REC/C

Julio N. A. Quiróz* julioargota@hotmail.com

Pablo G. P. Beltrán * tyomero@gmail.com

Agosto 25, 2006

Resumen

Este documento tiene como objetivo, facilitar el uso de REC/C como una herramienta de análisis para aritmética compleja, conteniendo las modificaciones realizadas al programa durante el 2004, como lo son, el poder empilar directamente constantes complejas a la pila y algunas otras modificaciones que han afectado positivamente al programa.

Contenido

*ESCOM[sic], IPN

1 Introducción

El lenguaje REC/C fue creado por el Dr. Harold V. McIntosh como una de las especializaciones de REC, Regular Expression Compiler(Compilador de Expresiones Regulares) para la representación gráfica de operaciones de aritmética compleja. Está desarrollado en lenguaje de programación C-Objetivo, para el sistema operativo NeXTSTEP el cual ya no es común de ver.

Para poder utilizar y comprender los resultados de este programa es indispensable tener conocimiento y experiencia en el uso de aritmética compleja, así como entender la sintaxis de REC ya que esta basado en él.

1.1 Números complejos

Los números complejos tienen gran importancia en la matemática, y por ende en todas las áreas donde se ocupa como la física y la ingeniería por mencionar algunas, ya que hay momentos en los que sólo los números complejos pueden aplicarse, tal fue el caso cuando desearon encontrar las raíces de un polinomio de este tipo $x^2 + 1 = 0$ el cual sólo tiene raíces complejas por la propiedad de la unidad imaginaria $i^2 = -1$. Así nace el calculo de números complejos llamado análisis complejo, en el cual existen varias nuevas facilidades para las matemáticas, tal es el caso de integrales, derivadas por mencionar algunas.

También es posible resolver problemas de variable real de una forma más rapida usando los números complejos, e ahí uno de los motivos de su gran rango de aplicación.

1.2 Representación geométrica de funciones de una variable compleja.

A lo largo de la historia a sido un problema la representación de funciones de variable compleja, ya que, la representación de un número complejo se representa como un punto en un plano (a este plano se le llama plano de Argand), lo cual implica que para representar una función (de una sola variable) es necesario el producto cartesiano de dos planos, es decir, una gráfica en 4 dimensiones. Recordando que las representaciones geométricas de las funciones en general son útiles para darnos una idea más clara del comportamiento de dichas funciones en cierto intervalo o en general, se han creado diversas técnicas y métodos de representación de estas funciones dependiendo de la información que deseamos saber, sacrificando a veces información para darle una representación más estética o más clara de interpretar.

Una forma de representar el plano complejo es la proyección estereográfica que usa la esfera de Riemann la cual consiste en una esfera con un plano tangente a ella en su polo sur, y los puntos en ella son proyectados en el plano uniendo el punto en la esfera con el polo norte de la esfera y alargando la línea hasta que toque el plano, el punto donde lo toque será el punto que le corresponde en el plano, el polo norte representa el infinito y es el único punto que no se proyecta en el plano, es conocido como punto de fuga.

Una de las formas para observar el comportamiento de un conjunto de puntos del plano z al plano $f(z)$, es decir, de una transformación (también llamado mapeo) consiste en representar la función compleja en dos planos, donde marcaremos en el plano z el conjunto de puntos deseados y aplicamos la función (regla de transformación) a cada punto los que nos dará un conjunto de puntos en el plano $f(z)$. Otra forma de representar una función compleja es gráficar el valor absoluto de la función con respecto a su valor real e imaginario para poder observar su comportamiento a través de una superficie. Una de las formas de observar las raíces de un polinomio complejo es marcar una línea donde los extremos serán un punto de la función y su raíz, esta forma no es muy estética pero si representativa.

Las transformaciones o mapeos conformes son muy útiles para ver el comportamiento de una función ya que respetan la topología de la misma, la transformación de Möbius (tambien llamada de fracciones líneales) tiene la siguiente forma:

$$f(z) = \frac{az + b}{cz + d}$$

donde a, b, c, y d son constantes complejas. Supongamos que tenemos una transformación $w = f(z)$ y le aplicamos la tranformación de fracciones líneales con lo que nos quedra así:

$$g(w) = \frac{aw + b}{cw + d} = \frac{af(z) + b}{cf(z) + d}$$

esto provocará que la transformación de la función tenga al menos un cero y al menos un polo dependiendo de $f(z)$. Si representamos $f(w)$ de alguna manera de las señaladass arriba tendremos un poco de información extra sobre la función.

Estas formas de representar son sugerencias ya que cada uno puede idear alguna manera según las necesidades.

2 La interfaz de REC/C

2.1 Ventanas

Ventana de Programa. El programa consta principalmente de dos ventanas, la ventana del programa se muestra en la figura 1, que contiene la definición del programa, es decir, el código, al fondo hay tres botones etiquetados "recompile"(re-compilar) y "run"(ejecutar) e "use internal source", este último permite utilizar código ya programado, y buscar referencias de las funciones de REC/C.

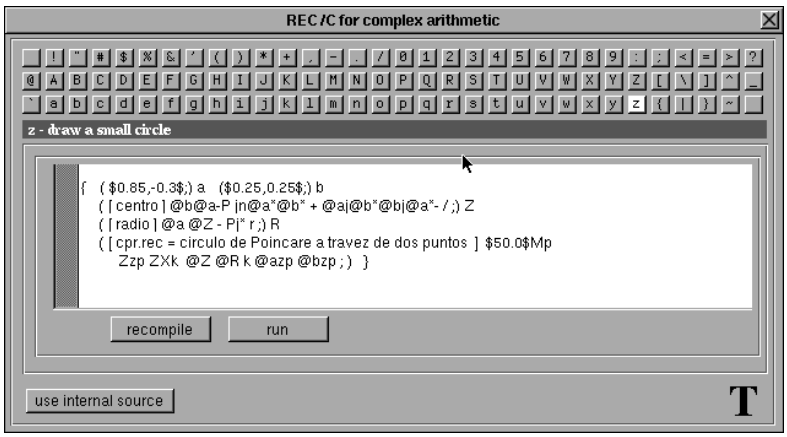

Figura 1: Ventana de programa.

Ventana del plano La ventana del plano(fig 2 donde se verán los resultados de las operaciones correspondientes al código después de compilarlo y ejecutarlo. Se puede manipular el tamaño de esta ventana para obtener la imagen deseada, deslizando las barras horizontal y vertical que se encuentran a la derecha y al fondo de la ventana respectivamente.

2.2 Menú de aplicación

La figura 3 muestra el menú de REC/C, con los submenús "Document"(Documento) y "Edit"(Editar), el submenú "Info" es el lugar donde se coloca el nombre del autor y colaberadores, afiliación, notas de derecho de autor y alguna información acerca del programa y como usarlo.

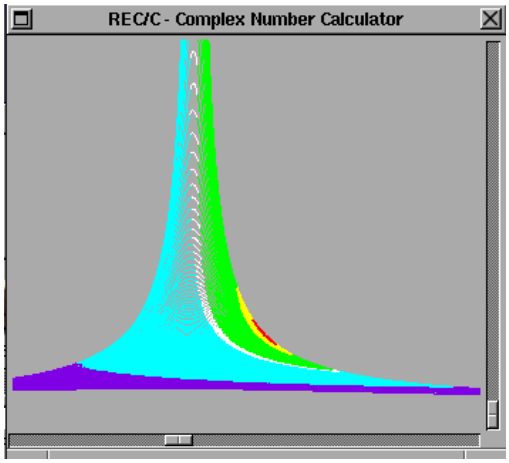

Figura 2: Ventana del plano.

3 Lenguaje REC/C

Para poder programar en REC/C es necesario dominar el concepto de pila y de la notación polaca inversa (postfija) ya que esta es la manera en la que REC/C evalúa las expresiones.

En REC/C hay operadores y predicados pero la distinción entre ellos sólo se hace por convención ya que un operador es considerado un predicado que siempre es verdadero, así pues los valores de los predicados definen el flujo de instrucciones del programa, indicando cuando son verdaderos que se debe seguir ejecutando de izquierda a derecha y cuando son falsos indica que se debe ir a los próximos dos puntos, o paréntesis derecho en ausencia de los dos puntos, dentro del mismo nivel de paréntesis.

Los elementos principales de la estructura del lenguaje son:

- Los paréntesis redondos '`()`', para agrupar las expresiones.
- El punto y coma '`;`', para la terminación de una expresioń, indicando que el el flujo del programa continuará en el primer paréntesis derecho que encuentre, si llega este paréntesis derecho sin haber encontrado otro signo de puntuación el '`;`' niega el valor del predicado.
- Los dos puntos '`:`', para indicar repetición de izquierda a derecha desde el primer paréntesis izquierdo que encuentre hasta los dos puntos de nuevo.
- Los espacios en blanco y los tabuladores son ignorados ya que sólo son utilizados como separadores para hacer más fácil la lectura de los programas.

Por lo mencionado anteriormente se puede observar que para crear estructuras de control sólo utilizaremos los paréntesis y los punto y coma. Siendo `p` y `q` predicados, podemos realizar las operaciones booleanas `(p q)` y `(p ó q)` con `(pq;)` y `(p;q;)` respectivamente. Para negar el predicado `p` bastará con la expresión `(p)`.

Todos los programas son predicados, incluso el principal, por lo que cada subrútina reporta su valor como predicado al siguiente nivel superior.

En la última versión de REC/C se agrega la funcionalidad de introducir constantes complejas en la cima de la pila escribiendo `$xx.xx,yy.yy$` donde `xx.xx` es la parte real y `yy.yy` es la imaginaria, las cuales pueden ser enteras o decimales. La conversión de la cadena de dígitos a números es realizada por la función estandar `sscanf()` de C, la cual es el inspector final de cadenas como `$$`, `$,$`, `$xx$`, `$xx.x,yy.y$`.

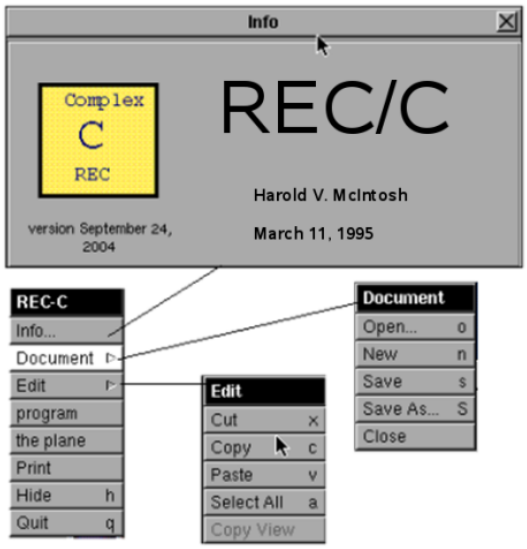

Figura 3: El menú de un programa, por lo regular contiene los elementos información(Info), Archivo(Document), Editar(Edit), Minimizar(Hide) y Salir(Quit), es apropiado incluir el menú Imprimir(Print).

3.1 Listado de operadores y predicados.

3.1.1 Operaciones aritméticas binarias:

Tomaremos como $[N]$ a el elemento de la cima de la pila y $[N\text{-}1]$ un elemento debajo de $[N]$. Cabe mencionar que el resultado de cada operación reemplaza a sus operandos en la pila, cada que se utilizan elementos de la pila para realizar una operación se realizan los siguientes pasos: *1)* Se toman el o los operandos necesarios para la operación; *2)* Se realiza la operación; *3)* Se desempilan los operandos; *4)* Se empila el resultado de la operación.

La siguiente es la lista de los operadores para las operaciones aritméticas:

- $+$ - realiza la suma $[N\text{-}1] + [N]$, deja sólo el resultado en la pila.
  textitEjemplo: `ab+` resulta en $a + b$.
- $-$ - realiza la resta $[N\text{-}1] - [N]$, deja sólo el resultado en la pila.
  Ejemplo: `ab-` resulta en $a - b$.
- $/$ - calcula el cociente $[N\text{-}1]/[N]$, deja sólo el resultado en la pila.
  Ejemplo: `ab/` resulta en a/b.
- $*$ - calcula el producto $[N\text{-}1] * [N]$, deja sólo el resultado en la pila.
  Ejemplo: `ab*` resulta en $a * b$.
- & - intercambia de posición los dos elementos de la cima de la pila.
  Ejemplo: El código `ab/`, resulta en a/b, mientras que `ab&/` resulta en b/a.
- Ejemplos:
  - El código `e a b cd*+-/` evaluará la expresión $\dfrac{e}{a-(b+(c*d))}$

– El código `e a b cd*+-&/` evaluará la expresión $\frac{a-(b+(c*d))}{e}$

– El código `az* b+ cz* d+ /` evaluará la expresión $\frac{az+b}{cz+d}$

3.1.2 Operaciones aritméticas unarias en la cima de la pila.

- j - calcula el conjugado complejo del elemento $[N]$.
- n - multiplica por -1 el elemento $[N]$.

3.1.3 Biblioteca de funciones comúnes.

- C - calcula el coseno hiperbólico del elemento en la cima, deja sólo el resultado.
- E - calcula la exponencial compleja del elemento en la cima, deja sólo el resultado.
- F - aplica la transformación de Möbius, también conocida como mapeo de Möbius $F(z) = (z+1)/(z-1)$ al elemento que está en la cima de la pila, dejando sólo el resultado.
- L - calcula el logaritmo complejo del elemento que está en la cima de la pila, deja sólo el resultado de la operación en la pila.
- r - calcula la raíz cuadrada del elemento en la cima, dejando sólo el resultado de la operación.
- T - calcula la tangente hiperbólica del elemento en la cima, dejando sólo el resultado en la cima.
- D - multiplica por 2 el elemento de la cima, deja sólo el resultado.
- d - divide por 2 el número que está en la cima, deja sólo el resultado.
- J - genera un número complejo aleatorio y lo empila.
- W - empila $e^{i\pi/6}$, que al ser multiplicado por el elemento en la cima, producirá una rotación de 30 grados en sentido contrario a las manecillas del reloj.
- w - empila $e^{i\pi/60}$, tiene la misma función que `W` sólo que la rotación será de 3 grados.

3.1.4 Empilar y desempilar elementos de en la pila.

- X - empilará `1`
- Y - empilará `i`
- u - empilará `0.1`
- v - empilará `0.1i`
- x - empilará `0.01`
- y - empilará `0.01i`
- Z - empilará `0`
- P - copiará el valor de la cima de la pila y lo colocará en la cima de la misma.
- p - sacará el elemento de la cima de la pila.
- Sn - salvará en la n-ésima constante el valor declarado anteriormente, como en " `$xx.x,yy.y$ Sn`". Deja la pila intacta.
- Rn - recuperará la n-ésima constante y la empilará.

3.1.5 Exhibición gráfica de la cima de la pila.

- A - indica que se graficará con algoritmo de ocultamiento de lineas.
- B - calcula el módulo del número complejo en la cima de la pila y después hace una transformación de R^3 a R^2 dejando el resultado en la cima pila.
- b - calcula el logaritmo del módulo del número complejo en la cima de la pila y después hace una transformación de R^3 a R^2 dejando el resultado en la cima pila.
- a - indica que se trazará una nueva línea con algoritmo de ocultamiento.
- H - indica el punto de inicio de una línea con algoritmo de ocultamiento indicado por `b` o `B`.
- h - continua el trazado de una línea con algoritmo de ocultamiento en el punto indicado por `b` o `B`.
- o - define el factor de corte, el valor por omisón es `$0.125,0.0$`
- M - define el tamaño de la gráfica, `$xx.x,0.0$Mp`, si no se define antes de iniciar el programa principal el valor predeterminado es 10.
- Qk - define el color `k` que sera usado por las funciones para dibujar, debe ser declarado antes de que se calcule la coordenadas de la figura a dibujar, los argumentos de esta función pueden ser los siguientes:
  - R, G, B - los colores aditivos primarios (red, green, blue) rojo, verde y amarillo.
  - C, M, Y, K - lo colores sustractivos primarios (cyan, magenta, yellow, black) ciánico, magenta, amarillo y negro.
  - K, D, L, W - éstos son tonos de gris negro, gris oscuro, gris claro y blanco respectivamente.
  - a - asigna color de acuerdo al valor de la fase del argumento del valor que hay en la cima de la pila.
  - v - asigna color de acuerdo al valor absoluto del argumento.
  - l - asigna color de acuerdo al logaritmo del valor absoluto del argumento .
  - s - color estratificado (en capas) de acuerdo al valor absoluto del argumento. *Ejemplo:* "`P...Qk B gp`"
- k - (z0 r k;), dibuja un círculo con centro en `z0` de radio `r` y deja en la pila `z0`
- s - dibuja un cuadradado para formar un par estereoscópico.
- q - dibuja un cuadrado pequeño del color definido por `Qk`, donde `k` es el color, y el tamaño debe ser definido por la línea `$xx.x,yy.y$ m p`, en caso de no definir ningún color para dibujar o el tamaño de cuadro se tomarán los valores predeterminados `QD` y `$1.25,0.0$mp`.
- z - Dibuja un pequeño círculo con centro en el valor del elemento en la cima de la pila.
- G - posiciona el cursor para una línea regular,el valor que hay en la cima de la pila indica el punto donde iniciará la línea.
- g - une con una línea recta de color `Qk` el punto definido por `G` o por si misma en un trazo anterior, con el punto que indicado por el valor del elemento que esta en la cima de pila.

3.1.6 Predicados.

- I - pregunta si la parte real del número que esta en la cima de pila es un entero.
- i - pregunta si la parte real del número en la cima de la pila es un múltiplo de `0.1`.
- !n! - la declaración es la siguiente, `(!n!...:;)`, la cual repite n veces el código a partir del paréntesis izquierdo hasta los dos puntos.
- # - evalúa un predicado generado aleatoriamente, donde la probabilidad de que el resultado sea "Verdadero"$(P(V))$ es igual a $1/2$, y la probailidad de que sea falso es $P(F) = 1 - P(V)$.
- % - evalúa un predicado generado aleatoriamente, donde la probabilidad de que el resultado sea "Verdadero"$(P(V))$ es igual a $1/10$, y la probailidad de que sea falso es $P(F) = 1 - P(V)$.

3.1.7 Otros operadores.

- `?` - imprime en la pantalla el valor del elemento que se encuentra en la cima de la pila en el momento de ser llamada.
  La definición de `?` es `(w z ?;)`, la cual escribe el valor númerico ASCII del número `w` justo sobre la posición `z`, dejando `z` en la pila y desempilando `w`. Escribir algo como `(w P ?;)` dará como resultado un punto etiquetado con el valor de su posición.
  Escribir `(XDD P ?;)` mostrará la etiqueta del punto $x = 4.0$, justo encima de este punto.
- `lx` - puede servir como un contador del nivel de profundidad de la recursión, donde `x` es un número entero. Cuando se utilizan los operadores `+,-` en conjunto con `lx`, como en `l+` o `l-`, la operación que realizan `+` y `-` cambia. *Ejemplo:* Al colocar `l9`, se establece que el nivel de profundidad de la recursión es 9, las operaciones `l+` y `l-`, aumentan y decrementan respectivamente el nivel de profundidad de recursión, es decir, si `l` guarda el valor 9, después de escribir `l-`, el valor de `l` será 8, sin embargo el nivel de profundidad no aumentará a 10 si se le aplica `l+`, ya que 9 es el valor máximo que se puede utilizar en ésta versión de REC/C. Sin embargo `(l8 l+;)` aumentará el valor del nivel de profuncidad de 8 a 9.

Comentarios. Para agregar comentarios a un programa escrito en REC sólo se nesecita colocar el comentario entre paréntesis cuadrados, esta caractéristica es heredada por REC/C.

Subrutinas. Las subrútinas son muy útiles para definir constantes o funciones, estas deben ser definidas antes de poder ser utilizadas. La forma de construirlas y llamarlas es la siguiente:

- Las subrútinas son útiles para, separar el programa en funciones o módulos, de manera que sea más comprensible para el programador. Una subrútina se define `(...código...;)x`, donde `x` es el identificador de la subrútina, cualquier letra es un identificador válido como identificador de un subrútina. El contenido de la subrútina debe terminar con punto y coma.

3.2 Estructura de un programa en REC/C

Con todo lo mencionado anteriormente podemos dar la estructura de un programa en REC/C que es la siguiente:

```
{
[los comentarios son opcionales, aunque nunca sobran]
(subrutina 1;)x [las subrutinas son opcionales y deben
estar identificadas]
(subrutina 2;)y
([programa principal]$50.0,0.0$Mp ...
...@x[esta sentencia llama a la subrutina x]
[Tanto las subrutinas como el programa principal debe
terminar con punto y coma];)}
```

3.3 Pila de datos

La figura 4 nos da un esquema gráfico de como se utiliza la pila, indicando con flechas hacia adentro cuando insertan datos y con flechas hacia afuera cuando toman el valor de la pila o sólo lo toman para hacer referencia sin sacarlo. Por ejemplo la flecha que tiene los operadores `G, g,` etc no los saca nada de la pila pero para su función toma el valor y lo gráfica.

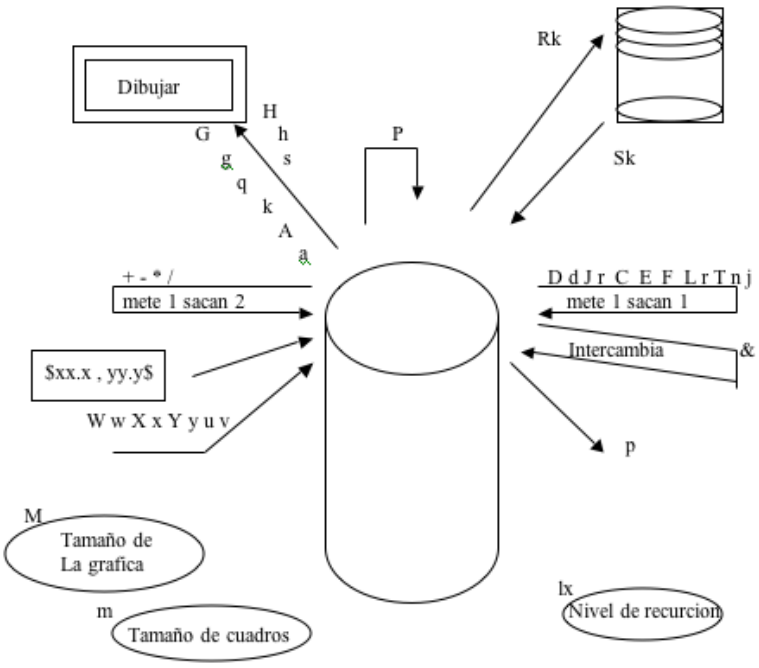

Figura 4: Representación de la pila de REC/C.

4 Tipos de representación gráfica.

Como ya se mencionó todos los resultados de los cálculos aritméticos hechos con REC/C se muestran en forma de una gráfica, así que se explica las diferentes formas de mostrarlos.

4.1 Representación por líneas

Es recomendable comenzar con ejercicios algo sencillo para poderse familiarizar tanto con la forma de trabajo en REC/C como en la interpretación de la solución gráfica que nos arroja. Una forma de comenzar es trazar figuras en las que estemos seguros de lo que queremos. Trazar figuras a partir con líneas puede ser un buen inicio.

Para dibujar líneas en REC/C marcaremos el inicio de una línea con el operador `G` que tomará el valor de encima de la pila y posteriormente el operador `g` continuará la línea ya sea desde el punto marcado por `G` o por el último que él mismo marcó. De esta forma si se requiere detalle en la línea para ver si es curva o recta sólo asígnaremos incrementos pequeños.

Un ejemplo para familiarizarnos es dibujar la figura 5. Hay varias maneras de hacerlo, pero mostraremos dos que consideramos las más representativas.

- Forma 1

```
{(
[dibuja lineas verticales]
Z(!10! P G $1.0,0.0$+ g p Y+:;)p
[dibuja lineas horizontales]
Z(!10! P G $0.0,1.0$+ g p X+:;)p
$;)}
```

- Forma 2

```
{(
[dibuja lineas verticales]
Z(!10! PG(!10! Y+g:;)pX+:;)p
[dibuja lineas horizontales]
Z(!10! PG(!10! X+g:;)pY+:;)p
;)}
```

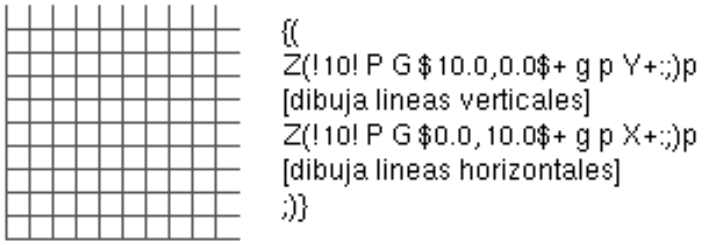

Figura 5: Rejilla generada con REC/C

Se puede observar que estos dos programas arrojan el mismo resultado pero hay una diferencia significativa en ellos, uno tiene iteraciones internas y otro no, esto se debe a que uno hace un incremento de

10 unidades reales o imaginarias, según se necesite, y el otro hace 10 incrementos de una unidad cada uno. En la forma 2 podemos ver que los contadores exteriores nos dicen el número de líneas a dibujar y en los contadores internos la longitud de la línea.

La representación mediante líneas tiene gran utilidad para ver como se comporta una función al momento de aplicarle una transformación, por ejemplo, como se comportará una línea recta, trazada en el plano z, en el plano $w = f(z)$.

En este tipo de representación debemos tener en cuenta la escala a la que se esta graficando y dependiendo lo que queremos graficar los incrementos que consideraremos. A continuación mostraremos ejemplos de como lo anterior con la transformación de una rejilla en el plano $f(z) = cosh(z)$(fig 6), en la que observamos lo importante que es la correcta asígnación del incremento.

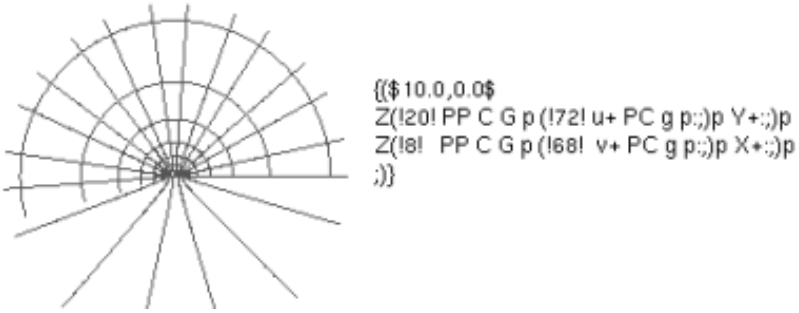

Figura 6: Coseno hiperbólico aplicado a la rejilla anterior.

En la figura 7 sólo se ha aumentado la escala y cambiado el número de iteraciones para el trazo de las líneas.

```
{($20.0,0.0$Mp
Z(!9! PP C G p (!140! x+ PC g p:;)p Y+:;)p
Z(!4! PP C G p (!250!  y+ PC g p:;)p X+:;)p
;)}
```

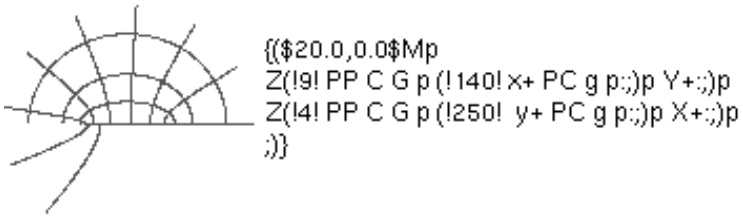

Figura 7: Coseno hiperbólico de mayor tamaño.

Ahora que ya estamos un poco familiarizados podemos hacer una transformación un poco más significativa, aplicaremos una transformación cuadrática, $f(z) = z^2$ a una rejilla, lo que nos dará dos familias ortogonales de parábolas, ya que las líneas de la rejilla son ortogonales.

```
{(
Z(!30! P PP* Gp (!30! u+ PP*  g p :;)p v+:;)p
Z(!30! P PP* Gp (!30! v+ PP*  g p :;)p  u+:;)p
;)}
```

Como podemos apreciar empilamos los cálculos a la cima de la pila, los graficamos y luego los sacamos para poder seguir haciendo el incremento correctamente.

4.2 Representación por contornos

Un contorno es la curva que se genera cuando se cruza un plano horizontal (normalmente) en cierto nivel con una superficie en tres dimensiones, en el caso de REC/C generará contornos que se distinguirán por colores, los colores indicarán diferentes valores, siendo así el color una especie de dimensión. La forma de colorear será designada según lo que se desee y es indicada por el operador `Qk` siendo `k` el tipo de coloreado (`s,a,v,`etc).

El color asignado a un punto representará la magnitud del valor absoluto o de la fase del punto el cual es resultado de la transformación (función), es decir, un punto $z = x + iy$ resultado de haber aplicado la fución se dibujara en las cordenadas (x, y) y se le asignara un color según el valor de su módulo o de su fase.

La representación por contornos nos es muy útil en la exploración de raíces, polos y puntos fijos de una función, ya que nos puede indicar un aproximado de su localización y el número de ellos. La distinción de ellos será por los colores.

El siguiente código genera la figura 8, la cual es un ejemplo básico de una gráfica de contornos.

```
{([evalua la funcion] P(I QK; Qv;) qp;) f
($20.0,0.0$Mp
Z(!60! P @f (!60! u + @f:;) pv+ :;) p
Z(!60! P @f (!60! u + @f:;) pv- :;) p
Z(!60! P @f (!60! u - @f:;) pv- :;) p
Z(!60! P @f (!60! u - @f:;) pv+ :;) p
;)}
```

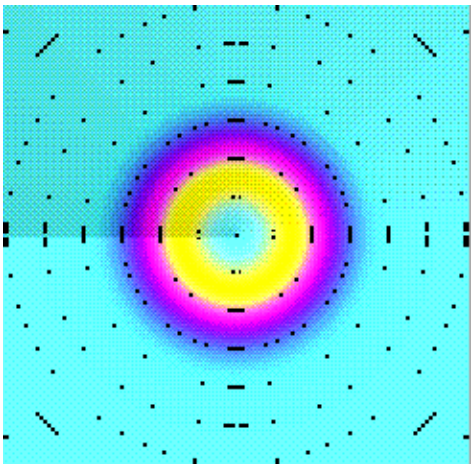

Figura 8: contornos de $f(z) = z$

4.3 Par estereoscópico.

Este tipo de representación genera dos imágenes de una misma figura, con una ligera diferencia de ángulo entre las imágenes,similar a la existente entre los ojos del espectador. Para observar en relieve la presente imágen se debe cruzar la visión (bizquear)hasta que aparezca una imágen central mezcla de las dos existentes que permaneceran a los lados. Puede ayudar situar un dedo tocando la pantalla entre las dos imágenes y poco a poco moverlo en dirección a la nariz. Si se mantienen los ojos enfocados en el dedo las imágenes borrosas de detrás se irán uniendo en una sola central, que, una vez retirado el dedo se enfocará cuidando que los ojos se mantengan cruzados. Con un poco de ejercicio resulta fácil de ver. El siguiente código genera la figura 9

```
{([define la funcion] PP* s p;)f
($10.0,0.0$Mp
  Z( !100! P @f  (!100! u+@f  :;) p v+:;)
  Z( !100! P @f  (!100! u+@f  :;) p v -:;)
  Z( !100! P @f  (!100! u-@f  :;) p v +:;)
  Z( !100! P @f  (!100! u-@f  :;) p v -:;)
;)}
```

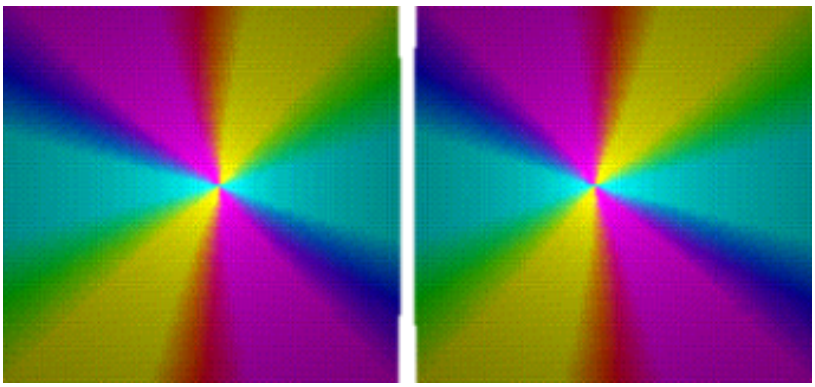

Figura 9: Par estereoscópico de $f(z) = z^2$.

También puede obtenerse el efecto en pantalla grande, utilizando gafas polarizadas de cartón o plástico de las que se entregan en la proyección de películas en 3d comerciales. Hay que disponer de dos proyectores de diapositivas iguales, a los que se adaptarán sendos filtros polarizadores que se pueden adquirir en tiendas de fotografía. Estos filtros están montados de forma que pueden girarse.

Un proyector llevará las diapositivas del ojo izquierdo y el otro las del ojo derhecho. Para ajustar el sistema se proyectará sólo la del ojo izquierdo y , con las gafas puestas se girará el filtro polarizado hasta que la imagen se vea lo mejor posible con el citado ojo (con el otro no debería verse). Se apaga el proyector izquierdo y se repite el ajuste para el ojo derhecho. Luego se encienden ambos proyectores de forma que coincidan las imágenes una encima de la otra y con las gafas puestas disfrutaremos de la proyección.

4.4 Representación de superficies

Se puede representar una función compleja mediante una superficie, tomado el eje real como el eje x, el eje imaginario como el y y el módulo del número complejo como el la altura para el eje z. De esta manera podemos darnos un mejor idea del comportamiento de la función, ya que los puntos críticos de esta gráfica también lo son de la función.

Para ver una superficie mejor definida se usan algoritmos de ocultamiento los cuales deciden que partes se ven y cuales no, si no se implementaran se vería una especie de cuerpo transparente pero crearía confución. REC/C implementa un algoritmo de este tipo y al hacer sus gráficas de superficies con sólo líneas es un algoritmo de ocultamiento de líneas.

Dependiendo de la manera en que coloremos la superficie podemos darnos cuenta también de puntos fijos, ceros, raíces o polos.

Hay dos formas principales de representar la función a través de superficies una con el valor absoluto tal cual (operador `B`) o con el logaritmo del valor absoluto (operador `b`). La diferencia entre éstas es que con valor absoluto podemos ver el comportamiento de la función en valores muy grandes, por las características de la función logarítmica (los valores de su imagen se incrementan muy lento), tiene un efecto como de compactación de la gráfica.

Las superficie son mostradas de frente, sólo con la posibilidad de ser rotadas con respecto al eje imaginario (con el operador `w` o `W`), por lo cual para ver ciertos lugares de la gráfica hay que valerse de ciertos trucos como dibujar sólo la mitad del intervalo que se fijo inicialmente.

Para graficar estas superficies hay que tomar en cuenta que serán formada por líneas en un sólo sentido y que también la escala y el incremento son aspectos que hay que cuidar, normalmente se recomiendan incrementos pequeños.

El algoritmo de ocultamiento de líneas y la transformación de R^3 a R^2 lo hace REC/C por si sólo así que sólo debemos preocuparnos por darle tanto el valor de z (parte real e imaginaria) como de la función $f(z)$ de una manera correcta. Para dibujar la superficie primero con el operador `A` indicaremos que nos preparamos para dibujar líneas que dependiendo de la gráfica se vera o no, posteriormente debemos calcular donde será el inicio de la línea con el operador B o b y después marcarlo como tal con el operador `H`, después empezaremos a calcular los valores donde continua la linea con el operador `b` o `B` y a trazarla con el operador `h`. La manera en la que operan los operadores `G` y `g` es muy parecida a `BH` y `Bh` o `bH` y `bh` con la notable diferencia que el uso de los operadores `g` y `G` no implica una tranformación (de 3 a 2 dimenciones) ni un algoritmo de ocultamiento de líneas.

A continuación mostraremos un ejemplo haciendo hincapié en que el siguiente código es válido para cualquier función que se le desee tan sólo tomando en cuenta que hay que variar el número de líneas y la longitud de las mismas según se desea así como los límites inferiores en el eje real e imaginario, ya que el número de líneas y la longitud de cada una determinara el límite superior para el eje real e imaginario dependiendo como se asignen los incrementos.

```
{ (PPPP** $2,2$+&$2.0,1.0$*+ ;)F
(PPPPP** $2,2$+&$2.0,1.0$*+- ;)G
 (
 $50.0$Mp
 $-3.0,-3.0$ A (!300!   P P@FF BHp
                      (!200! $0.02,0.0$+ P@GQsp @FFBhp  :;)
          p $0.0,0.02$ + a  :;)
; )}
```

En la figura 10 se muestra la función

$$f(w) = \frac{w+1}{w-1}$$

siendo $w = Z + (2+i)Z + (2+2i)$ la cual corresponde al código de arriba. En ella se pueden observar varias de sus carácteristicas, tales como son sus puntos fijos, polos y ceros, los cuales se pueden localizar por medio de los colores de las líneas.

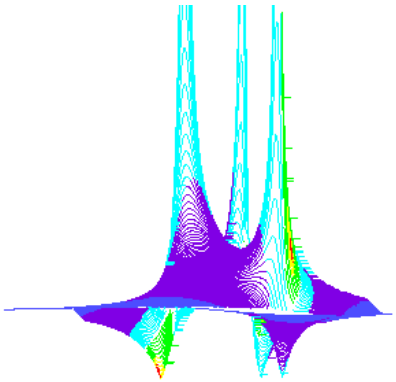

Figura 10: Mapeo o transformación de Möbius.

El color rojo representa los puntos fijos o en su caso los ceros de la función,el morado los polos, el blanco indica que el valor absoluto en ese punto es 1.

5 Agradecimientos

Este documento en gran medida está basado en el documento de la primera versión de REC/C [1], del cual se tomarón la mayoría de los ejemplos e imágenes, también el Dr. McIntosh aclaró dudas sobre las actualizaciones de REC/C y el funcionamiento de las nuevas funciones.

Referencias

[1] Harold V. McIntosh, *REC/C for Complex Arithmetic*, Departamento de Aplicación de Microcomputadoras, Instituto de Ciencias, Universidad Autónoma de Puebla, Apartado postal 461, 72000 Puebla, Puebla, México. 20 de Enero, 2001 (82 páginas)

[2] Harold V. McIntosh, *Complex Variables: The Möbius Transformations* `http://microcomputadoras.buap.mx/phpBB2/viewtopic.php?t=37`, 6 de Enero, 2004.

[3] Harold V. McIntosh, *Modifications to REC/C during the fall, 2004* `http://microcomputadoras.buap.mx/phpBB2/viewtopic.php?t=35`, 6 de Enero, 2004

[4] José Aguilera García, *Speleo Web pages at Cádiz University* `http://www2.uca.es/huesped/giex/estereo.htm`, 4 de Agosto, 2006

6 Referencia rápida de operadores, y predicados de REC/C

- + - realiza la suma [N-1] + [N], deja sólo el resultado en la pila.
  textitEjemplo: `ab+` resulta en $a + b$.
- − - realiza la resta [N-1] − [N], deja sólo el resultado en la pila.
  Ejemplo: `ab-` resulta en $a - b$.
- / - calcula el cociente [N-1]/[N], deja sólo el resultado en la pila.
  Ejemplo: `ab/` resulta en a/b.
- ∗ - calcula el producto [N-1] ∗ [N], deja sólo el resultado en la pila.
  Ejemplo: `ab*` resulta en $a * b$.
- & - intercambia de posición los dos elementos de la cima de la pila.
  Ejemplo: El código `ab/`, resulta en a/b, mientras que `ab&/` esulta en b/a.
- Ejemplos:
  - El código `e a b cd*+-/` evaluará la expresión $\frac{e}{a-(b+(c*d))}$
  - El código `e a b cd*+-&/` evaluará la expresión $\frac{a-(b+(c*d))}{e}$
  - El código `az* b+ cz* d+ /` evaluará la expresión $\frac{az+b}{cz+d}$
- # - evalúa un predicado generado aleatoriamente, donde la probabilidad de que el resultado sea "Verdadero"$(P(V))$ es igual a 1/2, y la probailidad de que sea falso es $P(F) = 1 - P(V)$.
- % - evalúa un predicado generado aleatoriamente, donde la probabilidad de que el resultado sea "Verdadero"$(P(V))$ es igual a 1/10, y la probailidad de que sea falso es $P(F) = 1 - P(V)$.
- !n! - la declaración es la siguiente, `(!n!...:;)`, la cual repite n veces el código a partir del paréntesis izquierdo hasta los dos puntos.
- `?` - imprime en la pantalla el valor del elemento que se encuentra en la cima de la pila en el momento de ser llamada.
  La definición de `?` es `(w z ?;)`, la cual escribe el valor númerico ASCII del número `w` justo sobre la posición `z`, dejando `z` en la pila y desempilando `w`. Escribir algo como `(w P ?;)` dará como resultado un punto etiquetado con el valor de su posición.
  Escribir `(XDD P ?;)` mostrará la etiqueta del punto $x = 4.0$, justo encima de este punto.
- a - indica que se trazará una nueva línea con algoritmo de ocultamiento.
- b - calcula el logaritmo del módulo del número complejo en la cima de la pila y después hace una transformación de R^3 a R^2 dejando el resultado en la cima pila.
- d - divide por 2 el número que está en la cima, deja sólo el resultado.
- g - une con una línea recta de color `Qk` el punto definido por `G` o por si misma en un trazo anterior, con el punto que indicado por el valor del elemento que esta en la cima de pila.
- h - continua el trazado de una línea con algoritmo de ocultamiento en el punto indicado por `b` o `B`.
- i - pregunta si la parte real del número en la cima de la pila es un múltiplo de `0.1`.
- j - calcula el conjugado complejo del elemento [N].

- k - (z0 r k;), dibuja un círculo con centro en `z0` de radio `r` y deja en la pila `z0`
- lx - puede servir como un contador del nivel de profundidad de la recursión, donde `x` es un número entero. Cuando se utilizan los operadores `+,-` en conjunto con `lx`, como en `l+` o `l-`, la operación que realizan `+` y `-` cambia. *Ejemplo:* Al colocar `l9`, se establece que el nivel de profundidad de la recursión es 9, las operaciones `l+` y `l-`, aumentan y decrementan respectivamente el nivel de profundidad de recursión, es decir, si `l` guarda el valor 9, después de escribir `l-`, el valor de `l` será 8, sin embargo el nivel de profundidad no aumentará a 10 si se le aplica `l+`, ya que 9 es el valor máximo que se puede utilizar en ésta versión de REC/C. Sin embargo `(l8 l+;)` aumentará el valor del nivel de profuncidad de 8 a 9.
- n - multiplica por -1 el elemento $[N]$.
- o - define el factor de corte, el valor por omisón es `$0.125,0.0$`
- p - sacará el elemento de la cima de la pila.
- q - dibuja un cuadrado pequeño del color definido por `Qk`, donde `k` es el color, y el tamaño debe ser definido por la línea `$xx.x,yy.y$ m p`, en caso de no definir ningún color para dibujar o el tamaño de cuadro se tomarán los valores predeterminados `QD` y `$1.25,0.0$mp`.
- r - calcula la raíz cuadrada del elemento en la cima, dejando sólo el resultado de la operación.
- s - dibuja un cuadradado para formar un par estereoscópico.
- u - empilará `0.1`
- v - empilará `0.1i`
- w - empila $e^{i\pi/60}$, tiene la misma función que `W` sólo que la
- x - empilará `0.001`
- y - empilará `0.001i`
- z - Dibuja un pequeño círculo con centro en el valor del elemento en la cima de la pila.
- A - indica que se graficará con algoritmo de ocultamiento de lineas.
- B - calcula el módulo del número complejo en la cima de la pila y después hace una transformación de R^3 a R^2 dejando el resultado en la cima pila.
- C - calcula el coseno hiperbólico del elemento en la cima, deja sólo el resultado.
- D - multiplica por 2 el elemento de la cima, deja sólo el resultado.
- E - calcula la exponencial compleja del elemento en la cima, deja sólo el resultado.
- F - aplica la transformación de Möbius, también conocida como mapeo de Möbius $F(z) = (z+1)/(z-1)$ al elemento que está en la cima de la pila, dejando sólo el resultado.
- G - posiciona el cursor para una línea regular,el valor que hay en la cima de la pila indica el punto donde iniciará la línea.
- H - indica el punto de inicio de una línea con algoritmo de ocultamiento indicado por `b` o `B`.
- I - pregunta si la parte real del número que esta en la cima de pila es un entero.
- J - genera un número complejo aleatorio y lo empila.

- L - calcula el logaritmo complejo del elemento que está en la cima de la pila,deja sólo el resultado de la operación en la pila.
- M - define el tamaño de la gráfica, `$xx.x,0.0$Mp`, si no se define antes de iniciar el programa principal el valor predeterminado es 10.
- P - copiará el valor de la cima de la pila y lo colocará en la cima de la misma.
- Qk - define el color `k` que sera usado por las funciones para dibujar, debe ser declarado antes de que se calcule la coordenadas de la figura a dibujar, los argumentos de esta función pueden ser los siguientes:
  - R, G, B - los colores aditivos primarios (red, green, blue) rojo, verde y amarillo.
  - C, M, Y, K - lo colores sustractivos primarios (cyan, magenta, yellow, black) ciánico, magenta, amarillo y negro.
  - K, D, L, W - éstos son tonos de gris negro, gris oscuro, gris claro y blanco respectivamente.
  - a - asigna color de acuerdo al valor de la fase del argumento del valor que hay en la cima de la pila.
  - v - asigna color de acuerdo al valor absoluto del argumento.
  - l - asigna color de acuerdo al logaritmo del valor absoluto del argumento .
  - s - color estratificado (en capas) de acuerdo al valor absoluto del argumento. *Ejemplo:* "`P...Qk B gp`"
- Rn - recuperará la n-ésima constante y la empilará.
- Sn - salvará en la n-ésima constante el valor declarado anteriormente, como en " `$xx.x,yy.y$ Sn`". Deja la pila intacta.
- T - calcula la tangente hiperbólica del elemento en la cima, dejando sólo el resultado en la cima.
- W - empila $e^{i\pi/6}$, que al ser multiplicado por el elemento en la cima, producirá una rotación de 30 grados en sentido contrario a las manecillas del reloj. rotación será de 3 grados.
- X - empilará `1`
- Y - empilará `i`
- Z - empilará `0`