ocurre si intentamos dividir:

```
--> z/w;
```

y si lo que queremos es que nos de en forma cartesiana, haremos:

```
--> rectform(z/w);
```

Podemos calcular el módulo de z, usando abs(z) o cabs(z);:

```
--> abs(z);
```

```
--> cabs(z);
```

NOTA:Si usamos abs(z) para calcular el módulo del complejo z, obtendremos el valor absoluto de z, y no el resultado dado por cabs(z), que nos da (x^2+y^2)^(1/2). La opción abs(z) suele usarse cuando se utilizan cálculos simbólicos (por ejemplo, calcular límites, derivadas, integrales,...)

Para hallar el conjugado de z haremos:

```
--> conjugate(z);
```

Podemos expresar el complejo z en forma polar |z|·e^(i·arg(z)):

```
--> polarform(z);
```

Observamos que esta última forma nos da el módulo y el argumento de z. No obstante, podemos calcular directamente su argumento (anteriormente ya hemos calculado su módulo) sin necesidad de expresarlo en forma polar. Para ello usaremos:

```
--> carg(z);
```

Podemos calcular potencias naturales de z:

```
--> z^2;
    expand(%);
```

También podemos calcular potencias enteras de z:

```
--> z^(-2);
    expand(%);
    rectform(%);
```

A la hora de calcular raíces complejas, hemos de tener en cuenta que Maxima no las va a obtener todas (sabemos que todo complejo z tiene n raíces n-simas), sino que nos va a dar la primera de ellas (recordemos que las demás siempre tienen el mismo módulo pero se diferencian en el argumento):

```
--> z^(1/2);
    rectform(%);
```

```
--> z^(1/3);
    rectform(%);
```

Podemos calcular la exponencial de z (es decir, e^z):

```
--> exp(z);
    rectform(%);
```

o el logaritmo principal de z:

```
--> log(z);
    rectform(%);
```

Podemos calcular senos y cosenos de z:

```
--> cos(z);
    rectform(%);
```

```
--> sin(z);
    rectform(%);
```

Si preferimos la notación exponencial, "exponentialize()" escribe todo en términos de exponenciales:

```
--> exponentialize(cos(z));
```

y "demoivre()" utiliza senos y cosenos en la salida en lugar de las exponenciales:

```
--> demoivre(cos(z));
```

2 Ecuaciones.

```
--> /*Primero borramos todas las variables que hemos
    usado anteriormente*/

    remvalue(all);
```

En Maxima, una ecuación es una igualdad entre dos expresiones algebraicas escrita con el símbolo =.

```
--> 3*x^2+2*x+x^3-x^2=4*x^2;
```

Podemos asignarle un nombre para referirnos a esa ecuación:

```
--> mi_ecuacion:3*x^2+2*x+x^3-x^2=4*x^2;
```

lo que nos permite operar con ella:

```
--> mi_ecuacion*4-3*x^3;
```

2.1 Resolución de ecuaciones y sistemas.

Maxima puede resolver los tipos más comunes de ecuaciones y sistemas de ecuaciones algebraicas de forma exacta. Por ejemplo, sabe encontrar las raíces de polinomios de grado bajo (2,3 y 4). En cuanto a polinomios de grado más alto o ecuaciones más complicadas, no siempre será posible encontrar la solución exacta. En este caso, podemos intentar encontrar una solución aproximada en lugar de la solución exacta.

La orden "solve" nos da todas las soluciones, ya sean reales o complejas de una ecuación sistema de ecuaciones:

```
--> solve(x^2-3*x+1=0,x);
```

También podemos resolver ecuaciones que dependan de algún parámetro:

```
--> solve(x^3-a*x^2-x^2+2*x=0,x);
```

Podemos obtener la multiplicidad de las raíces:

```
--> solve(x^7-2*x^6+2*x^5-2*x^4+x^3=0);
```

```
--> multiplicities;
```

La orden solve no sólo puede resolver ecuaciones algebraicas:

```
--> solve(sin(x)*cos(x)=0,x);
```

Como cualquiera puede imaginarse, Maxima no resuelve todo. Incluso en las ecuaciones más “sencillas”, los polinomios, se presenta el primer problema: no hay una fórmula en términos algebraicos para obtener las raíces de un polinomio de grado 5 o más. Pero no hay que ir tan lejos. Cuando añadimos raíces, logaritmos, exponenciales, etc., la resolución de ecuaciones se complica mucho. En esas ocasiones lo más que podemos hacer es ayudar a Maxima a resolverlas.

```
--> eq:x+3=sqrt(x+1);
```

```
--> solve(eq,x);
```

```
--> solve(eq^2);
```

También podemos resolver sistemas de ecuaciones. Sólo tenemos que escribir la lista de ecuaciones y de incógnitas. Por ejemplo:

```
--> solve([x^2+y^2=1,(x-2)^2+(y-1)^2=4],[x,y]);
```

Si la solución depende de un parámetro o varios, Maxima utilizará %r1, %r2,... para refererirse a estos. Por ejemplo,

```
--> solve([x+y+z=3,x-y=z],[x,y,z]);
```

¿Qué pasa si el sistema de ecuaciones no tiene solución? Veamos un ejemplo:

```
--> solve([x+y=0,x+y=1],[x,y]);
```

2.2 Resolución aproximada de ecuaciones polinómicas.

Las ecuaciones polinómicas se pueden resolver de manera aproximada. Los comandos allroots (que nos da todas las raíces) y realroots (que solo nos da las raíces reales) están especializados en encontrar soluciones racionales aproximadas de polinomios en una variable:

```
--> eq:x^9+x^7-x^4+x$
    allroots(eq);
```

y si solo nos interesan las que son reales:

```
--> realroots(eq);
```

3 Análisis Complejo I: Funciones analíticas.

Con wxMaxima podemos calcular directamente algunos límites de funciones complejas. En concreto, vamos a poder resolver límites siempre que la función venga dada en términos de z y la misma sea analítica (ya que en este caso usamos, básicamente, los mismos procesos que en el caso de funciones con una variable).
Así, vamos a poder calcular límites de la forma siguiente (es decir, límites que existen)

EJEMPLO 1: Calcular los siguientes límites:

```
--> limit(z/(z+1), z, 1);
```

```
--> limit((%i*z+3)/(z+1), z, -1);
```

NOTA: Para el caso que no exista el límite, o también siempre que la variable z venga expresada en términos de x+%iy, será más aconsejable expresar la función f(z) en la forma
f(z)=u(x,y)+%iv(x,y)
y calcular (como límites de funciones de dos variables) tanto el límite de u(x,y) como el de v(x,y).
Lo vemos en el siguiente ejemplo:

EJEMPLO 2: Calcular el siguiente límite:

```
--> limit((conjugate(z)+%i*z^2)/cabs(z), z, 0);
```

Observamos que nos da un mensaje de error ya que limit es una función de una sola variable, y en este caso, al llevar la función de la que se quiere calcular su límite tanto un conjugado como un valor absoluto (módulo), ésta no va a ser analítica, por lo que para obtener su límite tendremos que expresarla como dos funciones de dos variables reales (parte real u(x,y) y parte imaginaria v(x,y)):

```
--> z:x+%i*y;
```

```
--> (conjugate(z)+%i*z^2)/cabs(z);
```

```
--> expand(%);
```

```
--> f:rectform(%);
```

y elegimos la parte real u(x,y) y la parte imaginaria v(x,y):

```
--> u:realpart(f);
```

```
--> v:imagpart(f);
```

Entonces, y usando las técnicas de cálculo de límites dobles estudiadas en 1er curso, podremos probar la no existencia de los límites de u(x,y) y/o v(x,y) (a través de límites direccionales o mediante un cambio a polares). También puede verse como hacerlo con wxMaxima en la página 111 del manual que se encuentra en la web:
http://es.scribd.com/doc/52510808/Manual-wxMaxima

Con wxMaxima también podemos estudiar cuando una función compleja es analítica, ya que básicamente consiste en aplicar las ecuaciones de Cauchy-Riemann, que conlleva el cálculo de derivadas parciales de las funciones parte real u(x,y) y parte imaginaria v(x,y):

EJEMPLO 3. Estudiar si la función f(z)=abs(z)^2 admite derivada:

```
--> z:x+%i*y$
    cabs(z)^2;
```

```
--> u:realpart(cabs(z)^2);
    v:imagpart(cabs(z)^2);

--> /*calculamos d/dx(u) y d/dy(v)*/
    diff(u,x);
    diff(v,y);

--> /*calculamos d/dx(v) y -d/dy(u)*/
    diff(v,x);
    diff(u,y)*(-1);
```

Observamos que no se verifican las ecuaciones de C-R salvo si x=y=0. Por tanto, f'(z) no existe salvo si z=0.

EJEMPLO 4. Estudiar si la función f(z)=exp(z) admite derivada:

```
--> exp(z);
    rectform(%);

--> u:realpart(exp(z));
    v:imagpart(exp(z));

--> /*calculamos d/dx (u) y d/dy (v)*/
    diff(u,x);
    diff(v,y);

--> /*calculamos d/dx (v) y -d/dy (u)*/
    diff(v,x);
    diff(u,y)*(-1);
```

Observamos que se verifican las ecuaciones de C-R, por lo que f(z) es derivable. Además sabemos que f'(z)= diff(u,x)+%i*diff(v,x):

```
--> derivada_f:diff(u,x)+%i*diff(v,x);
```

Hemos llegado al conocido resultado de que si f(z)=exp(z), entonces f'(z)=f(z). A esta misma conclusión podríamos haber llegado si hubiésemos realizado

```
--> remvalue(all)$
    /*Primero hemos borrado todas las variables antes de calcular
    la derivada tomando z como una variable*/
    diff(exp(z),z);
```

De nuevo en este ejemplo hemos calculado una derivada como si la función a derivar no fuese una función compleja, sino que fuese una función con una sola variable. Esto siempre puede hacerse en caso que la función a derivar sea analítica, mientras que si no lo es (porque, por ejemplo, lleva un módulo o un conjugado en su definición) no lo vamos a poder hacer (como hemos visto en el Ejemplo 3 anterior).

Otro típico ejercicio de variable compleja relacionado con la derivación consiste en dada la parte real u(x,y) de una función analítica, hallar su función armónica conjugada (es decir, otra función v(x,y) tal que la función dada por f(z)=u(x,y)+iv(x,y) es analítica). Lo vemos en el siguiente ejemplo.

EJEMPLO 5. Dada la función u(x,y)=3*x^2*y+2*x^2-y^3-2*y^2, probar que es armónica y hallar su armónica conjugada:

```
--> u(x,y):=3*x^2*y+2*x^2-y^3-2*y^2;
```

```
--> diff(u(x,y),x,2)+diff(u(x,y),y,2);
```

Como este último resultado da 0, es cierto que u(x,y) es armónica. Para hallar la armónica conjugada, como sabemos que d/dy(v)=d/dx(u), entonces v(x,y)=int(d/dx(u),respecto y)+g(x)

```
--> diff(u(x,y),x);
```

```
--> integrate(diff(u(x,y),x), y);
```

Este último resultado nos da el valor para v(x,y), salvo una cte (que es una función que depende de x, g(x)). Usaremos la segunda de las condiciones de Cauchy-Riemann para calcular g(x). Por la segunda de estas ecuaciones se tiene que d/dx(v)=-d/dy(u). Calculamos primero d/dx(v), para lo que derivaremos respecto de x el último resultado obtenido anteriormente:

```
--> diff(integrate(diff(u(x,y),x), y),x,1);
```

Aquí tendremos que sumarle a la expresión anterior g'(x).
El segundo miembro de la igualdad d/dx(v)=-d/dy(u) también lo obtenemos haciendo

```
--> diff(u(x,y),y)*(-1);
```

y para obtener g'(x) hacemos

```
--> diff(u(x,y),y)*(-1)-diff(integrate(diff(u(x,y),x), y),x,1);
```

e integrando este último resultado obtendremos g(x)

```
--> integrate(%, x);
```

Al final resulta que v(x,y)=3*x*y^2+4*x*y-x^3.

4 Análisis Complejo II: Integración.

Como hemos visto en teoría, las integrales en el campo complejo son siempre integrales de línea en el plano, por lo que podremos usar todo lo visto en la práctica 2 (tanto a la hora de representar gráficos de las curvas como a la hora de calcular las integrales de línea). No obstante, existen unos resultados específicos para estas integrales complejas que nos permitirán calcular las mismas sin necesidad de acudir a la definición (es decir, sin necesidad de tener que tomar una parametrización de la curva plana y de resolver la posterior integral de línea). Estos resultados son la independencia del camino de integración (más conocida como regla de Barrow para estas integrales), y, para el caso de curvas cerradas, el teorema de Cauchy-Goursat y las fórmulas integrales de Cauchy (todos estos resultados tienen que ver con que f(z) sea analítica en la región encerrada por la curva). Intentaremos desarrollar todo ésto en los ejemplos que vienen a continuación.

EJEMPLO 6. Representar las curvas complejas abs(z)=2; abs(z-3)=2:

```
--> load(draw);
```

```
--> /*la curva abs(z)=2 equivale a la circunferencia de ecuación
    x^2+y^2=4, que es la que representamos a continuación*/

    draw2d(implicit(x^2+y^2=4,x,-2,2,y,-2,2));
```

```
--> /*la curva abs(z-3)=2 equivale a la circunferencia de ecuación
    (x-3)^2+y^2=4, que es la que representamos a continuación*/

    draw2d(implicit((x-3)^2+y^2=4,x,0,6,y,-2,2));
```

EJEMPLO 7. Representar las curvas complejas z=2*exp(i*theta); z=3+2*exp(i*theta) (en este caso las curvas vienen dadas por sus coordenadas polares; nosotros las representaremos en coordenadas paramétricas):

```
--> /*la curva z=2*exp(i*theta) equivale a la parametrización
    (2*cos(theta),2*sin(theta)), que es la que representamos
    a continuación*/

    plot2d([[parametric, 2*cos(theta), 2*sin(theta),
    [theta, 0, 2*%pi], [nticks, 300]]])$
```

```
--> /*la curva z=3+2*exp(i*theta) equivale a la parametrización
    (3+2*cos(theta),2*sin(theta)), que es la que representamos
    a continuación*/

    plot2d([[parametric, 3+2*cos(theta), 2*sin(theta),
    [theta, 0, 2*%pi], [nticks, 300]]])$
```

Notemos que las dos curvas de este Ejemplo 7 coinciden con las del Ejemplo 6.

EJEMPLO 8. Calcular, usando la definición de integral compleja, la integral de f(z)=conjugate(z) a lo largo de la parte derecha de la circunferencia abs(z)=2:

```
--> /*vamos a calcular la integral de f(z)=conjugate(z) a lo largo
    de la mitad derecha del círculo abs(z)=2*/

    /*La parametrización de la curva viene dada por
    curva(t)=(2cos(t),2sin(t)), donde t varia en [-%pi/2,%pi/2]*/

    plot2d([[parametric, 2*cos(t), 2*sin(t),
    [t, -%pi/2,%pi/2], [nticks, 300]]])$
```

```
--> /*La función a integrar será f(curva(t))*d/dt(curva(t)) en el
    intervalo [-%pi/2,%pi/2]*/

    curva(t):=2*cos(t)+%i*2*sin(t);
```

```
--> curva(t)*diff(curva(t),t,1);
```

```
--> integrate(%, t, -%pi/2, %pi/2);
```

EJEMPLO 9: Vamos a calcular la integral de f(z)=z^2 entre los puntos z1=0 y z2=1+i. Lo vamos a hacer:
a) A través de la recta que une ambos puntos.
b) A través de la curva y=x^3 (que también pasa por ambos puntos).
c) A través de la regla de Barrow.

Para el apartado (a):

```
--> /*La parametrización de la recta viene dada por
    curva(t)=(t,t)), donde t varia en [0,1]*/
    /*La función a integrar será f(curva(t))*d/dt(curva(t)) en el
    intervalo [0,1]*/

    curva(t):=t+%i*t;
```

```
--> (curva(t))^2*diff(curva(t),t);
```

```
--> integrate(%, t, 0, 1);
```

Para el apartado (b):

```
--> /*La parametrización de la curva viene dada por
    curva(t)=(t,t^3)), donde t varia en [0,1]*/

    /*La función a integrar será f(curva(t))*d/dt(curva(t)) en el
    intervalo [0,1]*/

    curva(t):=t+%i*t^3;
```

```
--> (curva(t))^2*diff(curva(t),t);
```

```
--> integrate(%, t, 0, 1);
```

Observamos que el resultado de (a) y (b) es el mismo a pesar de cambiar de curva. Esto es debido a que la función f(z)=z^2 es analítica, como podemos comprobar ya que verifica las ecuaciones de Cauchy-Riemann:

```
--> z:x+%i*y;
```

```
--> u:realpart(z^2);
    v:imagpart(z^2);
```

```
--> /*calculamos d/dx(u) y d/dy(v)*/
    diff(u,x);
    diff(v,y);
```

```
--> /*calculamos d/dx(v) y -d/dy(u)*/
    diff(v,x);
    diff(u,y)*(-1);
```

Por tanto también podremos calcular esta integral como nos pide el apartado (c), es decir, usando la regla de Barrow: Maxima no calcula integrales definidas si los límites de integración son complejos, por lo que procederemos a calcular la primitiva de f(z) y evaluarla en los puntos 0 y 1+i:

```
--> remvalue(all)$
    integrate(z^2, z);
```

```
--> f(z):=z^3/3;
```

```
--> f(1+%i)-f(0);
    rectform(%);
```

5 Análisis Complejo III: Series. Ceros y polos. Residuos.

wxMaxima nos permite obtener los ceros y los polos (que sabemos son conceptos relacionados) de funciones complejas. También podremos obtener el valor de los residuos en los polos.

Los ceros de una función compleja no son sino las raíces que tiene la propia función; mientras que los polos suelen ser las raíces del denominador (siempre que no anulen el numerador, porque entonces puede ser que tengamos una singularidad evitable). Veamos un ejemplo:

EJEMPLO 10: Vamos a definir la función
f(z)=(z^2+z-1)/(z^3-1.5*z^2-0.5*z-1)
y ver como podemos elegir su numerador y denominador:

```
--> remvalue(all)$
    f(z):=(z^2+z-1)/(z^3-1.5*z^2-0.5*z-1);
```

```
--> num(f(z));
```

```
--> denom(f(z));
```

EJEMPLO 11: Si queremos calcular los ceros de
f(z)=(z^2+z-1)/(z^3-1.5*z^2-0.5*z-1)
haremos

```
--> remvalue(all)$
    f(z):=(z^2+z-1)/(z^3-1.5*z^2-0.5*z-1);
```

```
--> solve([f(z)=0], [z]);
```

mientras que para calcular las singularidades (polos) haremos

```
--> solve([z^3-1.5*z^2-0.5*z-1=0], [z]);
```

Como observamos que no hay repetición entre las raíces del numerador y denominador, tanto los ceros como los polos son simples.

Si queremos comprobar si estos polos están, por ejemplo, en el interior del círculo unidad, obtendremos su módulo y preguntamos si éste es menor que la unidad:

```
--> cabs(-(sqrt(7)*%i+1)/4);
```

```
--> is(%<1)
```

Y lo mismo para el segundo polo:

```
--> cabs((sqrt(7)*%i-1)/4);
```

```
--> is(%<1);
```

mientras que para el tercero:

```
--> cabs(2);
    is(%<1);
```

Por tanto, los dos primeros sí que están en el interior del círculo unidad, mientras que el tercero (z=2) está fuera.

Podemos desarrollar una función en Serie de potencias en torno a un punto z0 (es decir, desarrollar f como potencias de z-z0):

EJEMPLO 12: Si queremos desarrollar como potencias de z (es decir, tomando z0=0) la función
g(z)=(z+1)/(z^3+4*z^2-3*z-18)
haremos

```
--> g(z):=(z+1)/(z^3+4*z^2-3*z-18);
```

```
--> powerseries(g(z),z,0);
```

```
--> niceindices(%);
```

Podemos desarrollar una función en Serie de Laurent en torno a un punto z0 (es decir, desarrollar f como potencias de z-z0, pudiendo ser los exponentes de z-z0 positivos o negativos):

EJEMPLO 13: Si queremos desarrollar como serie de Laurent en potencias de z+3 (ya que -3 es un polo de dicha función) la anterior función g(z) haremos

```
--> niceindicespref;
    niceindicespref:[n,m]$
    niceindices(powerseries(f(z),z,-3));
```

Una vez que conocemos los polos de una función, podemos calcular de manera sencilla sus residuos en dichos polos:

EJEMPLO 14: Si queremos calcular los residuos de la anterior función g(z) en sus puntos singulares (que son 2 y -3):

```
--> residue(g(z),z,2);
```

```
--> residue(g(z),z,-3);
```

6 Ejercicios a entregar.

PROBLEMA 1. Realizar las siguientes operaciones elementales:
1.a) Si z1=2+i, z2=3-2*i y z3=-1/2+i*(3^(1/2))/2, calcular abs(3*z1-4*z2), (z3)^(-4) y (z2)^(1/4).
1.b) Expresar en forma polar el complejo z=2+i*2*(3^(1/2)).
1.c) Calcular todas las raíces del polinomio p(z)=z^3+z^2+z+1=0

PROBLEMA 2. Resolver, con ayuda de wxMaxima, la Cuestión 2.b del examen de Matemáticas II de la convocatoria de febrero 2012, justificando lo que se realice.

PROBLEMA 3. Calcular, justificando lo realizado, el valor de la integral de la función f(z)=z+conjugate(z) entre los puntos z1=0 y z2=1+i a través de la curva y=x^2.

PROBLEMA 4. Dada la función f(z)=(z^2+3z)/(z^2-4):
4.1 Calcular los ceros y polos de f(z).
4.2 Determinar si las singularidades están o no localizadas en el interior del círculo unidad.
4.3 Obtener el desarrollo en serie de potencias de z.
4.4 Obtener el desarrollo en serie de Laurent como potencias de z+2.
4.5 Obtener el valor de los residuos en sus singularidades.

INSTRUCCIONES PARA LA ENTREGA DE LAS PRÁCTICAS:
- Las prácticas hay que hacerlas en fichero Maxima (extensión .wxm) y remitirlas por email.
- Aconsejable realizar un fichero para cada práctica (llamandolo, por ejemplo, Practica 1-nombre alumno.wxm)
- En los ficheros ir explicando lo que se va a realizar, incluyendo los comentarios que sean necesarios, como en el ejemplo que va a continuación.
- Las erratas que se cometan, se pueden eliminar, por lo que no es preciso que las entradas y salidas del programa sean consecutivas.