DESCRIPCIÓN BREVE DE LAS SUBRUTINAS DE GRAFICACIÓN DE PLOT84

José Pedro Hernández Enríquez
Departamento de Aplicación de Microcomputadoras
Instituto de Ciencias. Universidad Autónoma de Puebla
Apartado Postal 461, 72000 Puebla, Pue., México

1998

Contenido

Capítulo 1

Introducción

'PLOT' es un conjunto de subrutinas para trazar que consiste principalmente en programas de contornos y subrutinas de líneas ocultas. Se desarrolló en el INEN (Instituto Nacional de Energía Nuclear) por el Dr. Harold V. McIntosh a mediados de la década de los años 70 en lenguaje FORTRAN, desde entonces se ha implementado en diferentes computadoras a nivel nacional e internacional. A nivel internacional el Dr. Nelson H. F. Beebe de los Departamentos de Física y Química de la Universidad de Utah ha tomado como base el trabajo de McIntosh y ha realizado la versión de PLOT conocida como PLOT79. Esta versión de PLOT se ajusta a la definición del sistema de gráficas CORE y contiene gran variedad de alfabetos y símbolos, y rutinas de interpolación cuadrática. Estas últimas características fueron implementadas por iniciativa de Beebe.

En la UAP se tiene una versión de PLOT en la computadora IBM-1130 en lenguaje FORTRAN que utiliza el trazador CalComp. La implentación de PLOT conocida como PLOT84 utiliza el Sistema Operativo CP/M (8080), el trazador HIPLOT DMP-29 o el trazador CalComp 1044 (la biblioteca se llama PLOTLIB) y el lenguaje FORTRAN que fue realizada por el Lic. Francisco Gil Zepeda. Se tiene, también, una implementación de PLOT en lenguaje "C" para las microcomputadoras de tipo PC realizada por el Dr. Gerardo Cisneros Stoianowski y Angel Bernardo Canto Gómez.

Las fuentes para el presente trabajo son el libro PLOT75 de Harold V. McIntosh y el manual de guía de usuario de PLOT79 de Nelson H. F. Beebe.

Todas las subrutinas de PLOT están cargadas en la biblioteca de FORTRAN (FORLIB), el usuario sólo tiene que llamarlas y, al momento del encadenamiento ellas pasan a formar parte del programa. En el caso del lenguaje "C" la biblioteca se llama LIBL1, en "C" las subrutinas de PLOT se implementaron como funciones, ya que en "C" no existe el concepto de subrutina.

La presentación del paquete PLOT se divide en la forma siguiente:

1.1 Herramientas de trabajo

Breve descripción del equipo y el Sistema Operativo. Descripción del graficador HIPLOT DMP-29 y el CalComp 1044.

Debido a la gran cantidad de subrutinas que contiene PLOT se han impuesto una serie de convenciones en el nombre que se les asigna. El nombre de una subrutina sigue el patrón XXXYY en el cual XXX es un mnemónico que designa una categoría de subrutinas, y YY identifica a una subrutina individual dentro de esa categoría. Los valores más comunes de XXX serán:

KON	Programas de contorno para funciones almacenadas en un arreglo.
PLT	Subrutinas generales de graficación.
VIS	Subrutinas para gráficas con supresión de líneas ocultas.
YY	Se forma de las dos letras más significativas de la palabra o frase breve que describe a la subrutina.

1.2 Subrutinas básicas

Son aquellas que se comunican con los periféricos o las que efectúan acciones a nivel más bajo:

PLOTI	Prepara el ambiente de trabajo.
PLTDK	Define que la salida de las coordenadas y comandos será a un archivo.
PLOT	Traza una línea, llama a PLOTM.
PLOTM	Transmite las coordenadas al graficador llama a PLOMT.
PLOMT	Transforma las coordenadas en binario a ASCII llama a PLOTZ.
PLOTZ	Selecciona cuál es el destino de las coordenadas, el graficador (llama a PLOZT) o un archivo.
PLOZT	Transmite al graficador un bloque de N carácteres ASCII que sean comandos del graficador.
PLOTF	Define la orientación de los ejes X e Y.
PLOTO	El punto previo se toma como el origen.
PLOTS	Define las escalas en X y en Y.
PLOTG	Define la escala general.
PLOTP	Guarda el origen y las escalas.
PLOTR	Recupera el origen y las escalas.
TIME	Obtiene la hora del reloj de tiempo real.
DATE	Obtiene la fecha definida por el sistema.
ESCVID	Escribe en el video un bloque de N carácteres ASCII.

1.3 Subrutinas de uso general

Algunas de las subrutinas de uso general siempre aparecen en los programas de los usuarios.

PLT00	Preparación del graficador y del paquete PLOT.
PLTFR	Traza un marco.
PLTSO	Define las escalas y el origen.
PLTHO	Desaloja la pluma, la coloca en el origen.
PLTEJ	Desaloja la pluma, la coloca en la siguiente posición.
PLTCA	Coordenadas cartesianas.
PLTPO	Coordenadas polares.
PLTEL	Coordenadas elípticas.
PLTSP	Coordenadas esféricas.

1.4 Subrutinas para gráficas planas

Estas subrutinas generalmente grafican datos almacenados en arreglos:

PLTIG	Trazo en incrementos no utiliza arreglos, se emplea cuando no se conoce el número de puntos a graficar.
PLTFI	Grafica un arreglo cuyos valores en X son enteros.
PLTFM	Traza marcas.

PLTGA	Grafica un arreglo.
PLTGD	Grafica datos (coloca marcas).

1.5 Trazo de contornos

Estas subrutinas grafican los datos contenidos en un arreglo matricial, los cuales pueden ser obtenidos experimentalmente o por medio de la evaluación de una función:

PLTKP	Grafica contornos.
PLTOR	Grafica relieves ortográficos.
PLTKC	Grafica contornos de funciones complejas.
PLTKX	Traza rayas en la dirección X.
PLTKY	Traza rayas en la dirección Y.
KONSC	Traza un contorno simple.
KONOC	Traza un relieve ortográfico.

1.6 Trazo de superficies con supresión de líneas ocultas

Las subrutinas que realizan el trazo toman sus datos desde un arreglo matricial que es obtenido experimentalmente o por la evaluación de una función. Tanto en el caso de los contornos como en el caso de superficies los arreglos no son destruidos.

PLTEU	Genera la matriz de rotaciónde Euler.
PLTSW	Traza una superficie con vista lateral suroeste.
PLTSE	Traza una superficie con vista lateral sureste.
PLTRV	Traza una superficie con rotación entre -90 y 90 grados.
PLTIV	Traza una superficie con inclinación y rotación.
PLTPV	Traza una superficie en coordenadas polares.
PLTTV	Traza una superficie en coordenadas triangulares.
PLTSV	Traza una superficie en coordenadas esféricas.
PLTEV	Traza una superficie en coordenadas elípticas.
VISDS	El análisis de la matriz de datos se realiza en forma diagonal, esta subrutina proporciona las cuatro posibles secuencias en coordenadas cartesianas.

1.7 Trazo de superficies múltiples

Las subrutinas de superficies múltiples permiten dibujar diferentes superficies mostrando la forma en que se encuentran relacionadas, o permiten que se desplacen verticalmente para ver los detalles de sus intersecciones. Las subrutinas que se tienen son para superficies dobles o triples, en estos casos se requiere dos arreglos matriciales o tres de ellos según sea el caso.

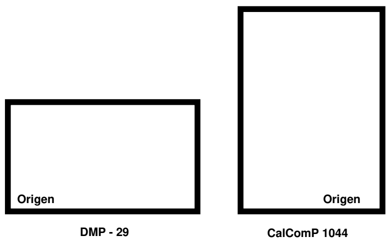
VISDO	Grafica dos superficies.
VISTR	Grafica tres superficies.

Capítulo 2

Características Generales de PLOT

- El graficador digital incremental Hiplot DMP-29 de Houston Instrument, el graficador CalComp 1044 de California Computer Products Corporation y la terminal de video de una computadora de tipo PC, serán los aparatos en los que se obtengan las gráficas.

- El tamaño de la página estándar de trazo es una hoja tamaño carta, 8.5" × 11". La hoja para el trazo se coloca horizontalmente en el graficador Hiplot DMP-29 y verticalmente en el CalComp 1044.

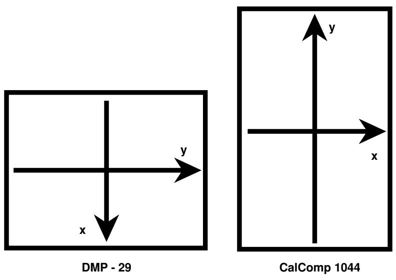


- Se considera que el origen de las coordenadas es la esquina inferior derecha de la hoja de trazo.

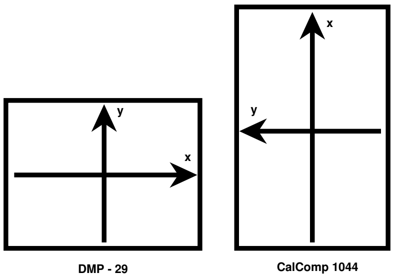
- El área de trazo se puede enmarcar con seis tipos de marcos. Si se enmarca el área de trazo, entonces el origen de las coordenadas será el centro del marco. Otra opción es definir el origen de las coordenadas en el centro de un marco ficticio de 8.5" × 11".

- Se mantiene un margen en los límites de la superficie de trazo de una pulgada, por lo tanto la superficie de trazo es de 6.5" × 9".

- La dirección de los ejes está definida como:



Puede cambiarse la dirección con PLOTF.



- Las unidades del sistema de coordenadas son pulgadas, se adoptó porque las rutinas primitivas del CalComp definían las coordenadas en pulgadas. Si el usuario no define escalas, entonces la escala para los ejes X e Y será 3.25".

- Se puede definir la escala de cada eje y el origen con PLTSO.

- La diferencia de escala con respecto a los ejes no afecta mucho el trazo de superficies y gráficas planas, pero en el caso de contornos si afecta y es preferible usar escalas iguales en ambos ejes, esto se puede lograr utilizando un marco adecuado (PLTFR) o redefiniendo las escalas (PLTSO).

- El almacenamiento de los datos para graficar se realiza en arreglos de una y dos dimensiones, cada uno de ellos tienen asociados números que indican su tamaño real. Algunas subrutinas emplean esos números para calcular la posición del dato, por lo cual los números asociados a los arreglos cuando se transmiten a las subrutinas deben de ser los correctos ya que de lo contrario aparecen errores o la gráfica generada es totalmente distinta a la que se desea.

- Se asocia también a los arreglos dos números que indican el valor mínimo (cota inferior) y el valor máximo (cota superior) del intervalo del rango que se desea graficar.

- Se ha cuidado que la lista de argumentos de llamados a las subrutinas de PLOT sean uniformes. Por ejemplo, cuando se emplean arreglos se sigue el patrón: cota inferior, arreglo, cota superior, número de elementos del arreglo. Cuando se emplean arreglos bidimensionales se sigue la forma: cota inferior, arreglo, cota superior, número de renglones, número de columnas, número de secciones de renglones, número de secciones de columnas. Algunas subrutinas en el llamado tienen como último argumento el nombre de una subrutina de movimiento de la pluma (en un sistema de coordenadas específico).

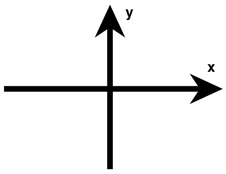
- Se emplea un sistema de coordenadas de mano derecha en las subrutinas de superficies, en el cual las partes positivas de los ejes están orientados de la siguiente forma: el eje X proporciona la coordenada horizontal a la derecha, el eje Y se aleja del observador, proporcionando la coordenada de profundidad y el eje Z proporciona la coordenada vertical hacia arriba. Cuando no se sigue este patrón, se indica explícitamente.

- Generalmente las especificaciones angulares positivas se miden en dirección contraria a las manecillas del reloj con respecto a un eje.

2.1 Subrutinas básicas de PLOT

Las subrutinas básicas definen los parámetros iniciales y realizan la comunicación básica con el graficador, están escritas en lenguaje ensamblador del microprocesador 8080 de Intel.

Subrutina: PLOTI
Llamado: CALL PLOTI
Argumentos: Ninguno
Acción: Prepara al graficador e inicia algunas variables
SX = 3.25" escala en X. XO = 0 Origen en X
SY = 3.25" escala en Y. YO = 0 Origen en Y
Orientación de la gráfica:



Factor de escala (FACT) = 100
Supone que el graficador por cada 100 movimientos se mueve una pulgada.
El origen de las coordenadas es la esquina inferior derecha del área de trazo.

Subrutina: PLTDK
Llamado: CALL PLTDK(UL,'NOMBRE ',UD)
Argumentos: UL unidad lógica, es un número que se asigna al archivo, cualquier referencia posterior al archivo se realiza con este número. Es un valor entero.
NOMBRE debe ser de por lo menos un carácter y a lo más de once (ocho posiciones para el nombre y tres para la extensión). Debe ocupar once posiciones y los carácteres deben ser en mayúsculas.
UD unidad de disco, es un número entero que designa la unidad de disco.
UD = 0, unidad de disco corriente
UD = 1, unidad de disco A
UD = 2, unidad de disco B, etc.
Acción: Define que la salida de las coordenadas y comandos generados con PLOT irán al archivo de nombre NOMBRE y unidad lógica UL que se encuentra en la unidad de disco UD.
Nota: Si se desea generar un archivo con los movimientos del graficador, el llamado a esta subrutina debe anteceder a cualquier llamado a subrutinas de PLOT. En la versión de PLOT en el lenguaje 'C', el significado de esta subrutina es diferente.

Subrutina:	PLOT
Llamado:	CALL PLOT(X,Y,L)
Argumentos:	X, Y coordenadas del punto, son variables reales e indican pulgadas.
	L indica visibilidad, es una variable lógica
	L = 0 pluma arriba .false. (no pinta)
	L =-1 pluma abajo .true. (pinta).
Acción:	Trazo de líneas, mueve la pluma a la región definida por X y Y, con la pluma en la posición definida por L. Las coordenadas X y Y se multiplican por el factor de escala FACT y se transforman a valores enteros. El punto se calcula con respecto a la orientación de la gráfica.
	Para realizar su función llama a PLOTM.

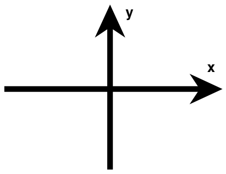
Subrutina:	PLOTM
Llamado:	CALL PLOTM(IX,IY,L)
Argumentos:	IX,IY coordenadas del punto, son variables enteras.
	L indica visibilidad, es una variable lógica
	L = 0 Pluma arriba (no pinta)
	L = 1 Pluma abajo (pinta).
	1.- Llama a PLOMT.
	2.- Prueba el estado del teclado por los carácteres:
	ctrl-C abortar el programa.
	ctrl-S suspender el programa hasta leer un ctrl-Q.
	ctrl-Q continuar con el programa.

Subrutina:	PLOMT
Llamado:	CALL PLOMT(IX,IY,L)
Argumentos:	IX,IY coordenadas del punto, son variables enteras.
	L indica visibilidad, es una variable lógica
	L = 0 pluma arriba (no pinta)
	L = 1 pluma abajo (pinta).
Acción:	1.- Si $L \neq 0$ le asocia el carácter D (down-abajo).
	Si L = 0 le asocia el carácter U (upper-arriba).
	Transforma las coordenadas (IX,IY) a sus correspondientes valores numéricos ASCII (DMP-29) o binario (CalComp), cada coordenada se almacena en un vector.
	2.- Llama a PLOTZ

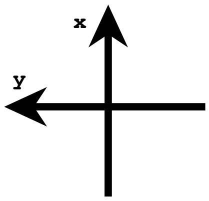
Subrutina:	PLOTZ
Llamado:	CALL PLOTZ(IAR,N)
Argumentos:	IAR arreglo lógico que contiene carácteres ASCII.
	N número de carácteres que contiene el arreglo IAR.
Acción:	Selecciona cúal es el destino de los carácteres en IAR (generalmente coordenadas o comandos del graficador).
	Si el destino es el graficador llama a PLOZT.
	Si el destino es un archivo (especificado por PLTDK) entonces escribe el arreglo IAR en el archivo y muestra en la pantalla el símbolo ==> por cada 50 carácteres que se escriban.

Subrutina: PLOZT
Llamado: CALL PLOZT(IAR,N)
Argumentos: IAR arreglo lógico que contiene carácteres ASCII.
N número de carácteres que contiene el arreglo IAR.
Acción: Transmite al graficador los N carácteres ASCII contenidos en IAR.

Subrutina: PLOTF
Llamado: CALL PLOTF(IO)
Argumentos: Si $IO = 0$ entonces



Si $IO \neq 0$ entonces



IO es un valor entero
Acción: Define la orientación de los ejes X y Y dependiendo del valor de IO.

Subrutina: PLOTO
Llamado: CALL PLOTO
Argumentos: Ninguno.
Acción: El punto previo (definido por PLOT) se toma como el origen.

Subrutina: PLOTS
Llamado: CALL PLOTS(SX,SY)
Argumentos: SX y SY son valores reales.
SX escala en el eje X.
SY escala en el eje Y.
Acción: Define el factor de escala en el eje X y en el eje Y.
Es decir, todo punto (X, Y) se transformará en $(X * SX, X * SY)$.

Subrutina: PLOTG
Llamado: CALL PLOTG(SG)
Argumentos: SG escala general es un valor real.
Acción: Define la escala general de la gráfica.
Es decir, todo punto (X,Y) se transforma en $(X * SX * SG,\ Y * SY * SG)$.

Subrutina:	PLOTP
Llamado:	CALL PLOTP
Argumentos:	Ninguno
Acción:	Guarda el origen (XO,YO) y las escalas (SX,SY). Únicamente puede hacerlo una vez.

Subrutina:	PLOTR
Llamado:	CALL PLOTR
Argumentos:	Ninguno
Acción:	Recupera el origen y las escalas guardadas por PLOTP.

Subrutina:	PLOTL
Llamado:	CALL PLOTL(SX,SY,IO)
Argumentos:	SX, SY las escalas en el eje X y Y, son valores reales. IO orientación de los ejes, es un valor entero.
Acción:	Lee las escalas de los ejes X y Y y su orientación es decir, después del llamado SX, SY e IO contienen dichos valores.

Subrutina:	TIME
Llamado:	CALL TIME(HORA)
Argumentos:	HORA es un vector de nueve entradas en el que se obtiene la hora en forma de carácteres ASCII.
Acción:	Obtiene la hora del reloj de tiempo real del sistema y la deposita en el vector HORA. La versión de PLOT en el lenguaje 'C' no tiene esta rutina.

Subrutina:	DATE
Llamado:	CALL DATE(FECHA)
Argumentos:	FECHA es un vector de nueve entradas en el que se obtiene la fecha en forma de carácteres ASCII.
Acción:	Obtiene la fecha que tiene definida el sistema y la deposita en el vector FECHA. La versión de PLOT en el lenguaje 'C' no tiene esta rutina.

2.2 Subrutinas de movimiento de la pluma

La subrutina principal de movimiento de la pluma es PLOT(X,Y,IP), en la que X y Y indican las coordenadas cartesianas absolutas con respecto al origen (DMP-29) o las coordenadas relativas con respecto al punto previo (CalComp 1044) del punto al que se desea mover la pluma (las coordenadas especifican pulgadas), IP indica si la pluma toca o no toca el área de trazo (pinta o no pinta).

La gran variedad de los sistemas de coordenadas y el escalamiento de la gráfica muestran lo poco práctico que es ajustarse al formato tamaño carta, la alternativa es seleccionar el cuadrado unitario como el estándar a pesar de la incompatibilidad con la forma de una hoja tamaño carta. Las rutinas de PLOT trabajan internamente en el cuadrado unitario, y es hasta el momento en el que van a trazar cuando realizan la transformación adecuada para que el trazo quede en la hoja tamaño carta.

Antes de realizar un movimiento se llama a las subrutinas de transformación de coordenadas, ellas esperan que las coordenadas estén definidas en el sistema especificado y que su rango de variación sea el intervalo [0,1], entonces las transforman a coordenadas cartesianas y definen los movimientos adecuados a una hoja tamaño carta.

Las subrutinas de movimiento de la pluma son:

PLTCA	Movimiento de la pluma en coordenadas cartesianas.
PLTPO	Movimiento de la pluma en coordenadas polares.
PLTSP	Movimiento de la pluma en coordenadas esféricas.
PLTEL	Movimiento de la pluma en coordenadas elípticas.
PLTTR	Movimiento de la pluma en coordenadas triangulares.

2.3 Descripción de las subrutinas

Subrutina:	PLTCA
Llamado:	CALL PLTCA(X, Y, L)
Argumentos:	X, Y coordenadas cartesianas del punto, son valores reales.
	$0.0 \leq X \leq 1.0$
	$0.0 \leq Y \leq 1.0$
	L indicador de visibilidad, es un valor lógico.
	$L = 0$ no traza (pluma arriba)
	$L \neq 0$ traza (pluma abajo)
Acción:	Movimiento de la pluma en coordenadas cartesianas.
	Transforma las coordenadas cartesianas X,Y definidas en el cuadrado unitario a los valores que deben tener en el área de trazo, tomando en cuenta la orientación de los ejes.

Subrutina:	PLTPO
Llamado:	CALL PLTPO(T, R, L)
Argumentos:	T, R coordenadas polares del punto, son valores reales. $0.0 \leq \mathrm{T} \leq 1.0$ $0.0 \leq \mathrm{R} \leq 1.0$ L indicador de visibilidad, es un valor lógico. L = 0 no traza (pluma arriba) L $\neq$0 traza (pluma abajo)
Acción:	Movimiento de la pluma en coordenadas polares. Transforma las coordenadas polares T, R definidas en el cuadrado unitario a los valores que deben tener en coordenadas cartesianas en el área de trazo, tomando en cuenta la orientación de los ejes. Debido a que la escala de los ejes del área de trazo puede ser diferente, se acota el radio, es decir, el radio máximo será igual al valor mínimo de la escala en X y la escala en Y. Además, el valor del ángulo sólo toma su valor principal, es decir, sólo valores entre cero y dos PI, por lo que hay que tener cuidado al definir segmentos menores o mayores a ese intervalo.

Subrutina:	PLTSP
Llamado:	CALL PLTSP(T, P, L)
Argumentos:	T, P coordenadas esféricas angulares del punto, son valores reales. $0.0 \leq \mathrm{T} \leq 1.0$ $0.0 \leq \mathrm{P} \leq 1.0$ L indicador de visibilidad, es un valor lógico. L = 0 no traza (pluma arriba) L $\neq$ 0 traza (pluma abajo)
Acción:	Movimiento de la pluma en coordenadas esféricas. Transforma las coordenadas angulares esféricas T, P definidas en el cuadrado unitario a los valores que deben tener en coordenadas cartesianas en el área de trazo, tomando en cuenta la orientación de los ejes. Se supone que la esfera está orientada de forma tal que Z apunta hacia arriba, X a la izquierda y Y apunta en dirección al observador. Un punto de la esfera es visible si $Y \geq 0.0$. El radio de la esfera es constante y su valor es el valor mínimo de la escala en X y la escala en Y.

Subrutina: PLTEL
Llamado: CALL PLTEL(X, E, L)
Argumentos: X, E coordenadas elípticas del punto, son valores reales.

$0.0 \leq$ X ≤ 1.0
$0.0 \leq$ E ≤ 1.0

L Indicador de visibilidad, es un valor lógico.
L = 0 no traza (pluma arriba)
L $\neq$ 0 traza (pluma abajo)

Acción: Movimiento de la pluma en coordenadas elípticas.
Transforma las coordenadas elípticas X, E definidas en el cuadrado unitario a los valores que deben tener en coordenadas cartesianas en el área de trazo, tomando en cuenta la orientación de los ejes.
La distancia del origen al foco de la elipse del semieje mayor se considera constante y su valor es el valor mínimo de la escala en X y la escala en Y dividido entre dos. Se supone que el sistema está orientado de forma tal que Z apunta hacia el observador, Y hacia arriba y X a la derecha.
PLTEL grafica la proyección de X, E al plano X-Y

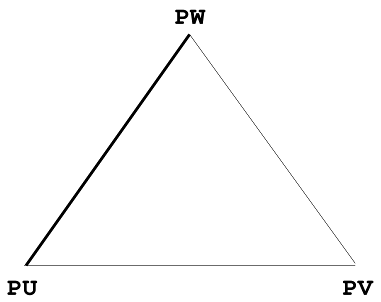
Subrutina: PLTTR
Llamado: CALL PLTTR(U, V, L)
Argumentos: U,V coordenadas triangulares del punto, son valores reales.

$0.0 \leq$ U ≤ 1.0
$0.0 \leq$ V ≤ 1.0

L indicador de visibilidad, es un valor lógico.
L = 0 no traza (pluma arriba)
L $\neq$ 0 traza (pluma abajo)

Acción: Movimiento de la pluma en coordenadas triangulares.
Transforma las coordenadas triangulares U,V definidas en el cuadrado unitario a los valores que deben tener en coordenadas cartesianas en el área de de trazo, tomando en cuenta la orientación de los ejes y grafica su proyección en el plano X-Y.
Las coordenadas triangulares (U, V, W) se definen como las distancias de un punto desde cada uno de los lados de un triángulo equilátero. Un triángulo equilátero tiene la propiedad de que la suma de tales distancias es igual a su altura. En PLOT se selecciona un triángulo equilátero de altura uno. Así, tenemos que $U + V + W = 1$, y si se fijan dos coordenadas cualesquiera se puede de determinar la tercera. En PLOT se selecciona U y V como las coordenadas y W se determina implícitamente por:

$$W = 1 - U - V$$



En la figura previa, W es la distancia desde un punto interior al lado PU-PV.

Capítulo 3

Gráficas Simples

Se describen en este grupo las rutinas siguientes:

- Rutinas que inician y terminan una sesión de graficado.

  Algunas de las rutinas que inician y terminan una sesión de graficado aparecen siempre en los programas de los usuarios.

- Rutinas que trazan curvas planas.

  Una curva plana contiene un conjunto infinito de puntos y por lo tanto su representación también es infinita. Sin embargo, es posible obtener una buena aproximación de la curva por medio de un conjunto finito de puntos. Así, el trazo de curvas planas requiere que se proporcione un conjunto de parejas (X,Y) (un par de números que son las coordenadas de un punto en un sistema de coordenadas), cada pareja define un punto de la curva. El conjunto de parejas (X, Y) se puede obtener por evaluar una función de la forma $Y = f(X)$ ó $X = u(T)$, $Y = v(T)$ con T definido en un intervalo, o por recolectar datos desde mediciones analógicas. Por lo anterior, debe notarse que el conjunto de parejas tiene un orden y que el trazo resultante se parecerá más a la curva deseada entre mayor sea el número de parejas que se proporcionan.

  PLOT generará la curva correspondiente al conjunto de parejas por unir las parejas con líneas rectas. Para el trazo de la curva PLOT requiere dos arreglos, uno contendrá los valores de la variable X y el otro los valores de la variable Y. Asociados a cada uno de los arreglos se tendrá un valor mínimo y un valor máximo que indican la porción de la curva que se desea ver. También se tendrá una variable que indica el número de elementos que contienen los arreglos y finalmente el nombre de la subrutina de movimiento de la pluma.

- Rutinas que trazan símbolos.

  En PLOT se tienen dos conjuntos de símbolos ASCII, (letras simples y letras dobles) que se pueden emplear para etiquetar las gráficas. El inicio de la cadena de símbolos se especifica con un par de números que pueden indicar pulgadas o un valor en el intervalo cero o uno.

3.1 Subrutinas de uso general

Algunas de las subrutinas de uso general siempre aparecen en los programas de los usuarios.

PLT00	Preparación del graficador y del paquete PLOT.
PLTFR	Enmarca la zona de trabajo.
PLTHO	Desaloja la pluma, la coloca en el origen.
PLTEJ	Desaloja la pluma, la coloca en la siguiente posición.
PLTSO	Define las escalas y el origen.
PLTEU	Genera la matriz de rotación correspondiente a los ángulos de Euler.

3.2 Subrutinas para gráficas planas

Estas subrutinas generalmente grafican datos almacenados en arreglos.

PLTGA	Traza una gráfica tomando los puntos desde dos arreglos.
PLTGD	Grafica datos (coloca marcas).
PLTFI	Grafica un arreglo cuyos valores en X son enteros.
PLTIG	Trazo en incrementos, no utiliza arreglos. Traza una gráfica punto por punto.
PLTFM	Traza marcas.

3.3 Subrutinas para el trazo de símbolos

- Especificación de las coordenadas en el intervalo [0,1] y en un sistema de coordenadas específico (PLTCA, PLTPO, etc.).

PLTTP	Traza en símbolos ASCII el contenido de un vector.
PLTLE	Traza en símbolos simples ASCII el contenido de un vector.
PLTBP	Traza en símbolos dobles ASCII el contenido de un vector.
PLTNU	Traza los símbolos ASCII que corresponden a un número de punto flotante.

- Especificación de las coordenadas en pulgadas, la rutina de movimiento de la pluma es PLOT(X,Y,L).

PLTTS	Traza en símbolos ASCII el contenido de un vector.
SYMBOL	Traza en símbolos simples ASCII el contenido de un vector.
PLTSB	Traza en símbolos dobles ASCII el contenido de un vector.
NUMBER	Traza los símbolos ASCII que corresponden a un número de punto flotante.

- Etiquetar una gráfica.

PLTLA	Letrero en la parte inferior de la gráfica.
PLTLLA	Letrero en la parte superior e inferior de la gráfica.

3.4 Secuencia de llamado entre las subrutinas de planas

PLT00	—>	PLOTI
		PLOTG
		PLOTO
		PLOT
PLTFR	—>	PLOTO
		PLOT
PLTHO	—>	LEOUT
PLTEJ	—>	PLOT
		LEOUT
PLTSO	—>	PLOT
		PLOTP
		PLOTO
		PLOTS

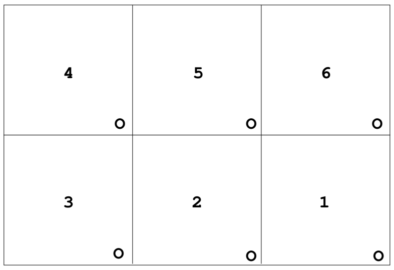
PLTEU				
PLTGA	—> PL			
PLTGD	—> PLTFM			
PLTFI	—> PL			
PLTIG	—> PL			
	PLTFM			
PLTTP	—> PL			
PLTTS	—> PLTMV			
PLTLE	—> PLTTP	—> PL		
PLTBP	—> PLTTP	—> PL		
PLTNU	—> PLTLE			
SYMBOL	—> PLTTS	—> PLTMV		
PLTSB	—> PLTTD	—> PLTMV		
NUMBER	—> SYMBOL	—> PLTTS	—> PLTMV	
PLTLA	—> DATE			
	TIME			
	PLOTL			
	SYMBOL			
PLTLLA	—> SYMBOL			

3.5 Descripción de las subrutinas

Subrutina: PLT00
Llamado: CALL PLT00
Argumentos: Ninguno.
Acción: Inicia los parámetros de PLOT.

Para iniciar una serie de gráficas debe ser llamada al menos una vez. Solicita del usuario se proporcionen los parámetros:

LP indicador de la zona de trabajo y origen. Las zonas de trabajo son seis.



O indica donde queda definido el origen de las coordenadas con respecto a la zona de trabajo.

SG escala general del trazo. La escala general de trazo está en función a la hoja tamaño carta.

Subrutina: PLTFR
Llamado: CALL PLTFR(LO)
Argumentos: LO define el tipo de acción que realizará PLTFR:

LO=0	define que el origen de las coordenadas será el centro de un marco ficticio de 8.5"×11".
LO=1	traza un marco de 8.5"×11" y define el origen de las coordenadas en el centro del marco.
LO=2	traza un marco de 6.5"×9" dentro de un marco ficticio de 8.5"×11" y define el origen de las coordenadas en el centro del marco.
LO=3	traza dos marcos uno de 8.5"×11" y otro de 6.5"×9" y define el origen de las coordenadas en el centro de los marcos.
LO=4	traza un marco de 8.5"×8.5" dentro de un marco ficticio de 8.5"×11" y define el origen de las coordenadas en el centro del marco.
LO=5	traza un marco de 6.5"×6.5" dentro de un marco ficticio de 8.5"×11" y define el origen de las coordenadas en el centro del marco.
LO=6	traza dos marcos uno de 8.5"×8.5" y otro de 6.5"×6.5" dentro de un marco ficticio y define el origen de las coordenadas en el centro de los marcos.

Acción: Enmarca la zona de trabajo.

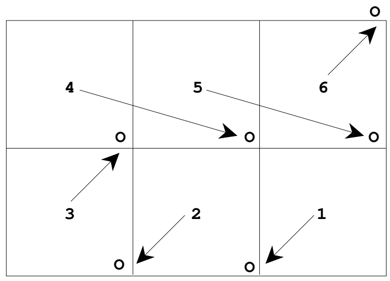
Dependiendo del valor de LO define la zona de trabajo enmarcándola y definiendo el origen de las coordenadas. El origen de las coordenadas para el trazo del marco generalmente lo define PLT00.

Subrutina: PLTHO
Llamado: CALL PLTHO
Argumentos: Ninguno.
Acción: Termina una sesión de graficado.

Concluye una sesión de graficado y coloca la pluma en el origen del marco (la esquina inferior derecha) lo cual permite observar la gráfica o retirar la hoja del graficador. Además, cierra el archivo de comandos del graficador si se abrió con PLTDK.

Subrutina: PLTEJ
Llamado: CALL PLTEJ
Argumentos: Ninguno.
Acción: Termina una sesión de graficado.

Concluye una sesión de graficado y coloca la pluma al inicio de la siguiente zona de trabajo lo cual permite observar la gráfica o retirar la hoja del graficador. Incrementa el indicador de la zona de trabajo. Además, cierra el archivo de comandos del graficador si se abrió con PLTDK.



O indica donde queda definido el origen de las coordenadas con respecto a la zona de trabajo.

Subrutina: PLTSO
Llamado: CALL PLTSO(SX, SY, OX, OY)
Argumentos: SX, SY factor de escala en X y en Y
OX, OY coordenadas del origen.
Acción: Define las escalas y el origen de la gráfica.
Salva el origen anterior, define el nuevo origen y define las escalas.

Subrutina: PLTEU
Llamado: CALL PLTEU(O, E1, E2, E3)
Argumentos: O(3,3) matriz de rotación
E1, E2, E3 ángulos de Euler, se especifican en grados.
Acción: Genera la matriz de rotación de Euler.
Construye la matriz de rotación O(3,3) correspondiente a los ángulos de Euler E1, E2 y E3, $O = R1 \times R2 \times R3$. R1 y R3 son rotaciones en sentido contrario a las manecillas del reloj en el plano X-Y en un ángulo E1 y E3 respectivamente. R2 es una rotación en el plano Y-Z en sentido contrario de las manecillas del reloj en un ángulo E2. La forma de realizar las rotaciones es la siguiente:
En un sistema de coordenadas cartesianas de mano derecha con ejes X, Y y Z, E1 es el ángulo de rotación en dirección contraria a las manecillas del reloj en el plano X-Y alrededor del eje Z, lo que nos proporciona un nuevo conjunto de ejes (X', Y', Z'=Z). Después se da una rotación en el plano Y'-Z' en un ángulo E2 en dirección contraria a las manecillas del reloj alrededor del eje X', lo que nos da un nuevo conjunto de ejes (X"=X', Y", Z"). La última rotación en dirección contraria a las manecillas del reloj por un ángulo E3 es alrededor del eje Z en el plano X"-Y". Esta subrutina se emplea principalmente con los programas de superficies.

Subrutina: PLTGA
Llamado: CALL PLTGA(X1, XX, X2, Y1, YY, Y2, NP, PL)
Argumentos: X1 valor mínimo para los valores de X.
XX arreglo vectorial que contiene los valores de X de la curva.
X2 valor máximo para los valores de X.
Y1 valor mínimo para los valores de Y.
YY arreglo vectorial que contiene los valores de Y de la curva.
Y2 valor máximo para los valores de Y.
NP número de elementos de los vectores X e Y.
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Traza una gráfica tomando los puntos desde dos arreglos.
Traza una gráfica por conectar los puntos de los datos con líneas rectas. Los puntos que definen la gráfica se toman desde los arreglos XX y YY que contienen los valores de X y los valores de Y. Las escalas respectivas se indican con los valores asignados a los márgenes (X1, X2, Y1, Y2) de la gráfica. Se obtiene una representación parcial de la curva si los valores de los márgenes son menores a los valores máximo y mínimo de los vectores XX y YY. Los valores en XX se asocian a la coordenada horizontal y los valores en YY se asocian a la coordenada vertical en el área de trazo.

Subrutina: PLTGD
Llamado: CALL PLTGD(X1, XX, X2, Y1, YY, Y2, NP, PL)
Argumentos: X1 valor mínimo para los valores de X.

XX arreglo vectorial que contiene los valores de X de la curva.
X2 valor máximo para los valores de X.
Y1 valor mínimo para los valores de Y.
YY arreglo vectorial que contiene los valores de Y de la curva.
Y2 valor máximo para los valores de Y.
NP número de elementos de los vectores X e Y.
PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Grafica datos (coloca marcas).

Genera una representación con marcas (dos rectas perpendiculares). Los puntos que definen el centro de cada marca se toman desde los arreglos XX y YY que contienen los valores de X y los valores de Y. Las escalas respectivas se indican con los valores asignados a los márgenes (X1, X2, Y1, Y2) de la gráfica. Se obtiene una representación parcial si los valores de los márgenes son menores a los valores máximo y mínimo de los vectores XX y YY. Los valores en XX se asocian a la coordenada horizontal y los valores en YY se asocian a la coordenada vertical en el área de trazo.

Subrutina: PLTFI
Llamado: CALL PLTFI(Y1, YY, Y2, NP, PL)
Argumentos: Y1 valor mínimo para los valores de Y.
YY arreglo vectorial que contiene los valores de Y de la curva.
Y2 valor máximo para los valores de Y.
NP número de elementos del vector Y.
PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Grafica un arreglo cuyos valores en X son enteros.

Traza una gráfica por conectar los puntos de los datos con líneas rectas. Los puntos que definen la gráfica se toman desde el arreglo YY que contiene los valores de Y. Los valores de X se supone que son números enteros en el intervalo 1 a NP. Las escalas respectivas se indican con los valores asignados a los márgenes (1, NP, Y1, Y2) de la gráfica. Se obtiene una representación parcial de la curva si los valores de los márgenes son menores a los valores máximo y mínimo de los vectores XX y YY. Los valores en YY se asocian a la coordenada vertical en el área de trazo.

Subrutina: PLTIG
Llamado: CALL PLTIG(X, Y, L, PL)
Argumentos: X primera coordenada
Y segunda coordenada
L acción a realizar:
L=1 (X,Y) definen el límite inferior
L=2 (X,Y) definen el límite superior
L=3 (X,Y) es el punto inicial de la curva
L=4 (X,Y) son los puntos subsecuentes de la curva
L=5 ejes rectangulares a través de (X,Y)
L=6 marca pequeña a través de (X,Y)
L=7 marca grande a través de (X,Y)
L=8 ejes rect. a través de (X,Y) con etiquetas (X,Y)
L=9 ejes rect. a través de (X,Y) con etiquetas (X,I)

L=10 ejes rect. a través de (X,Y) con etiquetas (X,Y), Y invertida
L=11 ejes rect. a través de (X,Y) con etiquetas (X,I), I invertida
PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Traza una gráfica punto por punto.

Los límites de la gráfica deben ser definidos antes de iniciar el trazo o antes de trazar las marcas o los ejes. Ya que se han definido los límites, el punto inicial de la gráfica debe ser definido con L=3 y los puntos subsecuentes con L=4. Si se inicia una nueva curva o para crear huecos en la gráfica se debe seguir la secuencia L=3 y L=4. El radio de la marca pequeña es 0.015" y el de la marca grande es 0.020".

Subrutina: PLTFM
Llamado: CALL PLTFM(X, Y, R, PL)
Argumentos: X, Y coordenadas del punto donde se colocará la marca.
R define el tamaño del radio de la marca.
Acción: Traza una marca (dos rectas perpendiculares).
Tanto las coordenadas como el radio se especifican en pulgadas.

Subrutina: PLTTP
Llamado: CALL PLTTP(X, Y, H, TEXT, THE, NC, ITIPO, PL)
Argumentos: X, Y coordenadas donde empezará el trazo del primer carácter del arreglo TEXT.
H el valor absoluto de H es la altura que tendrán los carácteres al momento del trazo.
Si $H > 0$ entonces X y Y definen como el inicio del trazo la esquina inferior izquierda del carácter.
Si $H < 0$ entonces al momento del trazo el carácter tendrá como centro a X y Y.
Si $H = 0$ la rutina no realiza ningún trazo.
TEXT arreglo que contiene los carácteres a trazar.
THE ángulo en grados (se mide en sentido contrario a las manecillas del reloj a partir del eje X) que indica la inclinación que tendrán los carácteres contenidos en TEXT.
NC número de carácteres que contiene TEXT.
ITIPO número que indica el tipo de símbolos a trazar.
ITIPO = 1 letras simples
ITIPO = 2 letras dobles
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Traza en símbolos ASCII el contenido de un vector.
Las coordenadas deben estar en el intervalo [0,1] y en el sistema de coordenadas especificado por PL.

Subrutina: PLTLE
Llamado: CALL PLTLE(X, Y, H, TEXT, THE, NC, PL)
Argumentos: X, Y, H, TEXT, THE, NC y PL tienen el mismo significado que los argumentos de PLTTP.
Acción: Traza en símbolos simples ASCII el contenido de un vector.
Las coordenadas deben estar en el intervalo [0,1] y en el sistema de coordenadas específicado por PL.

Subrutina: PLTBP
Llamado: CALL PLTBP(X, Y, H, TEXT, THE, NC, PL)
Argumentos: X, Y, H, TEXT, THE,NC y PL tienen el mismo significado que los argumentos de PLTTP.
Acción: Traza en símbolos dobles ASCII el contenido de un vector.
Las coordenadas deben estar en el intervalo [0,1] y en el sistema de coordenadas específicado por PL.

Subrutina: PLTNU
Llamado: CALL PLTNU(X, Y, H, R, THE, ID, PL)
Argumentos: X, Y, H, THE y PL tienen el mismo significado que los argumentos de PLTTP.
R variable que contiene el número que se representará.
ID indica el número de dígitos con el que se representará el contenido de R.
Acción: Traza los símbolos ASCII que corresponden al número de punto flotante que se encuentra almacenado en la variable R, el número de dígitos con los que se expresará el número se indica con ID. Las coordenadas deben estar en el intervalo [0,1] y en el sistema de coordenadas especificado por PL.

Subrutina: PLTTS
Llamado: CALL PLTTS(X, Y, H, TEXT, THE, NC, ITIPO)
Argumentos: X, Y coordenadas donde empezará el trazo del primer carácter del arreglo TEXT.
H el valor absoluto de H es la altura que tendrán los carácteres al momento del trazo.
Si $H > 0$ entonces X y Y definen como el inicio del trazo la esquina inferior izquierda del carácter.
Si $H < 0$ entonces al momento del trazo el carácter tendrá como centro a X y Y.
Si $H = 0$ la rutina no realiza ningún trazo.
TEXT arreglo que contiene los carácteres a trazar.
THE ángulo en grados (se mide en sentido contrario a las manecillas del reloj a partir del eje X) que indica la inclinación que tendrán los carácteres contenidos en TEXT.
NC número de carácteres que contiene TEXT.
ITIPO número que indica el tipo de símbolos a trazar.
ITIPO = 1 letras simples
ITIPO = 2 letras dobles
Acción: Traza en símbolos ASCII el contenido de un vector. Las coordenadas se especifican en pulgadas y la rutina de movimiento de la pluma es PLOT(X,Y,L).

Subrutina: SYMBOL
Llamado: CALL SYMBOL(X, Y, H, TEXT, THE, NC)
Argumentos: X, Y, H, TEXT, THE y NC tienen el mismo significado que los argumentos de PLTTS.
Acción: Traza en símbolos simples ASCII el contenido de un vector.
Las coordenadas se especifican en pulgadas y la rutina de movimiento de la pluma es PLOT(X,Y,L).

Subrutina: PLTSB
Llamado: CALL PLTSB(X, Y, H, TEXT, THE, NC)
Argumentos: X, Y, H, TEXT, THE y NC tienen el mismo significado que los argumentos de PLTTS.
Acción: Traza en símbolos dobles ASCII el contenido de un vector.
Las coordenadas se especifican en pulgadas y la rutina de movimiento de la pluma es PLOT(X,Y,L).

Subrutina: NUMBER
Llamado: CALL NUMBER(X, Y, H, R, THE, ID)
Argumentos: X, Y, H, THE tienen el mismo significado que los argumentos de PLTTS.
R variable que contiene el número que se representará.
ID indica el número de dígitos con el que se representará el contenido de R.
Acción: Traza los símbolos ASCII que corresponden al número de punto flotante que se encuentra almacenado en la variable R, el número de dígitos con los que se expresará el número se indica con ID. Las coordenadas se especifican en pulgadas y la rutina de movimiento de la pluma es PLOT(X,Y,L).

Subrutina: PLTLA
Llamado: CALL PLTLA(TEXTO, NCT)
Argumentos: TEXTO arreglo que contiene el letrero, es un arreglo de tipo lógico.
NCT1 variable entera que indica el número de carácteres que contiene el letrero en TEXTO.
Acción: Etiqueta la gráfica.
Coloca un letrero en la gráfica en la parte inferior junto con una leyenda y la fecha y hora. Ambos letreros son centrados. En la versión de PLOT en 'C' el arreglo TEXTO es de tipo 'CHAR'.

Subrutina: PLTLLA
Llamado: CALL PLTLLA(TEXTO1, NCT1, TEXTO2, NCT2, ITM)
Argumentos: TEXTO1 arreglo que contiene el letrero de la parte superior, es un arreglo de tipo lógico.
NCT1 variable entera que indica el número de carácteres que contiene el letrero en TEXTO1.
TEXTO2 arreglo que contiene el letrero de la parte inferior, es un arreglo de tipo lógico.
NCT1 variable entera que indica el número de carácteres que contiene el letrero en TEXTO2.
ITM variable entera que indica el tipo de marco que se definió con PLTFR.
Acción: Etiqueta la gráfica.
Coloca dos letreros en la gráfica, uno en la parte superior y otro en la parte inferior, en la parte inferior también se coloca la fecha y la hora. Ambos letreros son centrados. En la versión de PLOT en 'C' los arreglos TEXTO1 y TEXTO2 son de tipo 'CHAR'.

Capítulo 4

Contornos

Si a una superficie la cortamos con un plano, a la curva generada por la intersección del plano con la superficie se le llama "contorno". Si suponemos que la superficie tiene una ecuación de la forma z = f(x,y), entonces la ecuación que define un contorno es:

$$f(x,y) - c = 0$$

donde c es un valor constante.

Todas las rutinas de contornos requieren que el usuario proporcione un arreglo rectangular de datos (matriz), los cuales definen las altitudes de la superficie. Para una representación adecuada, la matriz debe ser al menos de 40×40 y de preferencia de 89×89 o más, aunque el tamaño de la matriz depende de la cantidad de memoria y del tamaño del programa.

PLOT traza un contorno de nivel ZC a la vez, empleando el método de la triangulación para la obtención del contorno.

Para obtener un contorno de nivel ZC se seleccionan celdas de la matriz de forma tal que formen un triángulo. Si i y j indican el número de renglón y el número de columna en la matriz, los vértices de un triángulo i en la matriz serán: (i,j), $(i+1,j)$, $(i,j+1)$ y se le llama triángulo superior. El triángulo inferior correspondiente sería $(i+1,j)$, $(i+1,j+1)$, $(i,j+1)$. Ya que deseamos un contorno de nivel de valor ZC, entonces le restamos a cada vértice este valor, lo cual nos deja el valor del vértice como positivo, negativo o cero. En función al signo de cada vértice del triángulo, se determina si el contorno cruza o no cruza el triángulo. Durante el proceso de localización del contorno se supone que el signo de los vértices es positivo, negativo e indeterminado. Por los primeros dos vértices entra el contorno; el signo del tercer vértice determina el borde de salida del contorno. Si el contorno cruza el triángulo, entonces se determina con interpolación lineal por donde pasa el contorno y traza. Cuando se localiza un contorno se anota cual es el triángulo inicial, ya que el contorno puede ir en dos direcciones en el trazo, primero se sigue una dirección y después la otra.
En PLOT tenemos dos rutinas para el trazo de contornos simples que son PLTKP y PLTKC.

Los trazos de contornos no dan ellos mismos una buena imagen de las depresiones y elevaciones de la superficie a menos que se les agregue información adicional, en PLOT84 se utilizan dos técnicas que son:

- Dibujar líneas de sombra (rayado) en las direcciones X o Y formando una red que cubre intervalos especificados del contorno. Esto lo realizan las subrutinas PLTKX y PLTKY.

- Relieve ortográfico.

  Esta técnica fue introducida por Tanaka K en 1964. Cuando se ejecuta adecuadamente es capaz de dar la impresión de elevación. La idea esencial del relieve ortográfico es imaginar una fuente de luz colocada sobre la superficie cuyos rayos se extienden e iluminan los caminos cruzando las líneas de contorno en un ángulo. Si se inclina la superficie de forma tal que una línea horizontal en la superficie sea paralela a un rayo de la fuente de luz, entonces los rayos incidirán (descansarán) a lo largo de los contornos. Así, si se trazan los contornos correspondientes a la superficie inclinada con un color claro, y después se superpone a ellos el contorno normal con un color obscuro, la impresión general será la de iluminación lateral. Es decir, las partes laterales de los cerros que se oponen a la fuente de luz aparecerán obscuros y aquellos alejados de la fuente se verán claros.

Este efecto es adecuado cuando se tienen fotografías aéreas de gran altitud. La principal desventaja del método de relieve ortográfico es el gran número de contornos que se requieren para realizar el apropiado efecto visual. Tanaka recomienda tener entre 30 y 70 contornos inclinados por centímetro. La rutina que realiza esta acción es PLTOR.

Si la superficie contiene paredes empinadas, picos puntiagudos o cañones estrechos, los datos deben ser lo suficientemente densos para que contengan los detalles suficientes para separar las líneas de contornos de esas áreas.

4.1 Subrutinas que grafican contornos

El usuario generalmente llamará a estas rutinas:

PLTKP	Grafica contornos simples.
PLTKC	Grafica contornos de funciones complejas.
PLTKX	Traza rayas en la dirección X.
PLTKY	Traza rayas en la dirección Y.
PLTOR	Grafica relieves ortográficos.

El usuario algunas veces llamará a estas rutinas:

KONSC	Traza un contorno simple.
KONOC	Traza un relieve ortográfico.

4.2 Secuencia de llamado entre las subrutinas de contornos

PLTKP	->	KONSC	->	KONC3
				IZIGN
				SET
				IASK
				KONIT
				KONXV
				KONNC
				KONSA
				RESET
				KONRE
				PL
PLTKC	->	KONSC	->	PL
PLTOR	->	KONOC	->	PL
PLTKX	->	PL		
		PLTIL		
PLTKY	->	PL		

PLTIL

4.3 Descripción de las subrutinas

Subrutina: PLTKP
Llamado: CALL PLTKP(Z1 ,ZE, Z2, NZ, KX, NX, KY, NY, PL)
Argumentos: Z1, Z2 intervalo del rango de ZE del que se desean los contornos.
ZE(NX,NY) arreglo matricial que contiene los valores que generan el contorno, estos valores deben tener la forma $Z = F(X,Y)$
NX, NY dimensión de ZE, número de renglones y número de columnas.
KX número de bloques en la dirección X, KX = (NX/64)+1
KY número de bloques en la dirección Y, KY = (NY/64)+1
NZ número de contornos en el rango (Z1,Z2), NZ debe ser mayor que 1
PL subrutina de movimiento de la pluma (PLTCA o PLTPO).
Acción: Grafica contornos simples.
Dibuja los contornos de los datos almacenados en el arreglo ZE. Los contornos se dibujan empezando en $Z = Z1$ continúan con pasos $DZ = (Z2 - Z1)/(NZ - 1)$ hasta $Z = Z2$.

Subrutina: PLTKC
Llamado: CALL PLTKC(Z1 ,ZE, Z2, NZ, KX, NX, KY, NY, PL)
Argumentos: Z1, Z2 intervalo del rango de ZE del que se desean los contornos.
ZE(NX,NY) arreglo con los valores del argumento de la función compleja.
NX, NY dimensión de ZE, número de renglones y número de columnas.
KX número de bloques en la dirección X, KX = (NX/64)+1.
KY número de bloques en la dirección Y, KY = (NY/64)+1.
NZ Los contornos del argumento son en intervalos de 30 grados desde -180 a 180 grados. Debe ser mayor a 1.
PL subrutina de movimiento de la pluma (PLTCA o PLTPO).
Acción: Grafica los contornos de fase de una función compleja.
Dibuja los contornos de los datos almacenados en el arreglo ZE. Los valores en ZE son el argumento del número complejo, es decir: tienen la forma $Z = ARG(F(W))$, $F(W)$ es una función compleja y ARG es una función que devuelve el argumento de un número complejo.

Subrutina: PLTKX
Llamado: CALL PLTKX(Z1, ZE, Z2, NX, NY, PL)
Argumentos: Z1, Z2 intervalo del rango de ZE que se marcará.
ZE(NX,NY) arreglo matricial que contiene los valores que generan el contorno, estos valores deben tener la forma Z=F(X,Y).
NX, NY dimensión de ZE, número de renglones y número de columnas.
PL subrutina de movimiento de la pluma (PLTCA o PLTPO).
Acción: Líneas de sombra en la dirección X.
Traza líneas horizontales cuya posición corresponde a los puntos del arreglo y cuya extensión se determina por interpolación lineal.

Subrutina: PLTKY
Llamado: CALL PLTKY(Z1, ZE, Z2, NX, NY, PL)
Argumentos: Z1, Z2 intervalo del rango de ZE del que se desean los contornos.
ZE(NX,NY) arreglo matricial que contiene los valores que generan el contorno, estos valores deben tener la forma Z=F(X,Y).
NX, NY dimensión de ZE, número de renglones y número de columnas.
PL subrutina de movimiento de la pluma (PLTCA o PLTPO).
Acción: Líneas de sombra en la dirección Y.
Traza líneas verticales cuya posición corresponde a los puntos del arreglo y cuya extensión se determina por interpolación lineal.

Subrutina: PLTOR
Llamado: CALL PLTOR (Z1, ZE, Z2, NZ, KX, NX, KY, NY, PL)
Argumentos: Z1, Z2 intervalo del rango de ZE del que se desean los contornos.
ZE(NX,NY) arreglo matricial que contiene los valores que generan el contorno, estos valores deben tener la forma Z=F(X,Y).
NX, NY dimensión de ZE, número de renglones y número de columnas.
KX número de bloques en la dirección X, KX = (NX/64)+1.
KY número de bloques en la dirección Y, KY = (NY/64)+1.
NZ número de contornos en el rango (Z1,Z2), NZ debe ser mayor que 1.
PL subrutina de movimiento de la pluma (PLTCA o PLTPO).
Acción: Relieve ortográfico.

4.4 Subrutinas básicas de contornos

Están escritas en lenguaje ensamblador del microprocesador 8080 de Intel, son subrutinas auxiliares de las rutinas que trazan contornos, el usuario generalmente no tiene necesidad de llamarlas. En el trazo de contornos se emplea un arreglo en el que se indica por donde pasa el contorno. El arreglo tiene la forma IC(I,J,K), en el que, si $K = 1$ indica triángulo superior, si $K = 2$ indica triángulo inferior.

Subrutina: SET
Llamado: CALL SET(L,K)
Argumentos: L es una posición dentro de un arreglo lógico.
K es un entero, indica el bit que se modificará.
Acción: Define el valor del K-ésimo bit del arreglo L a 1.
Si K es mayor a 8 entonces a L se le suma $K/8$ y el resto es el bit que se define.
Ejemplo:
Si $K = 11$, se enciende el bit 3 de $L + 1$.

Subrutina: RESET
Llamado: CALL RESET(L,K)
Argumentos: L es una posición dentro de un arreglo lógico.
K es un entero, indica el bit que se modificará.
Acción: Define el valor del K-ésimo bit del arreglo L a 0.
Si K es mayor a 8 entonces a L se le suma $K/8$ y el resto indica el bit que se define a 0. Ejemplo:
Si $K = 18$, se define el bit 2 a 0 de $L + 2$.

Función: IASK
Llamado: IASK(L,K)
Argumentos: L es una posición dentro de un arreglo lógico.
K es un entero, indica el bit que se probará.
Acción: Devuelve el valor '0' o '1' dependiendo del valor del K-ésimo bit del arreglo L.

Función: KONC3
Llamado: KONC3(M)
Argumentos: M es un valor entero (1 ó 2 ó 3).
Acción: Devuelve el valor 1 ó 2 ó 3.
Incrementa M, si M tenía el valor de 3 devuelve 1, si M era 2 devuelve 3, si M era 1 devuelve 2.

Función: IZIGN
Llamado: IZIGN(Z1,Z2)
Argumentos: Z1 y Z2 son valores reales.
Acción: Prueba igualdad de signos de Z1 y Z2.
0 (Falso) si, signo Z1 = signo Z2.
1 (Verdad) si, signo Z1 $\neq$ signo Z2.

Subrutina: KONIT
Llamado: CALL KONIT(I, J, K)
Argumentos: I, J, K Coordenadas del triángulo inicial.
Acción: Define los parámetros del triángulo inicial en la búsqueda de un contorno.

Subrutina: KONSA
Llamado: CALL KONSA
Argumentos: Ninguno.
Acción: Salva los vértices del triángulo inicial.

Subrutina: KONRE
Llamado: CALL KONRE
Argumentos: Ninguno.
Acción: Recupera los vértices del triángulo inicial.

Subrutina: KONNC
Llamado: CALL KONNC
Argumentos: Ninguno.
Acción: Selecciona el vértice V3 para definir el siguiente triángulo a lo largo de un contorno constante. Los vértices V1 y V2 del nuevo triángulo serán vértices del viejo triángulo, mientras que V3 será el nuevo vértice obtenido por reflexión del vértice perdido en el borde opuesto.

Subrutina: KONXV
Llamado: CALL KONXV(I,J)
Argumentos: I,J Números que indican los vértices.
Acción: Intercambia el vértice I con el vértice J, también intercambia sus valores.

Capítulo 5

Superficies

Una superficie es un objeto con tres dimensiones, es decir necesitamos especificar tres números (x, y, z) para representar un punto de una superficie. Una superficie puede representarse algunas veces mediante una ecuación explícita de la forma:

$$z = f(x, y) \qquad (x, y) \; en \; R \qquad \text{R es una región del plano xy}$$

o por medio de una ecuación implícita

$$F(x, y, z) \qquad (x, y, z) \; en \; R \qquad \text{R es una región del espacio}$$

o también por medio de ecuaciones paramétricas

$$x = f(u, v)$$
$$y = g(u, v)$$
$$z = h(u, v)$$

f, g y h son funciones de u y v definidas en alguna región R del plano uv.

Cada una de las ecuaciones tiene sus ventajas y desventajas al emplearlas para representar superficies. En PLOT graficaremos funciones continuas de un solo valor de dos variables reales, es decir, tienen la forma $z = f(x, y)$. También se pueden obtener gráficas de funciones multivaluadas o gráficas de los translapes de dos o más superficies por medio de las subrutinas de superficies dobles o triples.

Para obtener una representación de una superficie se dibujan trazos de la superficie en las direcciones paralelas a los ejes de coordenadas. Además, si se considera que la superficie es opaca y se suprimen las porciones de la malla que quedan ocultas por otras más cercanas al observador, entonces la representación da la sensación de volumen.

En PLOT la representación de superficies en un plano incluye la supresión de líneas ocultas. La rutina principal de PLOT que calcula las regiones visibles de la superficie, considera dos aspectos principales: un orden para el trazo de las curvas y la generación de un horizonte visual. El orden para el trazo de las curvas la da su cercanía al observador. De cada curva, se trazan sólo las porciones que no quedan ocultas por el horizonte visual, y se agregan al horizonte visual para el próximo trazo.

Las rutinas de PLOT que grafican superficies requieren que el usuario proporcione un arreglo matricial $ZE(M, N)$ el cual contiene los valores de las altitudes de la superficie, es decir:

$$ZE(I, J) = Z = f(X_i, Y_j)$$
$$I = 1, ..., M; \qquad J = 1, ..., N; \qquad i = 1, ..., M; \qquad j = 1, ..., N$$

Por lo cual en un renglón de la matriz, el valor de X permanece constante y el valor de Y varía. En una columna de la matriz, el valor de Y es constante y X varía. Otros parámetros que requiere PLOT son, el intervalo de los valores de Z que se desea graficar,número de renglones, número de columnas, sección de los renglones, sección de la columnas y la rutina de conversión de coordenadas.

5.1 Subrutinas que grafican superficies

El usuario generalmente llamará a estas subrutinas.

PLTSE	Vista sureste.
PLTSW	Vista suroeste.
PLTRV	Vista con rotación (alrededor del eje Z).
PLTIV	Vista con inclinación y rotación (con respecto al eje Z).
PLTPV	Vista polar.
PLTSV	Vista esférica.
PLTEV	Vista elíptica.
PLTTV	Vista triangular.
VISDO	Superficies dobles.
VISTR	Superficies triples.

El usuario llamará algunas veces a estas subrutinas. Estas subrutinas realizan el recorrido de la matriz de datos a partir de los puntos más cercanos al observador, generando un arreglo con esos puntos y proporcionándolos a VISHO o VISBO.

VISDS	Secuencia diagonal.
VISRS	Secuencia con rotación.
VISIS	Secuencia inclinada.
VISPS	Secuencia polar.
VISSS	Secuencia esférica.
VISES	Secuencia elíptica.
VISTS	Secuencia triangular.

El usuario no llama a estas subrutinas directamente

VISHO	Define los vectores de trabajo de horizontes y llama a VISBO.
VISBO	Cotas. Traza los puntos de la superficie que caen fuera de los horizontes y actualiza los horizontes. Todas las rutinas que grafican superficies llaman a esta rutina.

5.2 Secuencia de llamado entre las subrutinas de superficies

```
PLTSE -> VISDS -> VISHO -> VISBO -> PL
PLTSW -> VISDS -> VISHO -> VISBO -> PL
PLTRV -> VISRS -> VISHO -> VISBO -> VISLI
                                    PL
PLTIV -> PLTEU
         VISIS -> VISHO -> VISBO -> VISLI
                                    PL
PLTPV -> VISNH
```

```
        -> VISPS  -> VISHO  -> VISBO  -> VISLI
                                         PL

PLTSV   -> VISNP
           PR
           VISNH
           VISSS  -> VISNP
                     PR
                     VISCH  -> VISBO  -> VISLI
                                         PL
PLTEV   -> VISNH
           VISES  -> VISHO  -> VISBO  -> VISLI
                                         PL
PLTTV   -> VISNH
           VISTS  -> VISHO  -> VISBO  -> VISLI
                                         PL
```

5.3 Descripción de las subrutinas

Subrutina: PLTSE
Llamado: CALL PLTSE(Z1, ZE, Z2, NX, NY, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
NX número de renglones de ZE
número de divisiones en X
NY número de columnas de ZE
número de divisiones en Y
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Vista sureste.
Produce una gráfica en perspectiva de una función de valor real definida en coordenadas cartesianas -Z = F(X,Y)-, mostrando los arcos de la superficie que son paralelos a los ejes de coordenadas. Para mayor claridad en la presentación, la figura se corta horizontalmente lo cual da la ilusión de perspectiva lateral y verticalmente para dar la ilusión de profundidad y para mostrar los detalles remotos que de otra manera quedarían ocultos. Se seleccionó la técnica de corte en lugar de rotación porque se desea mantener las líneas horizontales, horizontales.

Subrutina: PLTSW
Llamado: CALL PLTSW(Z1, ZE, Z2, NX, NY, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
NX número de renglones de ZE
número de divisiones en X
NY número de columnas de ZE
número de divisiones en Y
PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Vista suroeste.

Produce una gráfica en perspectiva de una función de valor real definida en coordenadas cartesianas -Z = F(X,Y)-, mostrando los arcos de la superficie que son paralelos a los ejes de coordenadas. Para mayor claridad en la presentación, la figura se corta horizontalmente lo cual da la ilusión de perspectiva lateral y verticalmente para dar la ilusión de profundidad y para mostrar los detalles remotos que de otra manera quedarían ocultos. Se seleccionó la técnica de corte en lugar de rotación porque se desea mantener las líneas horizontales, horizontales.

Subrutina: PLTRV
Llamado: CALL PLTRV(Z1, ZE, Z2, NX, NY, TH, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
NX número de renglones de ZE
número de divisiones en X
NY número de columnas de ZE
número de divisiones en Y
TH ángulo de rotación alrededor del eje Z (en grados). Las rotaciones positivas son en contra de las manecillas del reloj
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Vista con rotación.

Produce una gráfica de una proyección en paralelo de una función de la forma $Z = F(X, Y)$, exhibiendo arcos de la superficie paralelos a los ejes de coordenadas. Para una mejor presentación toda la figura puede rotarse, el ángulo de rotación se especifica en grados en el intervalo -90 $<$ TH $<$ 90.

Subrutina: PLTIV
Llamado: CALL PLTIV(Z1, ZE ,Z2, NX, NY, RO, TI, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
NX número de renglones de ZE
número de divisiones en X
NY número de columnas de ZE
número de divisiones en Y
RO ángulo de rotación (en grados)
TI ángulo de inclinación (en grados)
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Vista con inclinación y rotación.

Produce una proyección en paralelo de una gráfica de una función Z = F(X,Y) mostrando arcos de la superficie paralelos a los ejes de coordenadas. La figura puede inclinarse en dirección al observador y entonces rotarse alrededor del eje Z. La inclinación es positiva cuando es en dirección al observador y negativa cuando se aleja de él. La inclinación es cero cuando se ve la figura por la parte superior y de noventa grados cuando se ve desde el piso. El ángulo de rotación es cero cuando el eje Y se aleja del observador y es positivo cuando el eje X positivo se mueve en dirección al observador.

Subrutina: PLTPV
Llamado: CALL PLTPV(Z1, ZE, Z2, NR, NP, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(R,P), la dimensión es ZE(NR,NP).
Z1, Z2 intervalo del rango de ZE que se desea graficar.
NR número de renglones
número de divisiones de un radio.
NP número de columnas
número de divisiones del ángulo polar, debe ser un entero de la forma $4 \times n + 1$ para asegurar que los cuatro cuadrantes de la figura se unan correctamente.
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Vista Polar.
Produce una vista ortográfica de una función de un solo valor definida y evaluada en coordenadas polares, mostrando los arcos radiales y circunferenciales. El eje Z se proyecta en el eje vertical del área de trazo, con una leve inclinación. El eje X se proyecta en el eje horizontal. El ángulo polar P se mide alrededor del eje Z en sentido contrario a las manecillas del reloj desde el eje X así, en el trazo $P = 0$ queda a la derecha, $P = 180$ queda en el extremo izquierdo y $P = 270$ queda en la parte central del frente del trazo. Para que la figura se una adecuadamente se debe cumplir que ZE(*,1) = ZE(*,NP).

Subrutina: PLTSV
Llamado: CALL PLTSV(ZE, NP, NT, S, O, PR, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(P,T)
NP número de renglones de ZE.
número de divisiones de la latitud, debe ser de la forma $2 \times n$.
NT número de columnas de ZE.
número de divisiones de la longitud.
S S = 1 gráfica de ZE positiva.
S = -1 gráfica de ZE negativa.
O(3,3) Matriz ortogonal de rotación.
PR subrutina de proyección de coordenadas generalmente VISSP(rho,phi,r,t,p,o).
PL subrutina de movimiento de la pluma (conversión de coordenadas), generalmente PLTPO.
Acción: Produce una gráfica de la proyección en paralelo de una función de un solo valor definida sobre una superficie esférica de forma tal que muestra los arcos de latitud y longitud.

Subrutina: PLTEV
Llamado: CALL PLTEV(Z1, ZE, Z2, NX, NE, PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(XI,ETA)
Z1, Z2 intervalo del rango de ZE que se desea graficar
NX número de renglones de ZE
número de divisiones de XI
NE número de columnas de ZE
número de divisiones de ETA debe ser de la forma $4 \times n + 1$
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Produce una gráfica en perspetiva de una función de un solo valor definida en coordenadas elípticas de forma tal que muestra los arcos de XI y ETA constantes.

Subrutina: PLTTV
Llamado: CALL PLTTV(Z1, ZE, Z2, N, M, PL)
Argumentos: ZE(M,M) arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
N número actual de las columnas
M número de columnas de ZE
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Vista triangular.
Produce una gráfica en perspectiva de una función definida en el triángulo unitario en términos de coordenadas homogeneas. Los valores de la función ocupan la porción triangular inferior de la matriz cuadrada ZE, de la cual sólo el fragmento ZE(N,N) se trazará. Al generar la gráfica los valores de ZE se escalarán al rango Z1, Z2.

Subrutina: VISDS
Llamado: CALL VISDS(Z1,ZE,Z2,J1,J2,MX,I1,I2,MY,US,VS,L,M,PL)
Argumentos: ZE arreglo matricial que contiene los valores que generan la superficie, estos valores deben tener la forma Z = F(X,Y)
Z1, Z2 intervalo del rango de ZE que se desea graficar
MX número de renglones de ZE
MY número de columnas de ZE
J1, J2 Renglón inicial y renglón final, rango de X (coordenada horizontal)
I1, I2 Columna inicial y columna final, rango de Y (coordenada de profundidad)
US,VS Proporción del corte, para dar la ilusión de vista lateral y vista con profundidad.

$$0.0 < \mathrm{US,VS} < 1.0$$

L L = 1 Vista oeste
L = -1 Vista este
M M = 1 Vista Sur
M = -1 Vista Norte
PL subrutina de movimiento de la pluma (conversión de coordenadas).
Acción: Secuencia Diagonal.
Realiza el recorrido diagonal de una matriz para las rutinas de superficies con líneas ocultas.

Subrutina: VISDO
Llamado: CALL VISDO(Z1,S1,S2,Z2,NX,MX,NY,MY,US,VS,VD,L,IS,PL)
Argumentos: S1,S2 Arreglos matriciales que contienen los valores que generan las superficies, estos valores deben tener la forma Z = F(X,Y)

Z1, Z2 intervalo del rango de S1 y S2 que se desea graficar

MX número de renglones de S1 y S2
número de divisiones de X

MY número de columnas de S1 y S2
número de divisiones de Y

NX, NY Sección de S1 y S2 que se graficará, es decir, desde 1 hasta NX y desde 1 hasta NY.

US,VS Proporción del corte, para dar la ilusión de vista lateral y vista con profundidad.

$$0.0 < US,VS < 1.0$$

VD Desplazamiento vertical entre los segmentos de la función.

L L = 1 Vista oeste
L = -1 Vista este

IS IS = -1 Superficies juntas
IS = 1 Superficies separadas

PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Realiza una gráfica con líneas ocultas de dos superficies que posiblemente se intersectan, ocultando aquellas partes de las superficies que son cubiertas por las otras. Permite un desplazamiento vertical para mostrar con mayor claridad los detalles de las intersecciones.

Subrutina: VISTR
Llamado: CALL VISTR(Z1,S1,S2,S3,Z2,NX,MX,NY,MY,US,VS,VD,L,IS,PL)
Argumentos: S1, S2, S3 Arreglos matriciales que contienen los valores que generan las superficies, estos valores deben tener la forma Z = F(X,Y)

Z1, Z2 intervalo del rango de S1, S2 y S3 que se desea graficar

MX número de renglones de S1 y S2
número de divisiones de X

MY número de columnas de S1 y S2
número de divisiones de Y

NX, NY Sección de S1, S2 y S3 que se graficará, es decir, desde 1 hasta NX y desde 1 hasta NY.

US, VS Proporción del corte, para dar la ilusión de vista lateral y vista con profundidad.

$$0.0 < US,VS < 1.0$$

VD Desplazamiento vertical entre los segmentos de la función.

L L = 1 Vista oeste
L = -1 Vista este

IS IS = -1 Superficies juntas
IS = 1 Superficies separadas

PL subrutina de movimiento de la pluma (conversión de coordenadas).

Acción: Realiza una gráfica con líneas ocultas de tres superficies que posiblemente se intersectan, ocultando aquellas partes de las superficies que son cubiertas por las otras. Permite un desplazamiento vertical para mostrar con mayor claridad los detatalles de las intersecciones.

Capítulo 6

Programas de demostración

- Los programas de demostración están escritos en el lenguaje 'C'.
- Los programas muestran la forma de llamar a las funciones de PLOT95c. Primero se muestra el programa y después la gráfica que genera.
- El llamado a las subrutinas de PLOT84 es muy parecido ya que se tienen los mismos argumentos en la misma posición.

6.1 Trazo de curvas planas

Aquí se muestra como generar los datos y el orden de llamado a las funciones que trazan curvas planas.

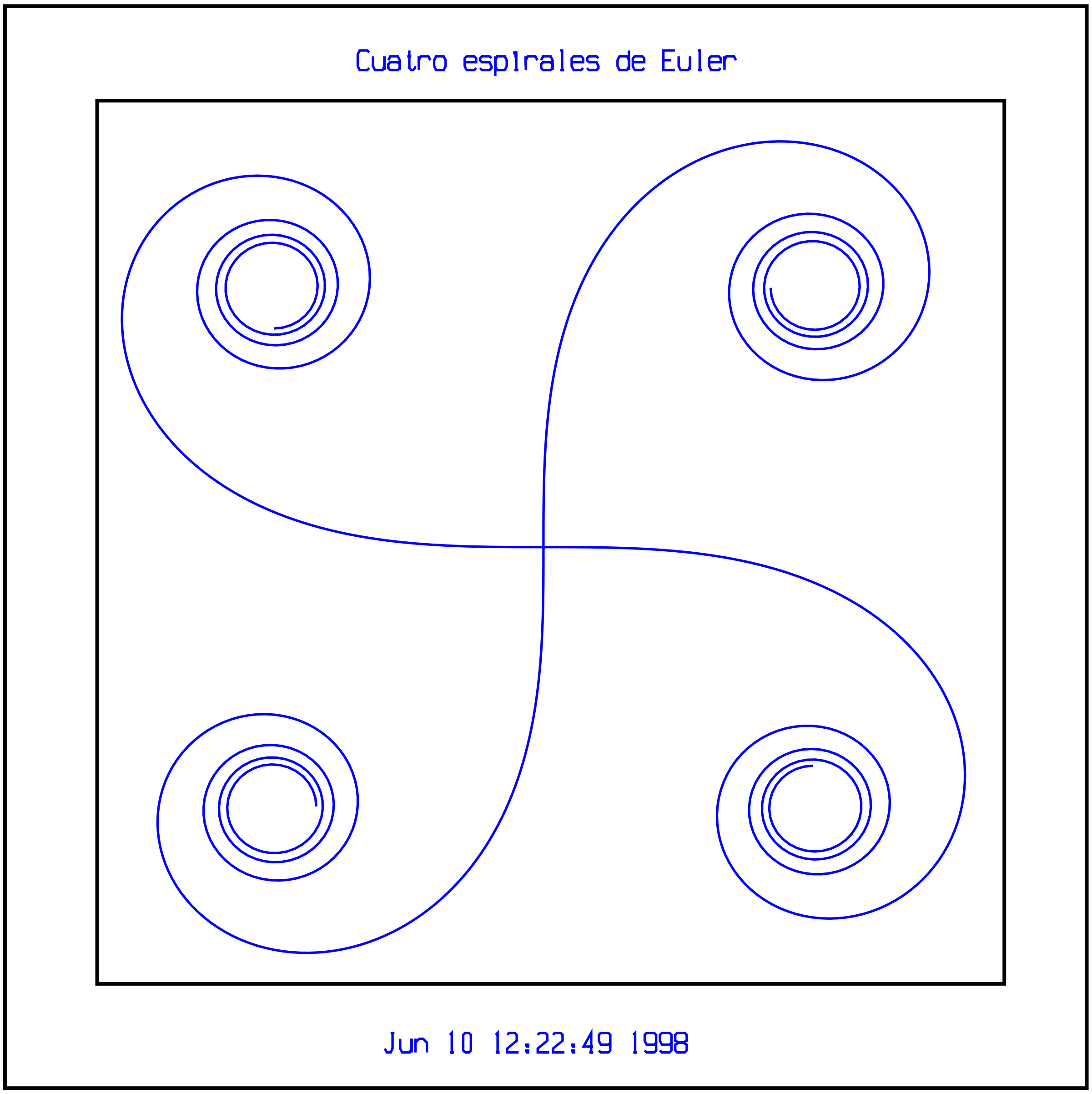
Secuencia de llamado:

> Generar los datos en arreglos vectoriales.
> o
> Obtener los datos desde un archivo.
>
> plt00();
> pltfr(y);
>
> Generar la curva o las curvas con pltga, pltig o pltfi.
>
> pltho();
> pltclose();

los primeros dos llamados inician los parámetros de PLOT y los dos últimos desalojan el área de trazo y cierran los archivos pertinentes.

```
/*   Graficas Planas  (PLTIG)    Espiral de Euler
                [J.P.H.E   JUN-1993]
*/

#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "plotp.h"

#define  NP          250
double   vx[NP],     vy[NP];  /* Arreglos para los valores de X y Y */

double fux(double x)          /* Valor de X */
{
  x  = ( x*x )*1.570797;
  return sin(x);
}

double fuy(double y)          /* Valor de Y */
{
  y  = (y*y)*1.570795;
  return cos(y);
}

/*    Calcula la integral por el metodo de Simpson      */

double sim(double a, double b, int m, double (*fu)() )
{
  double  h, x,  xi0, xi1, xi2;
  int     i, ia, np;

  h   = (b-a)/(2.0 * (double) m);
  xi0 = fu(a) + fu(b);
  xi1 = 0.0;
  xi2 = 0.0;
  np  = 2*m - 1;
  for( i  = 1; i <= np; i++) {
       x  = a + ((double) i * h);
       ia = i/2;
       if( (ia*2-i) == 0 ) { xi2 = xi2 + fu(x); continue; }
       xi1 = xi1 + fu(x);
  }
  return h*(xi0 + 2.0*xi2 + 4.0*xi1)/3.0;
}


main()
{  double  xmax = 0.0, t = 0.0, dt;
   double  ymax = 0.0, xx, yy;
   int     i;
   char    in[50];
   char    bu[60] = {"Cuatro espirales de Euler"};

   dt    = 4.0/(double) (NP-1);
```

```
    for( i = 0; i < NP; i++) {      /* Calculo de los valores de la funcion */
        xx = sim(0.0, t, 30, fux);
        yy = sim(0.0, t, 30, fuy);
        vx[i] = xx;
        vy[i] = yy;
        xmax = (xmax < xx) ? xx : xmax;
        ymax = (ymax < xx) ? yy : ymax;
        t  +=  dt;
    }
    xx = xmax/10.0;      yy = ymax/10.0;
    xx = xmax + xx;      yy = ymax + yy;

    plt00();                                                /* Iniciar PLOT */
    plotz("P1 ", 3);                                /* Comandos del DMP - 29 */
    pltfr(3);                                                      /* Marco */
    pltso(3.0, 3.0, 0.0, 0.0);                          /* Escalas y origen */
    pltig(-xx, -yy, 1, pltca);                    /* Valores minimos en X y Y */
    pltig( xx,  yy, 2, pltca);                    /* Valores maximos en X y Y */

    pltig(vx[0], vy[0], 3, pltca);                           /* Punto inicial */
    for( i = 1; i<NP; i++) pltig(vx[i], vy[i], 4, pltca); /* Punto siguien. */

    pltig(-vx[0], -vy[0], 3, pltca);
    for( i = 1; i < NP; i++) pltig(-vx[i], -vy[i], 4, pltca);

    pltig(-vx[0], vy[0], 3, pltca);
    for( i = 1; i < NP; i++) pltig(-vy[i], vx[i], 4, pltca);

    pltig(vx[0], -vy[0], 3, pltca);
    for( i = 1; i < NP; i++) pltig(vy[i], -vx[i], 4, pltca);

    pltlla(bu, 25, in, 0, 3);                                   /* Letreros */
    pltho();                                                   /* Desalojar */
    pltclose();                          /* Cerrar archivo de movimientos */
}
```

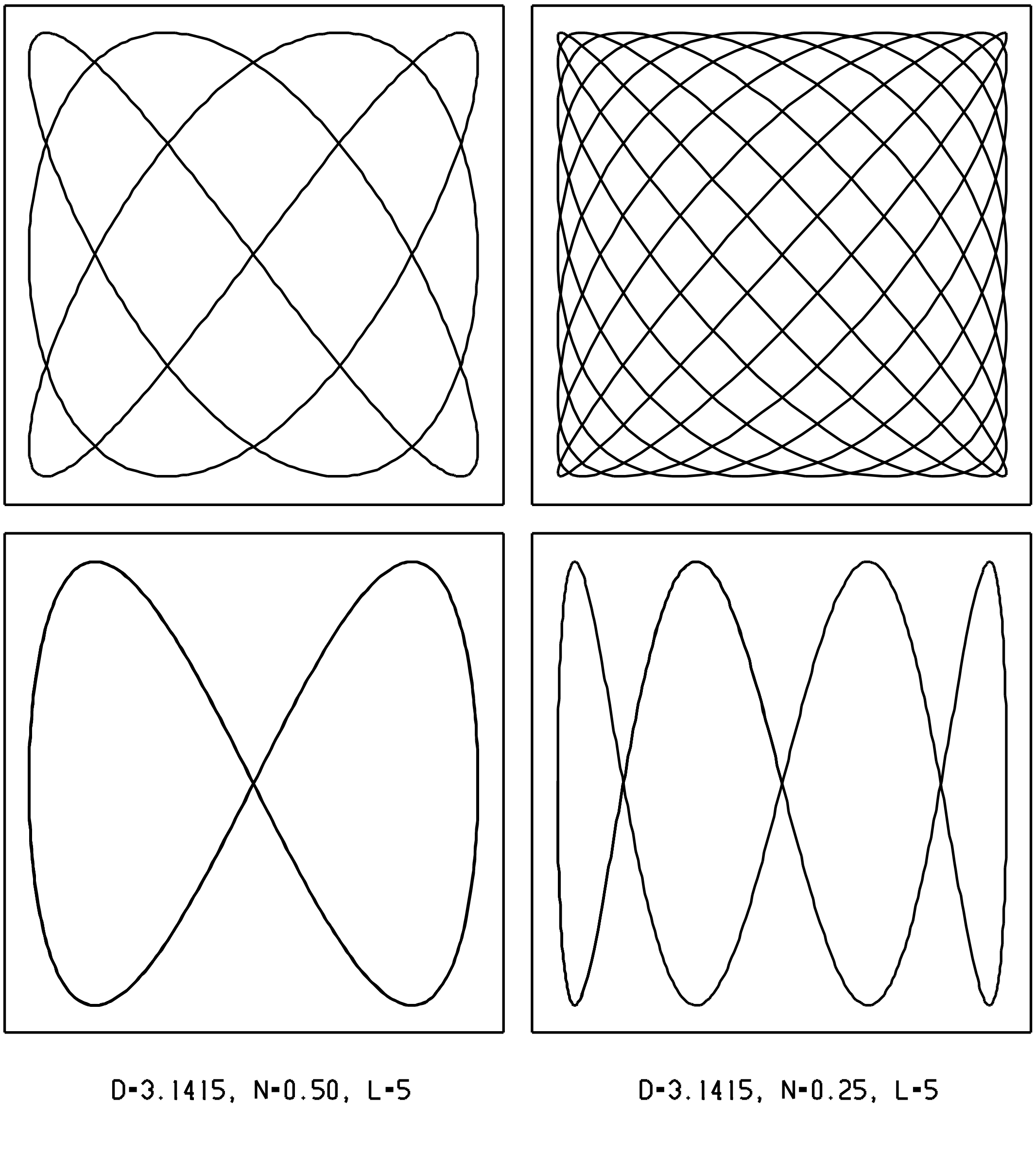


Cuatro espirales de Euler
Jun 10 12:22:49 1998

```
/*            CURVA BOWDITCH O CURVA DE LISSAJOUS
     La ecuacion es:      X = 9.0*SIN(N*T+D),   Y = 8.0*SIN(T)
                   [J.P.H.E     JUN-1993]                   */
#include  <stdio.h>
#include  <math.h>
#include  "plotp.h"
double VX[750], VY[750], T, DT;

void lis(double D, double N, double L)
{    int    I;
     DT = (L*6.24318)/744.0;
       for(I = 0, T=0.0; I < 750; I++, T += DT) {
           VX[I] = 9.0*sin( N*T+D );
           VY[I] = 8.0*sin(T);           }
}
main()
{   char  bu0[]={"Curva Bowditch o de Lissajous, X=9*SEN(N*T+D), Y=8*SEN(T)"};
    char  bu1[]={"D=3.1415, N=0.75, L=4"};
    char  bu2[]={"D=3.1415, N=0.90, L=10"};
    char  bu3[]={"D=3.1415, N=0.50, L=5"};
    char  bu4[]={"D=3.1415, N=0.25, L=5"};
    plt00();
    pltfr(1);
    plot( 3.6, 0.1, UP);                             /* Marco Superior D */
    plot( 0.1, 0.1, DOWN);    plot( 0.1, 3.6, DOWN);
    plot( 3.6, 3.6, DOWN);    plot( 3.6, 0.1, DOWN);
    plot(-0.1, 0.1, UP);                             /* Marco Superior I */
    plot(-3.6, 0.1, DOWN);    plot(-3.6, 3.6, DOWN);
    plot(-0.1, 3.6, DOWN);    plot(-0.1, 0.1, DOWN);
    plot( 3.6, -0.1, UP);                            /* Marco Inferior  D */
    plot( 0.1, -0.1, DOWN);    plot( 0.1, -3.6, DOWN);
    plot( 3.6, -3.6, DOWN);    plot( 3.6, -0.1, DOWN);
    plot(-0.1, -0.1, UP);                            /* Marco Inferior  I */
    plot(-3.6, -0.1, DOWN);    plot(-3.6, -3.6, DOWN);
    plot(-0.1, -3.6, DOWN);    plot(-0.1, -0.1, DOWN);
    pltso(1.75, 1.75, 1.85,  1.85);
    lis(3.1415, 0.90, 10);
    pltga(-10.0, VX, 10.0, -9.0, VY, 9.0, 750, pltca);     plotr();
    pltso(1.75, 1.75, -1.85,  1.85);
    lis(3.1415, 0.75, 4);
    pltga(-10.0, VX, 10.0, -9.0, VY, 9.0, 750, pltca);     plotr();
    pltso(1.75, 1.75, -1.85,  -1.85);
    lis(3.1415, 0.50, 5);
    pltga(-10.0, VX, 10.0, -9.0, VY, 9.0, 750, pltca);     plotr();
    pltso(1.75, 1.75, 1.85,  -1.85);
    lis(3.1415, 0.25, 5.0);
    pltga(-10.0, VX, 10.0, -9.0, VY, 9.0, 750, pltca);     plotr();
    plttp(-2.8,  4.0, -0.1, bu1, 0.0, (sizeof(bu1)-1), 1, pltmv);
    plttp( 0.8,  4.0, -0.1, bu2, 0.0, (sizeof(bu2)-1), 1, pltmv);
    plttp(-2.8, -4.0, -0.1, bu3, 0.0, (sizeof(bu3)-1), 1, pltmv);
    plttp( 0.9, -4.0, -0.1, bu4, 0.0, (sizeof(bu4)-1), 1, pltmv);
    pltlla(bu0, (sizeof(bu0)-1), "", 0, 1);
    pltho();                    pltclose();
}
```

Curva Bowditch o de Lissajous, X=9*SEN(N*T+D), Y=8*SEN(T)

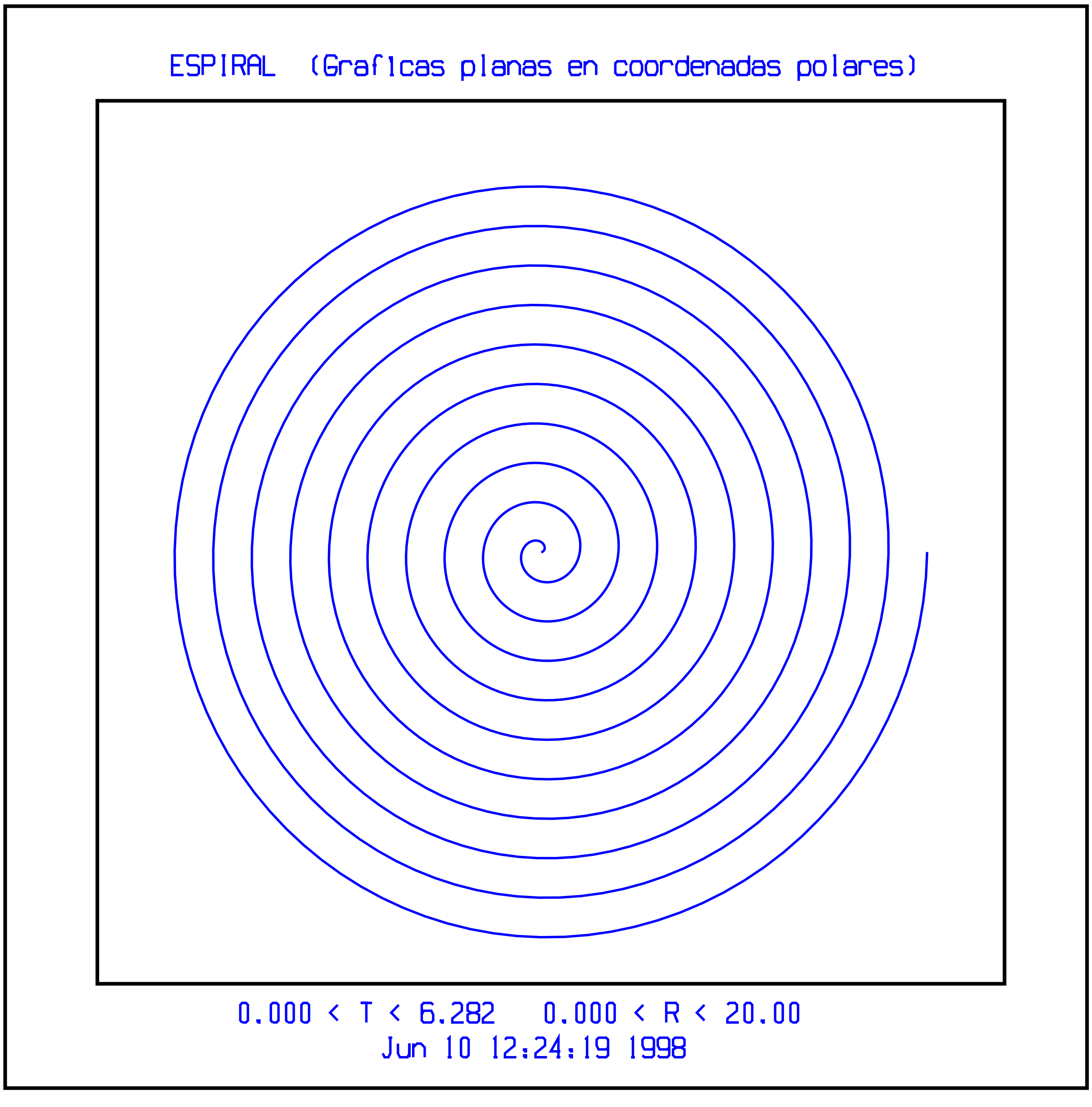
D=3.1415, N=0.75, L=4 D=3.1415, N=0.90, L=10



D=3.1415, N=0.50, L=5 D=3.1415, N=0.25, L=5

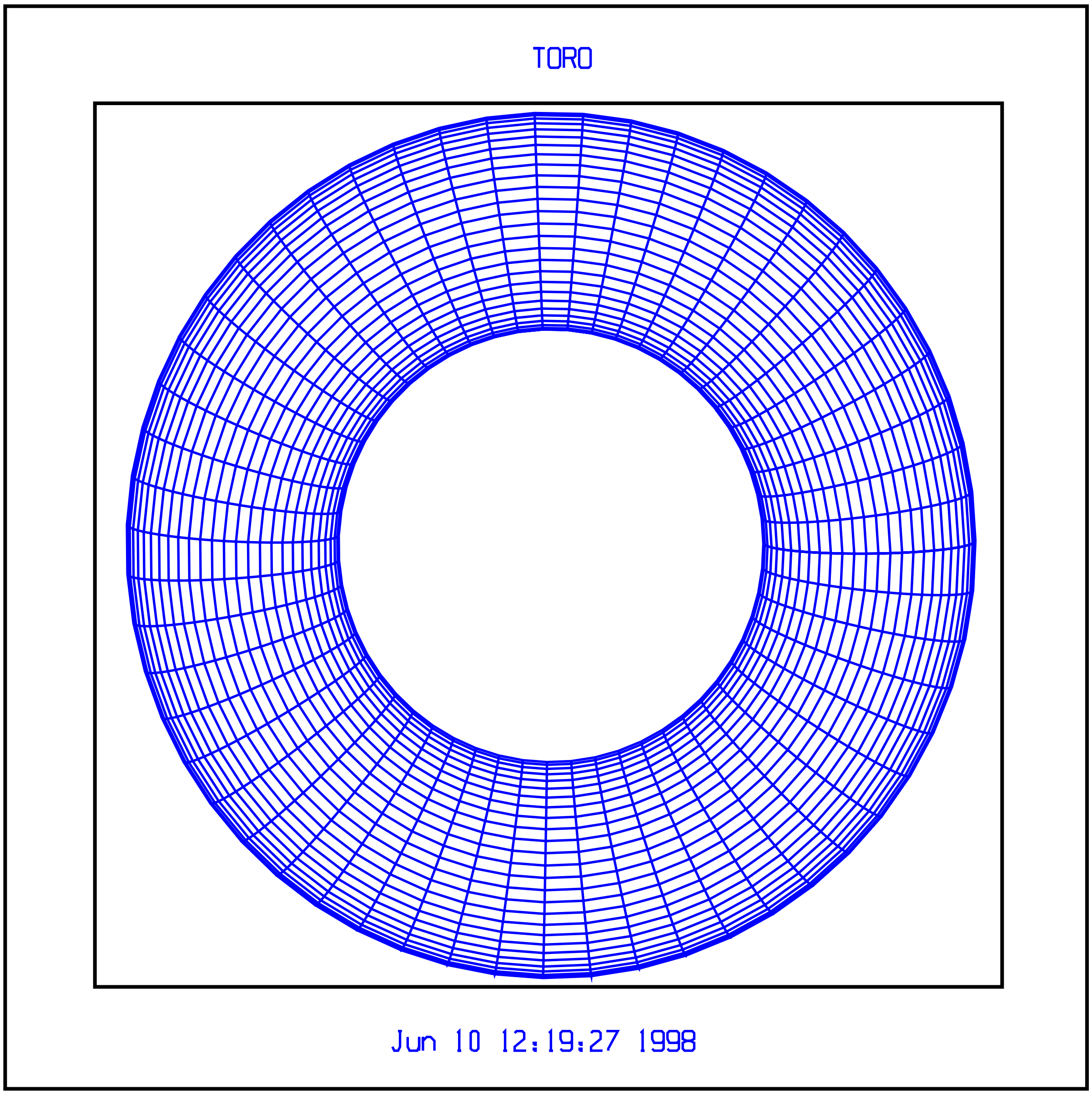
Jun 21 04:03:01 1998

```c
/*  ESPIRAL (Graficas planas en coordenadas polares)
                [J.P.H.E   JUN-1993]
*/

#include <stdio.h>
#include <math.h>
#include "plotp.h"

#define  NP   1000        /* Numero de puntos que generan la espiral */
#define  NG   10          /* Numero de giros de la espiral           */

double te[NP], ro[NP];

main()
{
   double tmi =  0.0, tma = 6.2815,   t, dt;
   double rmi =  0.0, rma = 20.0,     r, dr;
   int    i;
   char   in[50];
   char   bu[60] = {"ESPIRAL  (Graficas planas en coordenadas polares)"};

   dt  = fabs(tma-tmi)/(double) (NP-1);
   dr  = fabs(rma-rmi)/(double) (NP-1);

   t   = tmi;                 /* Calculo de los valores de la funcion */
   r   = rmi;
   for(i = 0; i < NP; i++) {
       te[i]  =  t;    t += dt;
       ro[i]  =  r;    r += dr;
   }
   dr = fabs(rma-rmi)/10.0;
   r  = rma + dr;

   plt00();
   plotz("P1 V1 ", 6);
   pltfr(6);
   pltso(3.1, 3.1, 0.0, 0.0);

   pltga(tmi, &te[0], tma/(double) NG, rmi, &ro[0], r, NP, pltpo);

   i = pltgi(in, tmi, 'T', tma, rmi, 'R', rma);
   pltlla(bu, 49, in, i, 6);
   pltho();
   pltclose();
}
```


ESPIRAL (Graficas planas en coordenadas polares)
0.000 < T < 6.282 0.000 < R < 20.00
Jun 10 12:24:19 1998

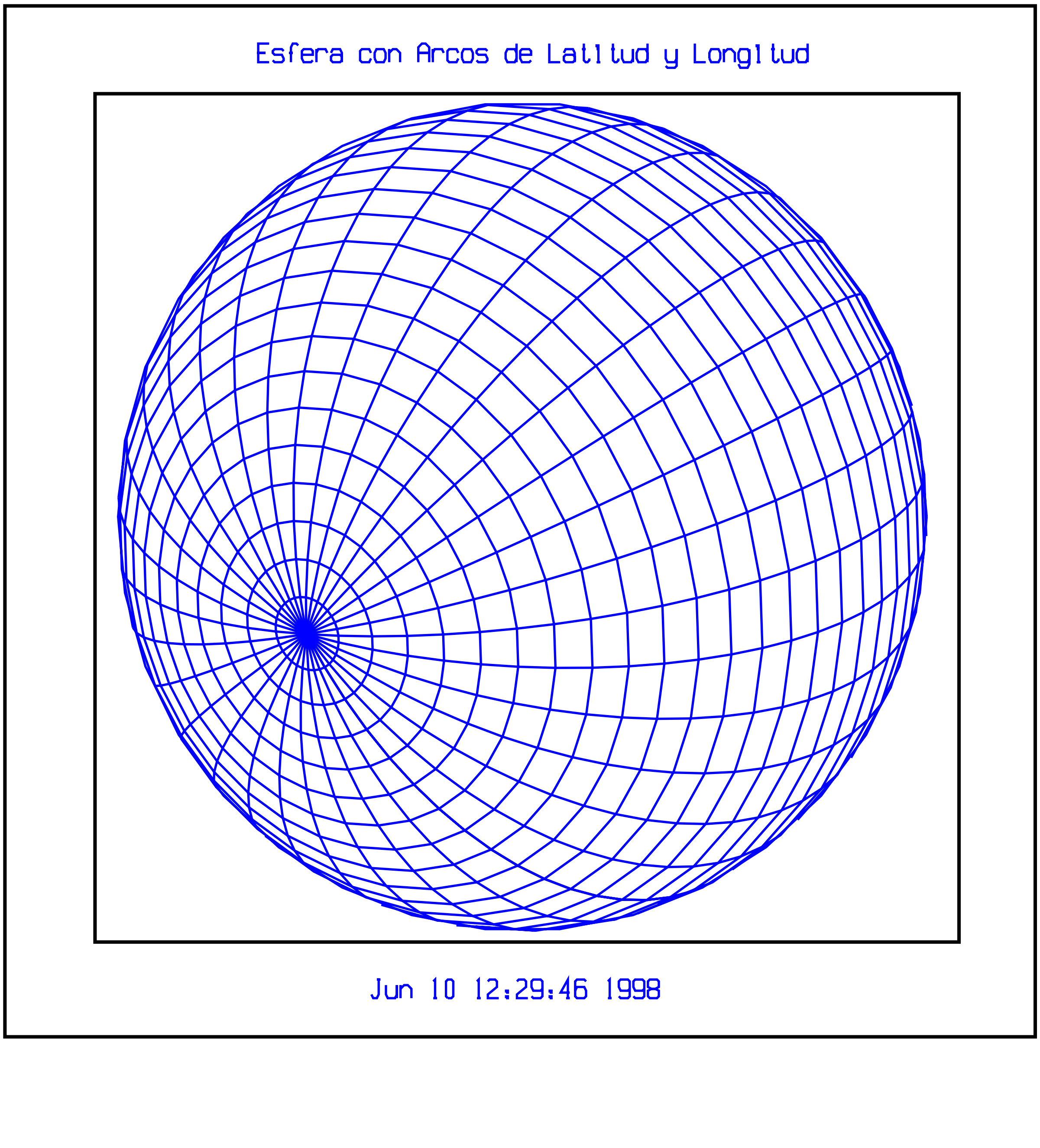
```
/*     GRAFICAS PLANAS  (PLTIG)   TORO
        [J.P.H.E           JULIO-1993]
*/
#include <math.h>
#include <stdio.h>
#include "plotp.h"

#define  npu  56
double   O1[3][3];           /* Matriz de rotacion de Euler  */

                             /* Funcion de movimiento de la pluma */
void d19to( PH, TH, P)       /* Proyeccion de las coordenadas     */
double PH,TH;                /* toroidales al plano polar         */
int    P;
{   double the,phi,x,y,z,u,v,w,ro,fi;
    the = 6.28318 * TH;
    phi = 6.28318 * PH;
    x   = 0.25*((3.0+cos(the)) * cos(phi));
    y   = 0.25*((3.0+cos(the)) * sin(phi));
    z   = 0.25*sin(the);
    u   = O1[0][0]*x + O1[0][1]*y + O1[0][2]*z;
    v   = O1[1][0]*x + O1[1][1]*y + O1[1][2]*z;
    w   = O1[2][0]*x + O1[2][1]*y + O1[2][2]*z;
    ro  = sqrt(u*u+v*v);
    fi  = atan2(v,u)/6.28318;
    pltpo(fi, ro, P );
}

main()
{       double  phmi  = 0.0;
        double  phma  = 2.0*3.14159;
        double  thmi  = 0.0;
        double  thma  = 2.0*3.14159;
        double  dph   = (phma-phmi)/(double) (npu-1);
        double  dth   = (thma-thmi)/(double) (npu-1);
        double  ph, th;
        int     i,  j,  anpu = npu/2;
        char    in[2],   bu[]={"TORO"};

        plteu(&O1[0][0], -30.0, 30.0, -10.0);

        plt00();
        plotz("P1 V1 ",6);
        pltfr(6);
        pltso(3.0,  3.0,  0.0, 0.0);
        pltig(phmi, thmi, 1, d19to);
        pltig(phma, thma, 2, d19to);

        th   = thmi;
        for(i=1; i <= anpu; i++) {
            ph  =  phmi;
            pltig(ph,th, 3, d19to);
            for(j=1; j < npu; j++) {
                ph  =  ph + dph;
```

```
                pltig(ph, th, 4, d19to);
            }
              th = th + dth;
        }

        ph   = phmi;
        for(i=1; i <= npu; i++)  {
            th  = thmi;
            pltig(ph,th, 3, d19to);
            for(j=1; j < anpu; j++)  {
                th = th + dth;
                pltig(ph, th, 4, d19to);
            }
            ph = ph + dph;
        }

        pltlla(bu, (sizeof(bu)-1), in, 0, 6);
        pltho();
        pltclose();
}
```


TORO
Jun 10 12:19:27 1998

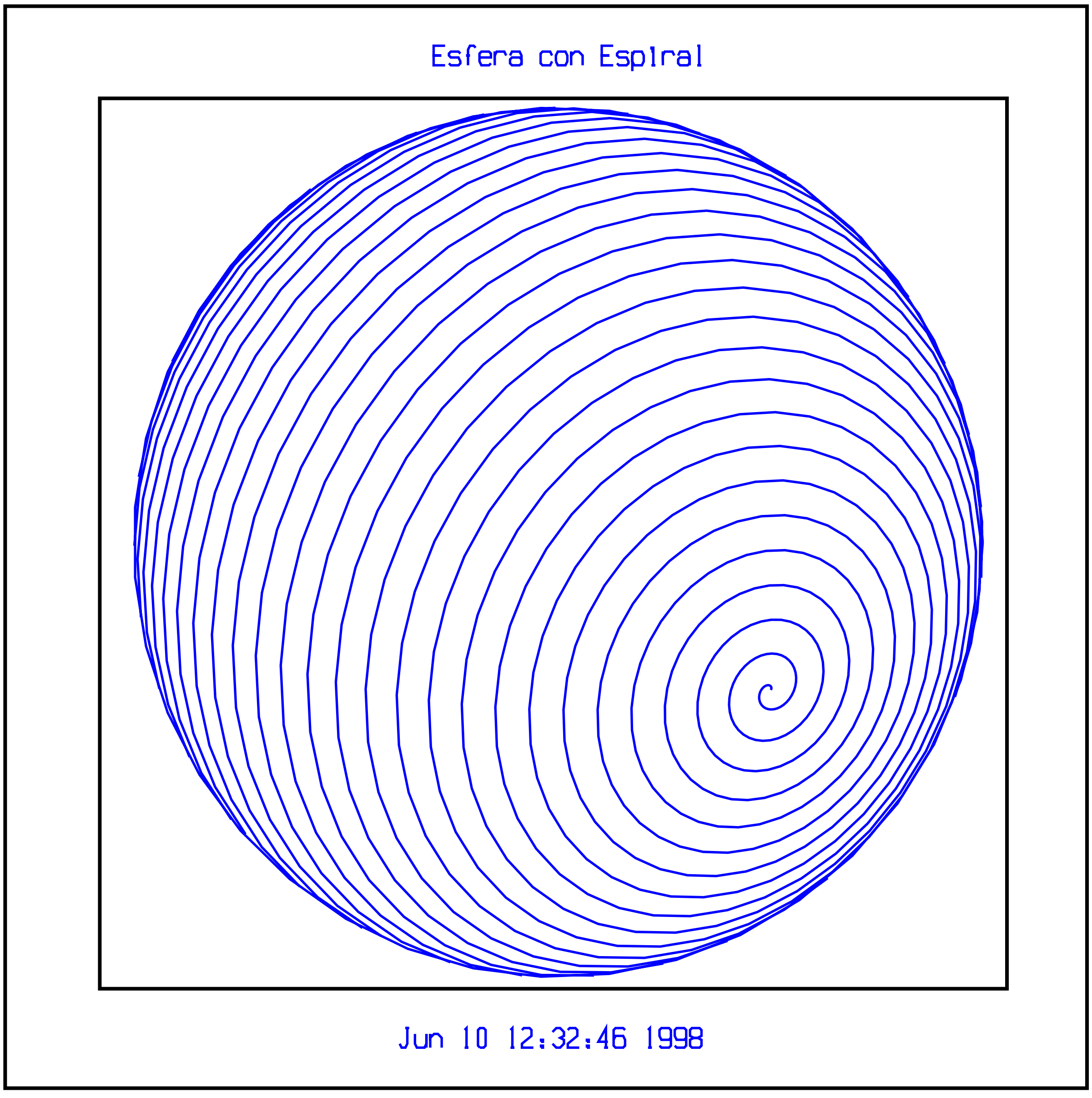
```
/*      ESFERA CON ARCOS DE LATITUD Y LONGITUD
                [J.P.H.E.   JULIO-1992]
*/
#include <math.h>
#include <stdio.h>
#include "plotp.h"
#define  npu  35

double O1[3][3];                        /* Matriz de rotacion de Euler  */

void d19sp( PH, TH, P)              /* Funcion de movimiento de la pluma */
double PH,TH;
int    P;
{   double the,phi,x,y,z,u,v,w,ro,fi;

    the = 3.14159 * TH;
    phi = 6.28318 * PH;
    x   = sin(the) * cos(phi);
    y   = sin(the) * sin(phi);
    z   = cos(the);
    u   = O1[0][0]*x + O1[0][1]*y + O1[0][2]*z;
    v   = O1[1][0]*x + O1[1][1]*y + O1[1][2]*z;
    w   = O1[2][0]*x + O1[2][1]*y + O1[2][2]*z;
    ro  = sqrt(u*u+v*v);
    fi  = atan2(v,u)/6.28318;
    pltpo(fi, ro, (int)(P && (w >= 0.0)) );  /* Proyeccion al plano polar */
}

main()
{       int i, j;
        double  phmi  = 0.0;
        double  phma  = 2.0*3.14159;
        double  thmi  = 0.0;
        double  thma  = 3.14159;
        double  dph   = (phma-phmi)/(double) (npu-1);
        double  dth   = (thma-thmi)/(double) (npu-1);
        double  ph, th;

        plteu(&O1[0][0], -10.0, 30.0, 70.0);

        plt00();
        plotz("V1 ",3);
        pltfr(4);
        pltso(3.10, 3.10, 0.0, 0.0);
        pltig(phmi, thmi, 1, d19sp);
        pltig(phma, thma, 2, d19sp);

        th   = thmi;                    /* Traza arcos de latitud */
        for(i=0; i < npu; i++) {
            ph  =  phmi;
            pltig(ph,th, 3, d19sp);
            for(j=1; j < npu; j++) {
                ph  =  ph + dph;
                pltig(ph, th, 4, d19sp);
```

```
           }
             th = th + dth;
       }

       ph   = phmi;                   /* Traza arcos de longitud */
       for(i=0; i < npu; i++)  {
           th  = thmi;
           pltig(ph,th, 3, d19sp);
           for(j=1; j < npu; j++)  {
               th = th + dth;
               pltig(ph, th, 4, d19sp);
           }
           ph = ph + dph;
       }

      ph  = phmi;           /* Enmarcar la esfera con una circunferencia */
      pltig(ph, 3.14159, 3, pltpo);
      for(j=1; j < npu; j++) {
          ph   = ph + dph;
          pltig(ph, 3.14159, 4, pltpo);
      }
      pltlla("Esfera con Arcos de Latitud y Longitud", 38, " ",0, 4);
      pltho();
      pltclose();
}
```


Esfera con Arcos de Latitud y Longitud
Jun 10 12:29:46 1998

```
/*                ESFERA CON ESPIRAL
                [J.P.H.E.   MARZO-1993]
*/
#include <math.h>
#include <stdio.h>
#include "plotp.h"
#define  npu    40

double O1[3][3];                          /* Matriz de rotacion de Euler */

void d19sp( PH, TH, P)              /* Funcion de movimiento de la pluma */
double PH,TH;
int    P;
{
    double the,phi,x,y,z,u,v,w,ro,fi;

    the = 3.14159 * TH;
    phi = 6.28318 * PH;
    x   = sin(the) * cos(phi);
    y   = sin(the) * sin(phi);
    z   = cos(the);
    u   = O1[0][0]*x + O1[0][1]*y + O1[0][2]*z;
    v   = O1[1][0]*x + O1[1][1]*y + O1[1][2]*z;
    w   = O1[2][0]*x + O1[2][1]*y + O1[2][2]*z;
    ro  = sqrt(u*u+v*v);
    fi  = atan2(v,u)/6.28318;
    pltpo(fi, ro, (int)(P && (w >= 0.0)) );  /* Proyeccion al plano polar */
}


main()
{
        int     i, j;
        double  phmi  = 0.0;
        double  phma  = 2.0*3.14159;
        double  thmi  = 0.0;
        double  thma  = 3.14159;
        double  dph   = (phma-phmi)/(double) (npu-1);
        double  dth   = (thma-thmi)/(double) (npu-1);
        double  dt    = dth/(double) (npu-1);
        double  ph, th, t;

        plteu(&O1[0][0], -10.0, 30.0, 70.0);

        plt00();
        plotz("V1 ",3);
        pltfr(4);
        pltso(3.10, 3.10, 0.0, 0.0);
        pltig(phmi, thmi, 1, d19sp);
        pltig(phma, thma, 2, d19sp);

        th   = thmi;
        for(i=0; i < npu; i++) {
            ph  =  phmi; t = th;
```

```
            pltig(ph,th, 3, d19sp);
            for(j=1; j < npu; j++) {
                ph  =  ph + dph; t = t + dt;
                pltig(ph, t, 4, d19sp);
            }
              th = th + dth;
        }

       ph  = phmi;           /* Enmarcar la esfera con una circunferencia */
       pltig(ph, 3.14159, 3, pltpo);
       for(j=1; j < npu; j++) {
           ph   = ph + dph;
           pltig(ph, 3.14159, 4, pltpo);
       }
       pltlla("Esfera con Espiral", 18, " ", 0, 4);
       pltho();
       pltclose();
}
```


Esfera con Espiral
Jun 10 12:32:46 1998

6.2 Trazo de superficies y contornos

Aquí se muestra como generar los datos y el orden de llamado a las funciones que trazan superficies.
La secuencia de llamado típica es:

Generar los datos en un arreglo matricial.
o
Obtener los datos desde un archivo y depositarlos en un arreglo matricial.

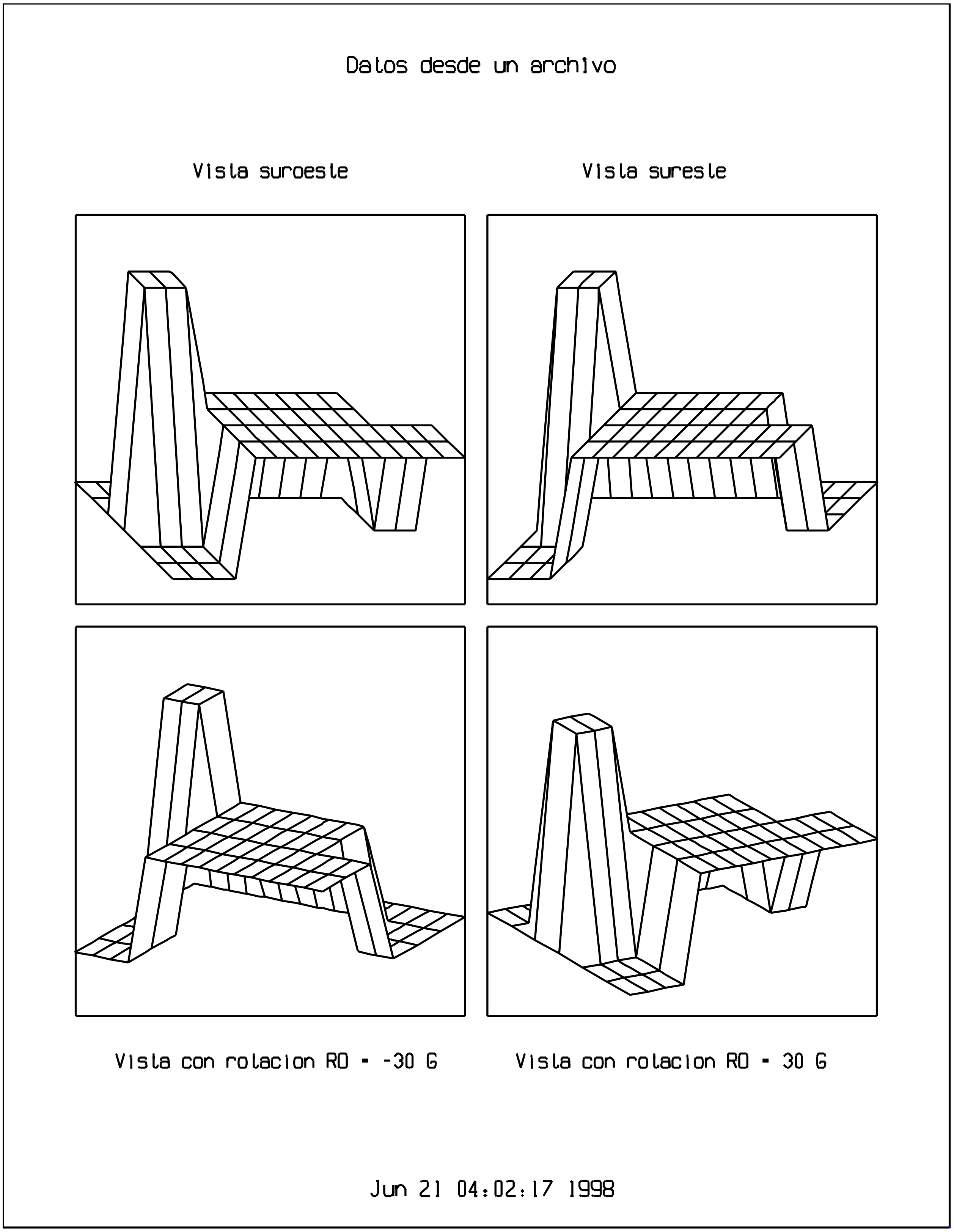
plt00();
pltfr(y);

Generar la superficie con pltrv, pltiv, pltpv, visds, visdo o vistr.
o
Generar los contornos con pltkp, pltkx, pltky, pltor.

pltho();
pltclose();

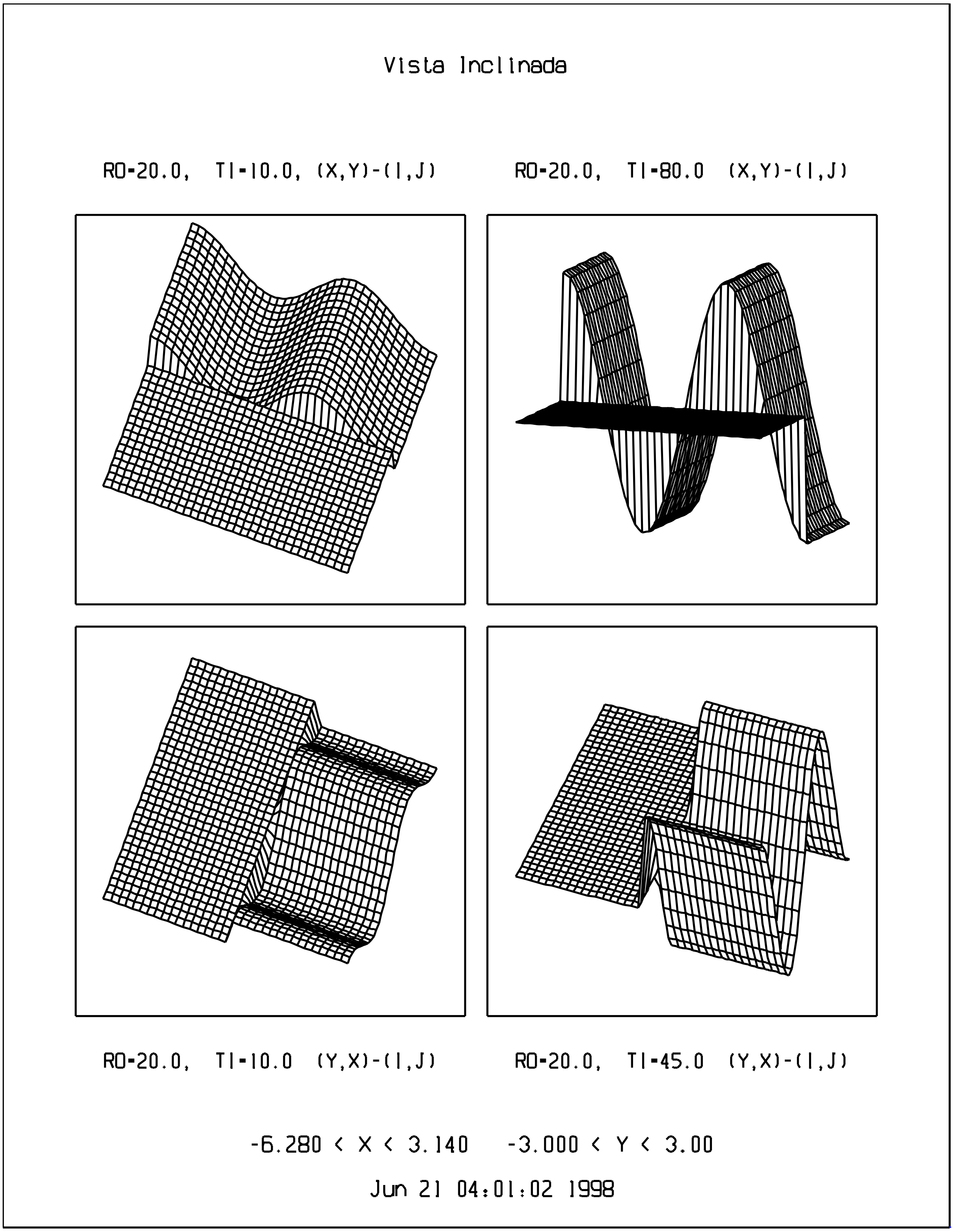
```
/*      SUPERFICIE DESDE DATOS EN UN ARCHIVO
          [J.P.H.E         JUNIO-1993]        */
#include <stdio.h>
#include <math.h>
#include "plotp.h"
#define   NR   15            /* Dimension de la matriz y de los datos */
#define   NC   7
double    m[NR][NC];       /* Matriz en la que se almacenan los datos */

/*        Leer valores de la matriz.  Devuelve el valor min.
          y el valor max. de los valores del archivo.                */
void lvm(double *mmi, double *mma, int nr, int nc, FILE *fp)
{   int    i, j;
    double vr, vmi, vma;
    for( i=0; i < nr; i++ )
         for( j=0; j < nc; j++ ) {
              fscanf(fp,"%lf", &vr);
              m[i][j] = vr;
         }
    vmi = m[0][0];        vma = vmi;
    for( i=0; i < nr; i++ )
         for( j=0; j < nc; j++ ) {
             vmi = (vmi > m[i][j]) ? m[i][j] : vmi;
             vma = (vma < m[i][j]) ? m[i][j] : vma;
         }
    *mmi = vmi;      *mma = vma;
}

main()
{  double x,   y,  dx, zmin, zmax;
   int    nr,  nc, i,  j;
   char   in[5];
   char   bu0[] = {"Datos desde un archivo"};
   char   bu1[] = {"Vista suroeste"};
   char   bu2[] = {"Vista sureste"};
   char   bu3[] = {"Vista con rotacion RO = -30 G"};
   char   bu4[] = {"Vista con rotacion RO = 30 G"};
   FILE   *fpi;

   printf("\n\n\n\n\n\n Nombre del archivo de datos? ");   scanf("%s",ae);
   fpi   = fopen(ae,"r");
   if(fpi ==NULL) { printf("\nERROR, No puedo abrir el archivo de ent.\n");
                 return; }
   fscanf(fpi,"%d %d",&nr, &nc);  /* El 1er renglon del arch. tiene la dim. */
    lvm(&zmin, &zmax, nr, nc, fpi);
   printf("\nLa matriz es %d*%d \n", nr, nc);          /* Muestra la matriz */
   for(i = 0; i < nr; i++) {
       printf("\n");
       for(j = 0; j < nc; j++)  printf("%lf ", m[i][j]);
   }             i = getchar();                          /* Anular ENTER */
   dx  = fabs(zmax-zmin)/10.0;    /* Ajuste de los valores maximo y minimo */
   x   = zmin - dx;    y = zmax + dx;
   plt00();
   pltfr(1);
```

```
   plot( 3.6, 0.1, UP);                                /* Marco Superior D */
   plot( 0.1, 0.1, DOWN);    plot( 0.1, 3.6, DOWN);
   plot( 3.6, 3.6, DOWN);    plot( 3.6, 0.1, DOWN);
   plot(-0.1, 0.1, UP);                                /* Marco Superior I */
   plot(-3.6, 0.1, DOWN);    plot(-3.6, 3.6, DOWN);
   plot(-0.1, 3.6, DOWN);    plot(-0.1, 0.1, DOWN);
   plot( 3.6, -0.1, UP);                               /* Marco Inferior D */
   plot( 0.1, -0.1, DOWN);    plot( 0.1, -3.6, DOWN);
   plot( 3.6, -3.6, DOWN);    plot( 3.6, -0.1, DOWN);
   plot(-0.1, -0.1, UP);                               /* Marco Inferior I */
   plot(-3.6, -0.1, DOWN);    plot(-3.6, -3.6, DOWN);
   plot(-0.1, -3.6, DOWN);    plot(-0.1, -0.1, DOWN);
   pltso(1.75, 1.75, -1.85,  1.85);
   visnh();
   visds(x, &m[0][0], y, 1, NR, NR, 1, NC, NC, 0.25, 0.25, 1, 1, pltca);
   plotr();
   pltso(1.75, 1.75,  1.85,  1.85);
   visnh();
   visds(x, &m[0][0], y, 1, NR, NR, 1, NC, NC, 0.25, 0.25, -1, 1, pltca);
   plotr();
   pltso(1.75, 1.75, -1.85,  -1.85);
   pltrv(x, &m[0][0], y, NR, NC, -30, pltca);     plotr();
   pltso(1.75, 1.75, 1.85,  -1.85);
   pltrv(x, &m[0][0], y, NR, NC, 30, pltca);      plotr();

   plttp(-2.5,  4.0, -0.1, bu1, 0.0, (sizeof(bu1)-1), 1, pltmv);
   plttp( 1.0,  4.0, -0.1, bu2, 0.0, (sizeof(bu2)-1), 1, pltmv);
   plttp(-3.2, -4.0, -0.1, bu3, 0.0, (sizeof(bu3)-1), 1, pltmv);
   plttp( 0.4, -4.0, -0.1, bu4, 0.0, (sizeof(bu4)-1), 1, pltmv);
   pltlla(bu0, sizeof(bu0)-1, in, 0, 3);
   pltho();
   pltclose();
}
-------------------------------------------------------------------------
Datos en el archivo
15   7
0.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  2.0  2.0  0.0  0.0
0.0  0.0  0.0  2.0  2.0  0.0  0.0
0.0  0.0  0.0  2.0  2.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  1.0  1.0  0.0  0.0
1.0  1.0  1.0  0.0  0.0  0.0  0.0
1.0  1.0  1.0  0.0  0.0  0.0  0.0
1.0  1.0  1.0  0.0  0.0  0.0  0.0
```

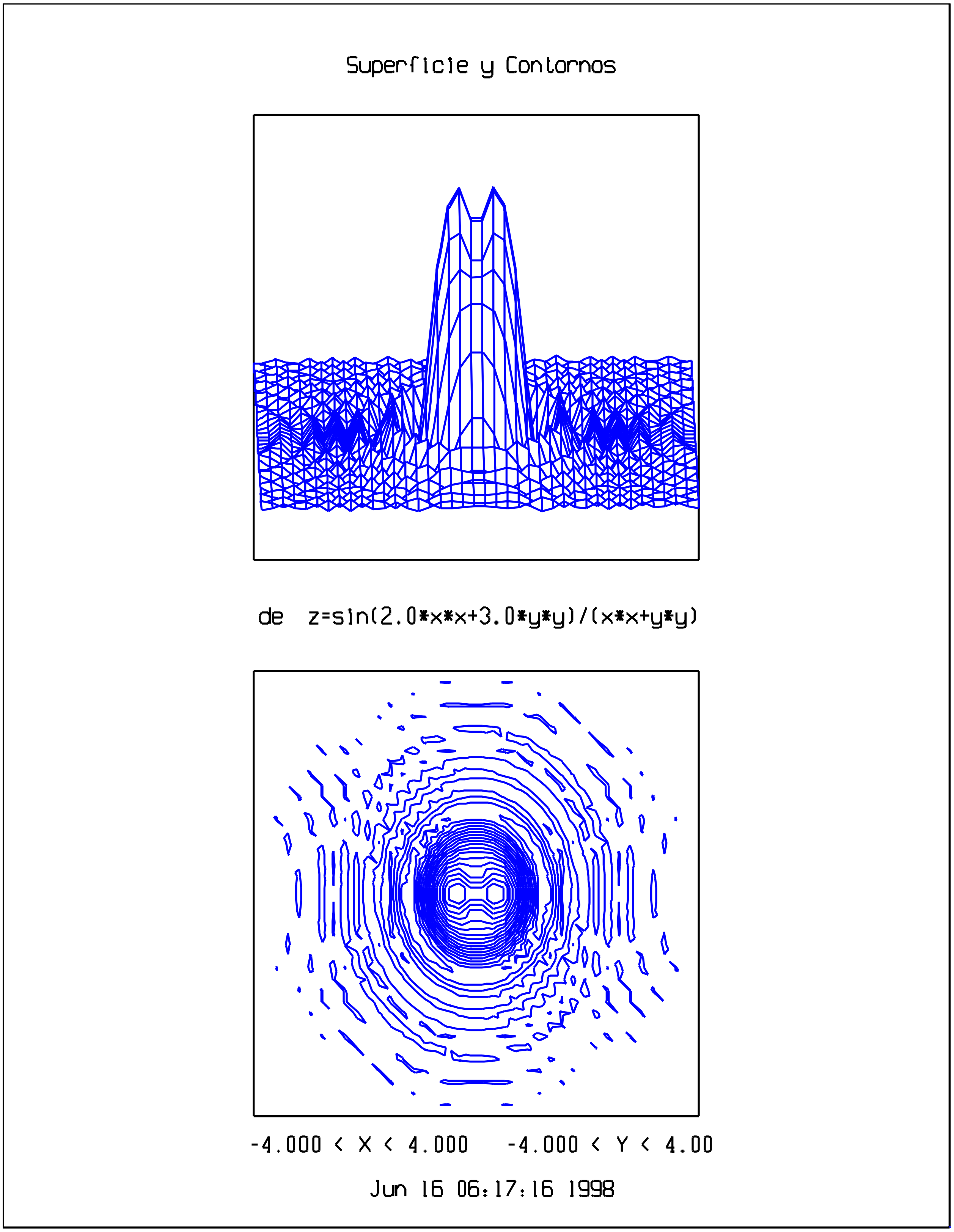

Datos desde un archivo
Vista suroeste
Vista sureste
Vista con rotacion RO = -30 G
Vista con rotacion RO = 30 G
Jun 21 04:02:17 1998

```
/*          SUPERFICIES VISTA INCLINADA
            [J.P.H.E        JUNIO-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"

#define NX      35   /* Dimension de la matriz de valores para la superficie */
#define NY      35
#define xmi  -6.28                              /* Dominio de la funcion */
#define xma   3.14
#define ymi  -3.0
#define yma   3.0
#define fun(x,y) cos(x)                         /* Funcion */
double  WY[NX][NY];

void xy(double dx, double dy)                  /* Dos graficas superiores */
{ double x, y;
  int    i, j;
  x   = xmi;                        /* Calculo de los valores de la funcion */
  for(i=0 ; i < NY; i++, x+=dx)
      for (j=0, y=ymi; j < NX; j++, y+=dy) {
          if( y<=0.0) WY[i][j] = 0.0;                 /* Colocar un plano */
          else WY[i][j] = fun(x,y);        }
}

void yx(double dx, double dy)                  /* Dos graficas inferiores */
{ double x, y;
  int    i, j;
  y   = ymi;                       /* Calculo de los valores de la funcion */
  for(i=0 ; i < NY; i++, y+=dy)
      for (j=0, x=xmi ; j < NX; j++, x+=dx) {
          if( y<=0.0) WY[i][j] = 0.0;                /* Colocar un plano */
          else WY[i][j] = fun(x,y);         }
}

main()
{ int    i;
  double dx,dy;
  char   in[50];
  char   bu0[] = {"Vista Inclinada"};
  char   bu1[] = {"RO=20.0,  TI=10.0, (X,Y)-(I,J)"};
  char   bu2[] = {"RO=20.0,  TI=80.0  (X,Y)-(I,J)"};
  char   bu3[] = {"RO=20.0,  TI=10.0  (Y,X)-(I,J)"};
  char   bu4[] = {"RO=20.0,  TI=45.0  (Y,X)-(I,J)"};
  dx = (xma-xmi)/(double)(NX-1);
  dy = (yma-ymi)/(double)(NY-1);

  plt00();
  plotz("P1 V1 ",6);

  pltfr(1);
  plot( 3.6, 0.1, UP);                          /* Marco Superior D */
  plot( 0.1, 0.1, DOWN);    plot( 0.1, 3.6, DOWN);
```

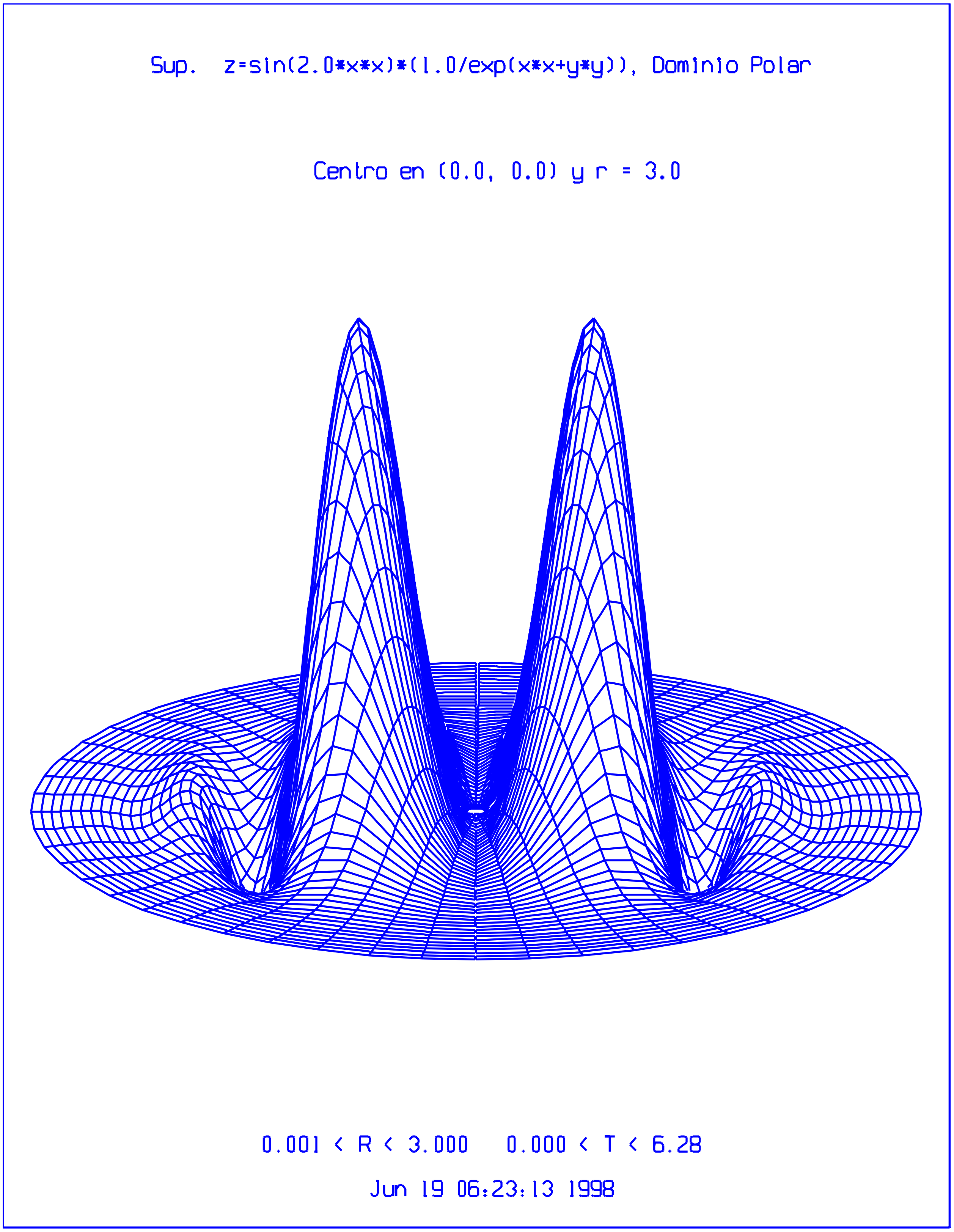
```
  plot( 3.6, 3.6, DOWN);    plot( 3.6, 0.1, DOWN);
  plot(-0.1, 0.1, UP);                             /* Marco Superior I */
  plot(-3.6, 0.1, DOWN);    plot(-3.6, 3.6, DOWN);
  plot(-0.1, 3.6, DOWN);    plot(-0.1, 0.1, DOWN);
  plot( 3.6, -0.1, UP);                            /* Marco Inferior  D */
  plot( 0.1, -0.1, DOWN);    plot( 0.1, -3.6, DOWN);
  plot( 3.6, -3.6, DOWN);    plot( 3.6, -0.1, DOWN);
  plot(-0.1, -0.1, UP);                            /* Marco Inferior  I */
  plot(-3.6, -0.1, DOWN);    plot(-3.6, -3.6, DOWN);
  plot(-0.1, -3.6, DOWN);    plot(-0.1, -0.1, DOWN);

  pltso(1.75, 1.75, -1.85,  1.85);
  xy(dx, dy);
  pltiv(0.0, &WY[0][0], 2.0, NX, NY, 20.0, 10.0, pltca);    plotr();

  pltso(1.75, 1.75,  1.85,  1.85);
  xy(dx, dy);
  pltiv(0.0, &WY[0][0], 2.0, NX, NY, 20.0, 80.0, pltca);    plotr();

  pltso(1.75, 1.75, -1.85,  -1.85);
  yx(dx, dy);
  pltiv(0.0, &WY[0][0], 2.0, NX, NY, 20.0, 10.0, pltca);    plotr();

  pltso(1.75, 1.75, 1.85,  -1.85);
  yx(dx, dy);
  pltiv(0.0, &WY[0][0], 2.0, NX, NY, 20.0, 45.0, pltca);    plotr();

  plttp(-3.3,  4.0, -0.1, bu1, 0.0, (sizeof(bu1)-1), 1, pltmv);
  plttp( 0.4,  4.0, -0.1, bu2, 0.0, (sizeof(bu2)-1), 1, pltmv);
  plttp(-3.3, -4.0, -0.1, bu3, 0.0, (sizeof(bu3)-1), 1, pltmv);
  plttp( 0.4, -4.0, -0.1, bu4, 0.0, (sizeof(bu4)-1), 1, pltmv);

  i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
  pltlla(bu0, (sizeof(bu0) - 1), in, i, 3);
  pltho();
  pltclose();
}
```

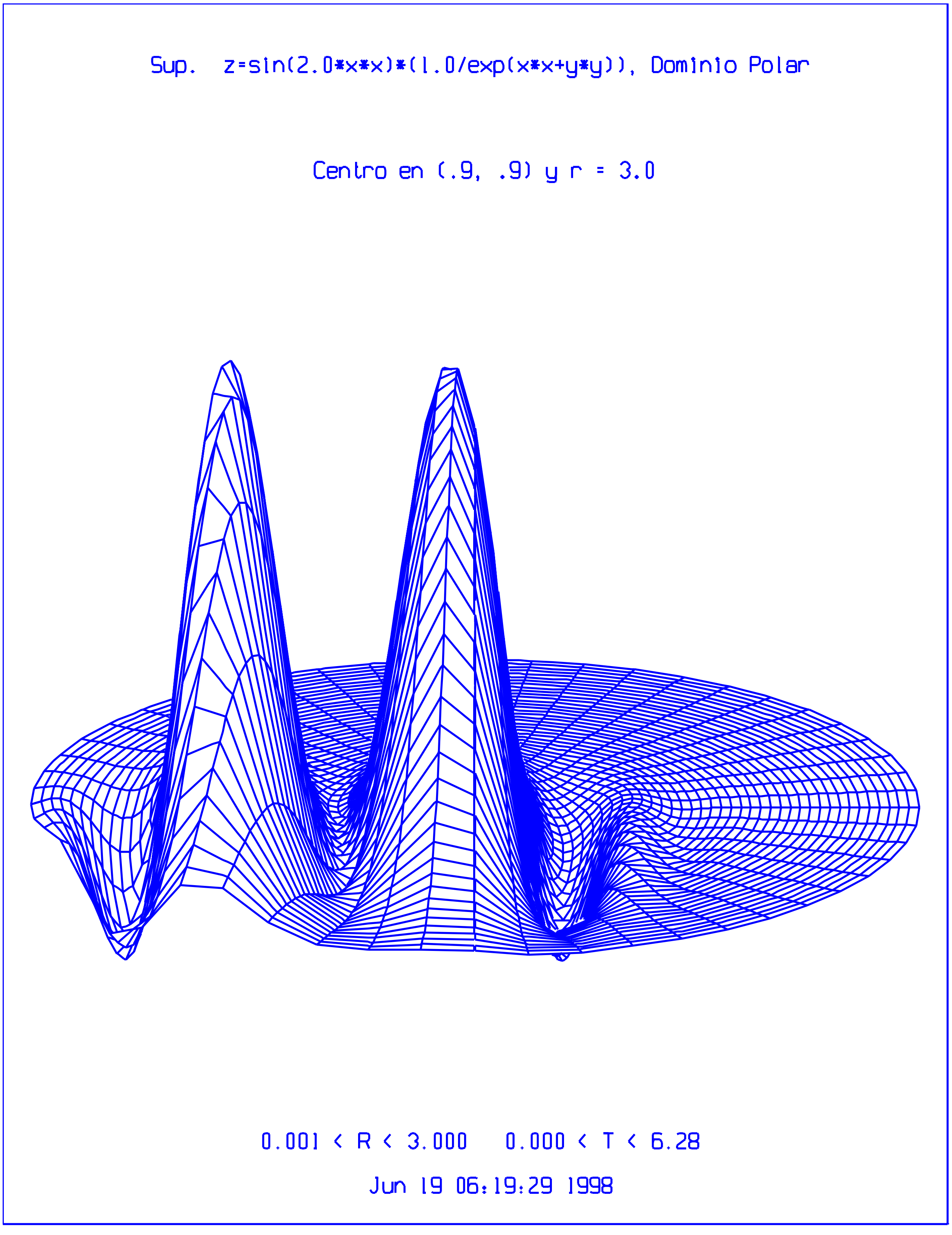

Vista Inclinada
RO=20.0, TI=10.0, (X,Y)=(I,J)
RO=20.0, TI=80.0 (X,Y)=(I,J)
RO=20.0, TI=10.0 (Y,X)=(I,J)
RO=20.0, TI=45.0 (Y,X)=(I,J)
-6.280 < X < 3.140 -3.000 < Y < 3.00
Jun 21 04:01:02 1998

```c
/*          SUPERFICIES (PLTRV)  y  CONTORNOS (PLTKP)
            [J.P.H.E                        JUN-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"
#define NX     40 /* Dimension de la matriz de valores para la superficie */
#define NY     40
#define xmi   -4.0                            /* Dominio de la funcion */
#define xma    4.0
#define ymi   -4.0
#define yma    4.0
#define fun(x,y)   sin(2.0*x*x+3.0*y*y)/(x*x+y*y)             /* Funcion */
double  WA[NX][NY];

main()
{ int    i,  j;
  double dx, dy, zmi, zma, x, y;
  char   in[50];
  char   bu[] = {"Superficie y Contornos"};
  char   bin[] = {"de  z=sin(2.0*x*x+3.0*y*y)/(x*x+y*y)"};
  dx  =  fabs(xma-xmi)/(double)(NX-1);
  dy  =  fabs(yma-ymi)/(double)(NY-1);
  zmi =  fun(xmi, ymi);
  zmi =  (zmi < 0.0) ? 0.0 : zmi;                zma =  zmi;

  for(i=0, y=ymi ; i < NX; i++, y += dy)  {       /* Calculo de la funcion */
      for(j=0, x=xmi ; j < NY; j++, x += dx) {
          WA[i][j] =  fun(x, y);
          if(WA[i][j] < zmi) zmi=WA[i][j];  /* Calculo del valor min. y max.*/
          if(WA[i][j] > zma) zma=WA[i][j];
      }
  }

  plt00();
  pltfr(1);
  plot(2.0,  0.5, UP);                             /*   Marco Superior  */
  plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
  plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);
  plot(2.0,  -0.5, UP);                            /*   Marco Inferior */
  plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
  plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);
  plttp(-1.9, 0.0, -0.11, bin, 0.0, (sizeof(bin) - 1), 1, pltmv);
  i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
  pltlla(bu, (sizeof(bu) - 1), in, i, 1);
  pltso(2.0, 2.0, 0.0,  2.5);                            /* Superficie */
  pltrv(zmi, &WA[0][0], zma, NX, NY, 8.0, pltca);
  plotr();
  pltso(2.0, 2.0, 0.0, -2.5);                            /*  Contorno  */
  pltkp(zmi, &WA[0][0], zma, 20, (NX/64+1), NX, (NY/64+1), NY, pltca);
  plotr();
  pltho();
  pltclose();
}
```


Superficie y Contornos
de z=sin(2.0*x*x+3.0*y*y)/(x*x+y*y)
-4.000 < X < 4.000 -4.000 < Y < 4.00
Jun 16 06:17:16 1998

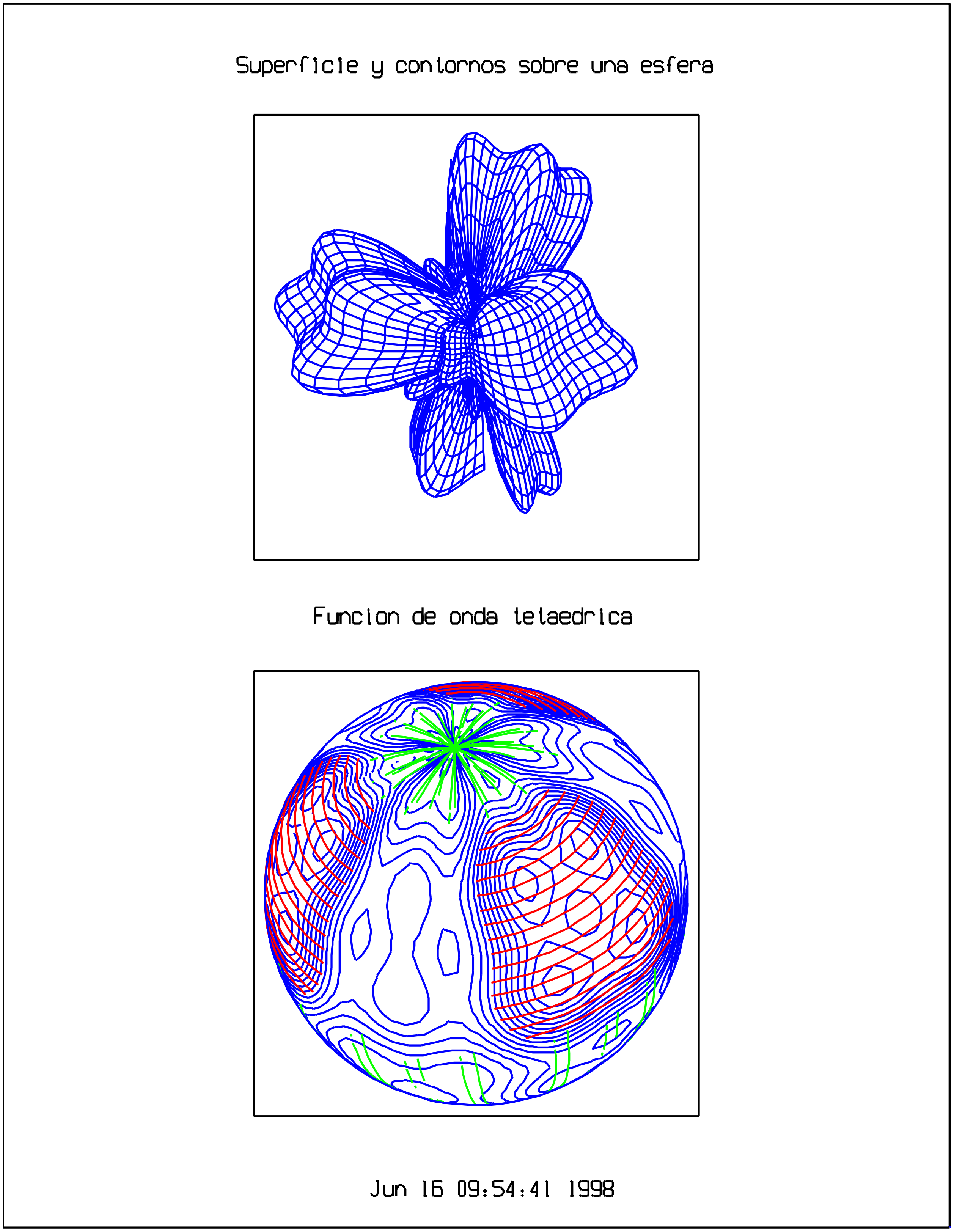
```
/*     SUPERFICIES, DOMINIO POLAR (PLTPV)
       [J.P.H.E.                    1995]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"

#define NX      45
#define NY      53
#define fun(x,y) sin(2.0*x*x)*(1.0/exp(x*x+y*y))

double WA[NX][NY];

main()
{ int     i,  j,  nci;
  double  zmi, zma,  x,  y;
  double  xc, yc, r, ri, rf, dr, t, tmi, tma, dt;
  char    in[50];
  char    bu[] = {"Sup.  z=sin(2.0*x*x)*(1.0/exp(x*x+y*y)), Dominio Polar"};
// char    bu1[]= {"Centro en (.9, .9) y r = 3.0"};
  char    bu1[]= {"Centro en (0.0, 0.0) y r = 3.0"};

  tmi  =  0.0;        tma = 6.28318;
  xc   =  0.0;        yc  = 0.0;                       /* Centro */
  ri   =  0.001;      rf  = 3.1;
  dr   = (rf - ri)/(double) (NX-1);
  dt   = (tma-tmi)/(double) (NY-1);

  x    = ri*cos(tmi) + xc;
  y    = ri*sin(tmi) + yc;
  zmi = fun(x, y);
  zma = zmi;
  for( r=ri, i=0 ; i < NX; i++, r += dr)
       for( t=tmi, j=0 ; j < NY; j++, t += dt) {
            x  =  r*cos(t) + xc;
            y  =  r*sin(t) + yc;
            WA[i][j] =  fun(x, y);
            if(WA[i][j] < zmi) zmi=WA[i][j];
            if(WA[i][j] > zma) zma=WA[i][j];
      }

 plt00();
 pltfr(1);
 pltso(4.0, 4.0, 0.0, 0.0);

 pltpv(zmi, &WA[0][0], zma, NX, NY, pltca);

 nci = pltgi(in, ri, 'R', rf, tmi, 'T', tma);
 pltlla(bu, (sizeof(bu)-1), in, nci, 1);
 plttp(-1.4, 4.0, -0.11, bu1, 0.0, (sizeof(bu1) - 1), 1, pltmv);
 pltho();
 pltclose();
}
```


Sup. z=sin(2.0*x*x)*(1.0/exp(x*x+y*y)), Dominio Polar
Centro en (0.0, 0.0) y r = 3.0
0.001 < R < 3.000   0.000 < T < 6.28
Jun 19 06:23:13 1998

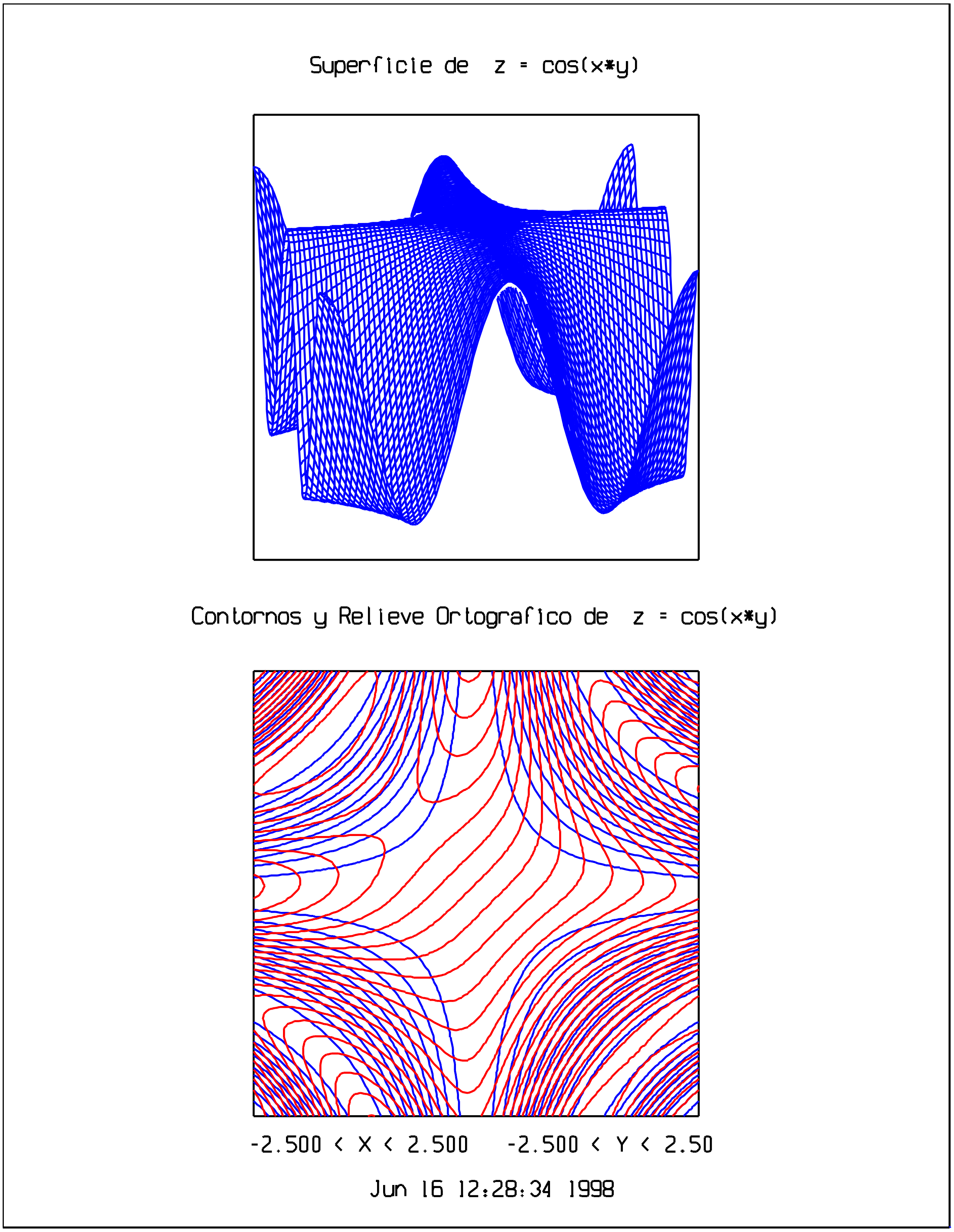

Sup. z=sin(2.0*x*x)*(1.0/exp(x*x+y*y)), Dominio Polar
Centro en (.9, .9) y r = 3.0
0.001 < R < 3.000   0.000 < T < 6.28
Jun 19 06:19:29 1998

```
/*                  DEM19C - FUNCION DE ONDA TETAEDRICA
  SUPERFICIE (PLTSV) Y CONTORNOS SOBRE UNA ESFERA (PLTKP, PLTKX, PLTKY)
               [ H.V.M.- MAY-1975        A.B.C.G. - 1992]
*/
#include <math.h>
#include "plotp.h"

#define  NT  45
#define  NP  90

double  O1[3][3];                /* Matriz de rotacion de EULER          */
double  EF[NP][NT];              /* Matriz con los valores de la funcion */

void d19sp( PH, TH, P)           /* Funcion de movimiento de la pluma    */
double PH, TH;
int    P;
{
   double the, phi, x, y, z, u, v, w, ro, fi;

   the = 3.14159 * TH;
   phi = 6.28318 * PH;
   x   = sin(the) * cos(phi);
   y   = sin(the) * sin(phi);
   z   = cos(the);
   u   = O1[0][0]*x + O1[0][1]*y + O1[0][2]*z;
   v   = O1[1][0]*x + O1[1][1]*y + O1[1][2]*z;
   w   = O1[2][0]*x + O1[2][1]*y + O1[2][2]*z;
   ro  = sqrt(u*u+v*v);
   fi  = atan2(v,u)/6.28318;
   pltpo(fi, ro, (int)(P && (w >= 0.0)) );
}


main()
{  int     i, j;
   double  s, dt, dp, t, p, tf, pf, ra;
   char    bu[]  = {"Superficie y contornos sobre una esfera"};
   char    bu1[] = {"Funcion de onda tetaedrica"};

   s = -1.0;
   plteu(&O1[0][0], 10.0, -60.0, 10.0);

   dt = 3.14159 / (double)(NT-1);
   dp = 6.28318 / (double)(NP);
   t  = 3.14159;

   for(i = 0 ; i < NT ; i++) {
       p  = 0.475;
       for (j = 0 ; j < NP ; j++) {
            tf = 0.67 * (1.0 + 0.5*cos(3.0*t) + s*0.1667*cos(9.0*t) );
            pf = 0.67 * (1.0 + 0.5*sin(3.0*p) - s*0.1667*sin(9.0*p) );
            ra = tf*pf*0.95;
            EF[j][i] = ra;
            p += dp;
```

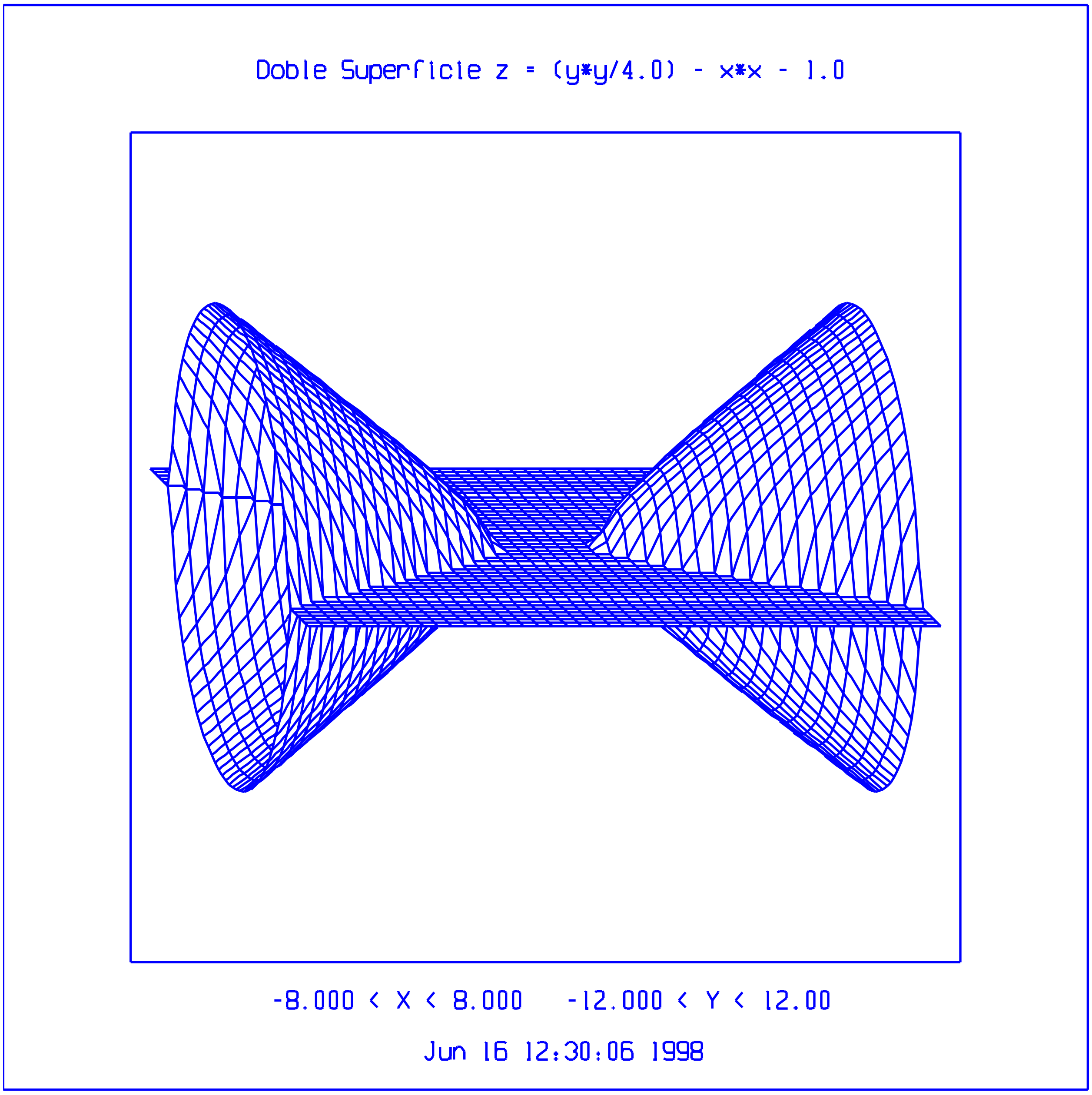
```
        }
        t -= dt;
    }

    plt00();
    plotz("P2 V1 ",6);                  /*  de color negro  el marco */
    pltfr(1);
    plot(2.0,  0.5, UP);                            /*   Marco Superior  */
    plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
    plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);

    plot(2.0,  -0.5, UP);                           /*   Marco Inferior */
    plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
    plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);

    plttp(-1.4, 0.0, -0.11, bu1, 0.0, (sizeof(bu1) - 1), 1, pltmv);
    pltlla(bu, (sizeof(bu) - 1), bu, 0, 1);

    pltso(2.0, 2.0, 0.0,  2.5);
    pltsv(&EF[0][0], NP, NT, 1.0, &O1[0][0], vissp, pltpo);
    plotr();

    plotz("P1 ",3);                     /*  de color azul la esfera  */
    pltso(1.9, 1.9, 0.0,  -2.5);
    pltkp(0.0 , &EF[0][0], 1.0, 20, 3, NP, 2, NT, d19sp);

    plotz("P2 ",3);                     /*  rayado en X de negro */
    pltkx(0.50, &EF[0][0], 1.0, NP, NT, d19sp);

    plotz("P3 ",3);                     /*  rayado en Y de verde   */
    pltky(0.80, &EF[0][0], 1.0, NP, NT, d19sp);

    pltig(0.0, 0.0, 1, pltpo);
    pltig(1.0, 1.0, 2, pltpo);
    pltig(0.0, 1.0, 3, pltpo);
    p = 100.0/99.0;
    t = 0.0;
    for(i=0; i<100; i++, t += p) pltig(t, 1.0, 4, pltpo);

    plotr();
    pltho();
    pltclose();
}
```


Superficie y contornos sobre una esfera
Funcion de onda tetraedrica
Jun 16 09:54:41 1998

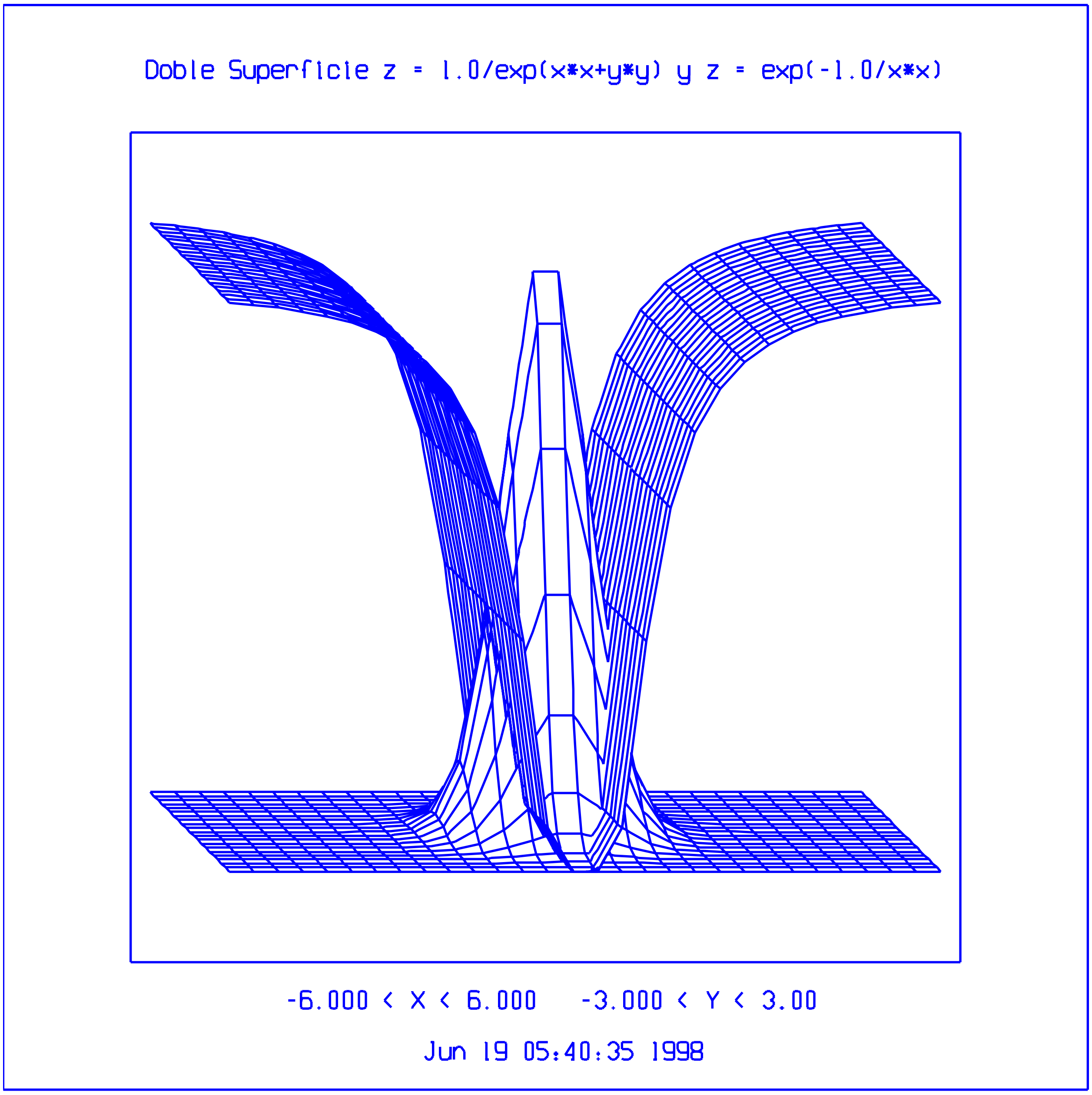
```
/*  CONTORNOS (PLTKP) y RELIEVE ORTOGRAFICO (PLTOR)
             [J.P.H.E   JUL-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"
#define   NX         80
#define   NY         80
#define   xmi       -2.5
#define   xma        2.5
#define   ymi       -2.5
#define   yma        2.5
#define   fun(x,y)   cos(x*y)
double    WA[NX][NY];

main()
{ int    i,j;
  double dx, dy, zmi, zma, x, y;
  char   in[50];
  char   bin[] = {"Contornos y Relieve Ortografico de  z = cos(x*y)"};
  char   bu[]  = {"Superficie de  z = cos(x*y)"};
  dx  = fabs(xma-xmi)/(double)(NX-1);
  dy  = fabs(yma-ymi)/(double)(NY-1);
  zmi = fun(xmi, ymi);
  zmi = (zmi < 0.0) ? 0.0 : zmi;
  zma = zmi;
  for(i=0, y=ymi ; i < NX; i++, y += dy)  {
       for (j=0, x=xmi ; j < NY; j++, x += dx) {
            WA[i][j] =  fun(x, y);
            if(WA[i][j] < zmi) zmi=WA[i][j];
            if(WA[i][j] > zma) zma=WA[i][j];   }
  }
  plt00();
  plotz("P1 V1 ",6);
  pltfr(1);
  plot(2.0,  0.5, UP);                            /*   Marco Superior  */
  plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
  plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);
  plot(2.0,  -0.5, UP);                           /*   Marco Inferior */
  plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
  plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);
  pltso(2.0, 2.0, 0.0,  2.5);
  pltrv(zmi, &WA[0][0], zma, NX, NY, 10.0, pltca);
  plotr();
  pltso(2.0, 2.0, 0.0,  -2.5);
  pltkp(zmi, &WA[0][0], zma, 15, (NX/64+1), NX, (NY/64+1), NY, pltca);
  plotz("P2 ",3);
  pltor(zmi, &WA[0][0], zma, 40, (NX/64+1), NX, (NY/64+1), NY, pltca);
  plotr();                                            plotz("P1 ",3);
  plttp(-2.5, 0.0, -0.11, bin, 0.0, (sizeof(bin) - 1), 1, pltmv);
  i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
  pltlla(bu, (sizeof(bu) - 1), in, i, 1);
  pltho();                    pltclose();
}
```

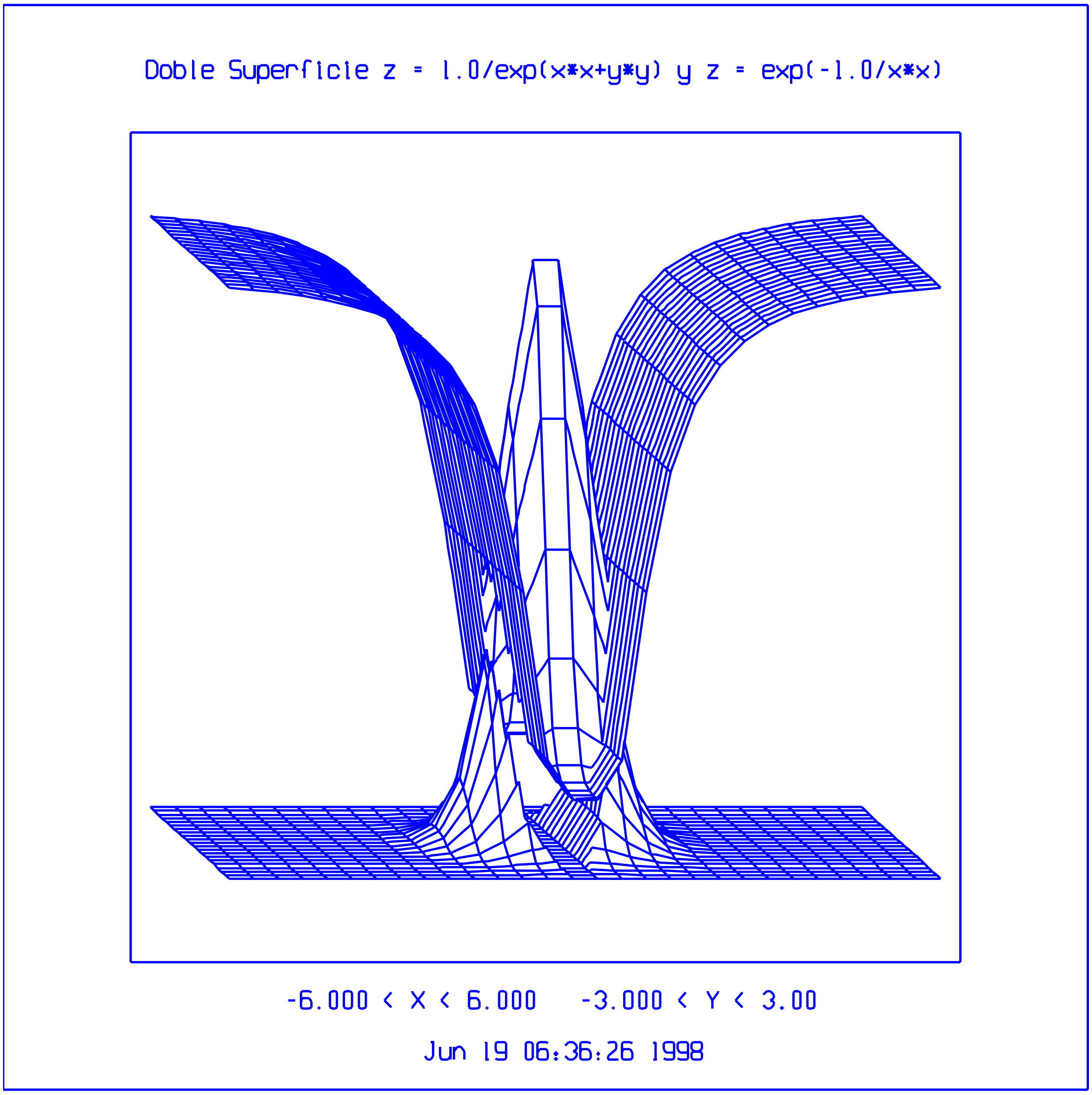

Superficie de z = cos(x*y)
Contornos y Relieve Ortografico de z = cos(x*y)
-2.500 < X < 2.500
-2.500 < Y < 2.50
Jun 16 12:28:34 1998

```
/*              DOBLE SUPERFICIE (VISDO)
                [J.P.H.E     JULIO-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"
#define  NX       45    /*Dim. de las matrices de valores para la superficie */
#define  NY       45
#define  xmi    -8.0        /* Calculamos los valores de cada funcion con el */
#define  xma     8.0        /* mismo dominio. Aunque el dominio de cada fun- */
#define  ymi   -12.0        /* cion puede ser diferente.                     */
#define  yma    12.0
#define  fun(x,y)  (y*y/4.0) - x*x - 1.0               /* Funcion a graficar */
double   WA[NX][NY], WB[NX][NY];

main()
{
  int    i,j;
  double dx, dy, zmi, zma, x, y, d;
  char   in[50];
  char   bu[] = {"Doble Superficie z = (y*y/4.0) - x*x - 1.0"};

  dx  = (xma-xmi)/(double)(NX-1);
  dy  = (yma-ymi)/(double)(NY-1);
  zmi = fun(xmi, ymi);
  zmi = (zmi < 0.0) ? 0.0 : zmi;
  zma = zmi;

  for(i=0, y=ymi; i < NX; i++, y+=dy)  {      /* Calculo  de la superficie */
      for (j=0, x=xmi; j < NY; j++, x+=dx) {
           d  = fun(x, y);
           if( d < 0.0 ) d = 0.0;
           d  = sqrt(d);
           WA[i][j] =  d;
           WB[i][j] = -d;
           if(WA[i][j] < zmi) zmi=WA[i][j]; /* Calculo del valor maximo y */
           if(WA[i][j] > zma) zma=WA[i][j]; /* del valor minimo           */
           if(WB[i][j] < zmi) zmi=WB[i][j];
           if(WB[i][j] > zma) zma=WB[i][j];
      }
  }

  plt00();
  plotz("P1 V1 ",6);                    /* Comandos del graficador DMP-29 */
  pltfr(6);
  pltso(3.1, 3.1, 0.0, 0.0);

  visdo(zmi, &WA[0][0], &WB[0][0], zma, NX, NX, NY, NY, 0.2, 0.2, 0.0, 1, -1, pltca);

  i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
  pltlla(bu, (sizeof(bu) - 1), in, i, 6);                   /* Letreros */
  pltho();                   pltclose();
}
```


Doble Superficie z = (y*y/4.0) - x*x - 1.0
-8.000 < X < 8.000   -12.000 < Y < 12.00
Jun 16 12:30:06 1998

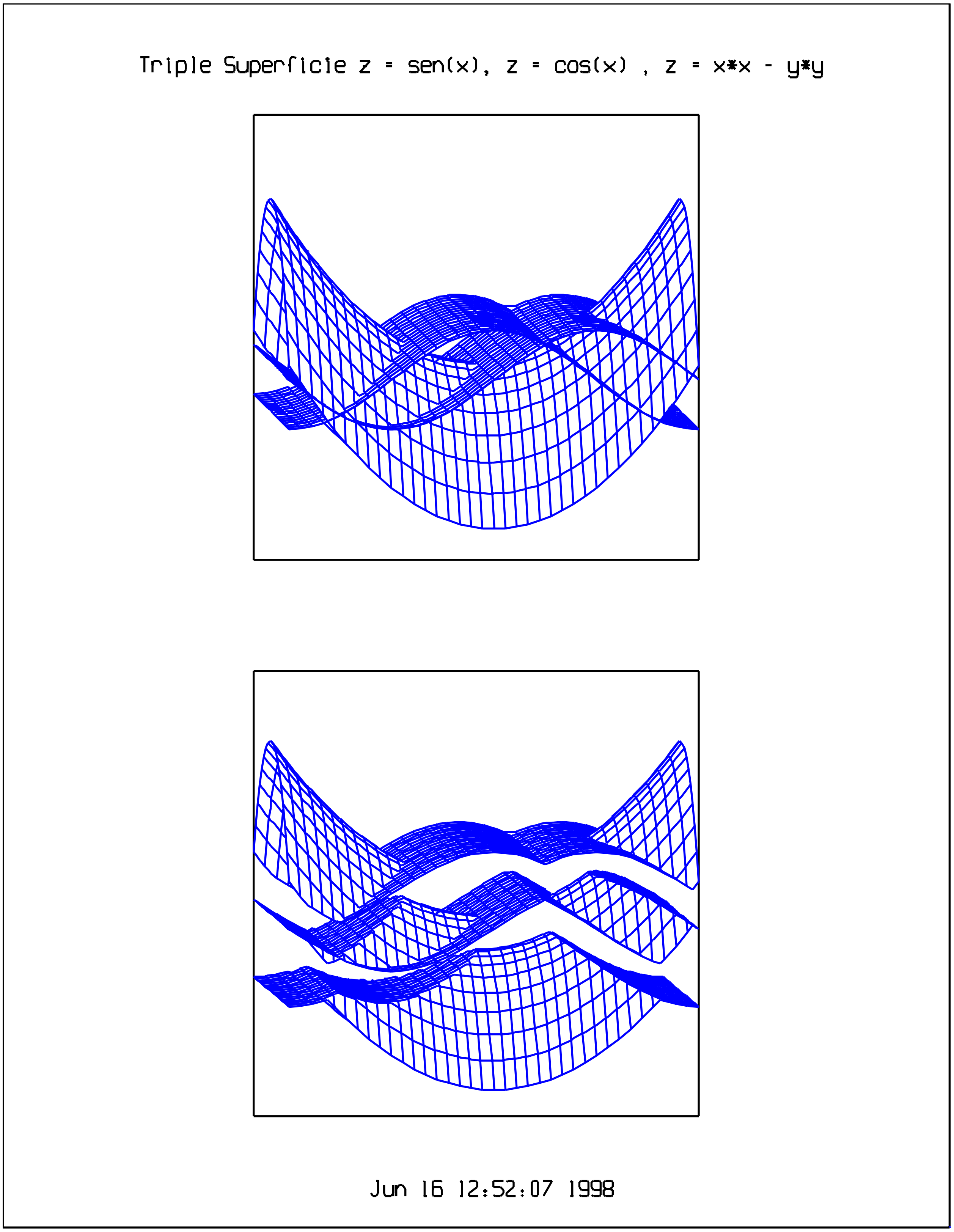
```
/*         DOBLE SUPERFICIE (VISDO)
           [J.P.H.E   JUL-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"

#define  NX       30    /*Dim. de las matrices de valores para la superficie */
#define  NY       21
#define  xmi    -6.0        /* Calculamos los valores de cada funcion con el */
#define  xma     6.0        /* mismo dominio. Aunque el dominio de cada fun- */
#define  ymi    -3.0        /* cion puede ser diferente.                     */
#define  yma     3.0
double   WA[NX][NY], WB[NX][NY];

main()
{
 int    i,j;
 double dx, dy, zmi, zma, x, y, d;
 char   in[50];
 char   bu[] = {"Doble Superficie z = 1.0/exp(x*x+y*y) y z = exp(-1.0/x*x)"};

 dx  = (xma-xmi)/(double)(NX-1);
 dy  = (yma-ymi)/(double)(NY-1);
 zmi = sin(xmi);
 zma = zmi;

 for(i=0, x=xmi; i < NX; i++, x+=dx)  {
      for(j=0, y=yma; j < NY; j++, y-=dy) {
           WA[i][j] =  1.0/exp(x*x+y*y);      /* 1a. funcion a graficar */
           d        = -1.0/(x*x);
           WB[i][j] =  exp(d);                /* 2a. funcion a graficar */
           if(WA[i][j] < zmi) zmi=WA[i][j];
           if(WA[i][j] > zma) zma=WA[i][j];
           if(WB[i][j] < zmi) zmi=WB[i][j];
           if(WB[i][j] > zma) zma=WB[i][j];
      }
 }
 dx = fabs(zma-zmi)/8.0;
 x  = zmi - dx;    y = zma + dx;

 plt00();
 plotz("P1 V1 ",6);
 pltfr(6);
 pltso(3.1, 3.1, 0.0, 0.0);

 visdo(x, &WA[0][0], &WB[0][0], y, NX, NX, NY, NY, 0.1, 0.1, 0.0, 1, -1, pltca);
 i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
 pltlla(bu, (sizeof(bu) - 1), in, i, 6);
 pltho();
 pltclose();
}
```


Doble Superficie z = 1.0/exp(x*x+y*y) y z = exp(-1.0/x*x)
-6.000 < X < 6.000   -3.000 < Y < 3.00
Jun 19 05:40:35 1998


Doble Superficie z = 1.0/exp(x*x+y*y) y z = exp(-1.0/x*x)
-6.000 < X < 6.000   -3.000 < Y < 3.00
Jun 19 06:36:26 1998

```
/*            TRIPLE SUPERFICIE (VISTR)
              [J.P.H.E      JUNIO-1993]
*/
#include <stdio.h>
#include <math.h>
#include "plotp.h"

#define  NX     37         /*Dim. de las matrices de valores para la super. */
#define  NY     18
#define  xmi   -3.1415
#define  xma    3.1415
#define  ymi   -3.0
#define  yma    3.0
double   WA[NX][NY], WB[NX][NY], WC[NX][NY];

main()
{
 int    i,j;
 double dx, dy, zmi, zma, x, y;
 char   in[50];
 char   bu[] = {"Triple Superficie z = sen(x), z = cos(x) , z = x*x - y*y"};

 dx  = (xma-xmi)/(double)(NX-1);
 dy  = (yma-ymi)/(double)(NY-1);
 zmi = 3.0*sin(xmi);
 zma = zmi;

 x   = xmi;                   /* Calculo de los valores de las superficies */
 for(i=0 ; i < NX; i++)  {
      y = yma;
      for (j=0 ; j < NY; j++) {
           WA[i][j] =  3.0*sin(x);
           WB[i][j] =  x*x - y*y;
           WC[i][j] =  3.0*cos(x);
           if(WA[i][j] < zmi) zmi=WA[i][j];  /* Calculo del valor maximo */
           if(WA[i][j] > zma) zma=WA[i][j];  /* y del valor minimo       */
           if(WB[i][j] < zmi) zmi=WB[i][j];
           if(WB[i][j] > zma) zma=WB[i][j];
           if(WC[i][j] < zmi) zmi=WC[i][j];
           if(WC[i][j] > zma) zma=WC[i][j];
           y -= dy;
      }
      x += dx;
 }
 dx = fabs(zma-zmi)/10.0;
 x  = zmi - dx;    y = zma + dx;

 plt00();
 plotz("P1 V1 ",6);                               /*  de color negro */
 pltfr(1);

 plot(2.0,  0.5, UP);                             /*   Marco Superior  */
 plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
 plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);
```

```
 plot(2.0,  -0.5, UP);                            /*   Marco Inferior */
 plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
 plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);

 pltso(2.0, 2.0, 0.0,  2.5);
 vistr(x,&WA[0][0],&WB[0][0],&WC[0][0],y,NX,NX,NY,NY,0.08,0.08,0.0,1,-1,pltca);
 plotr();


 plotz("P2 ",3);                              /*  de color azul */
 pltso(2.0, 2.0, 0.0,  -2.5);
 vistr(x,&WA[0][0],&WB[0][0],&WC[0][0],y,NX,NX,NY,NY,0.08,0.08,0.0,1,1, pltca);
 plotr();

 i = pltgi(in, xmi, 'X', xma, ymi, 'Y', yma);
 pltlla(bu, (sizeof(bu) - 1), bu, 0, 1);
 pltho();
 pltclose();
}
```


Triple Superficie z = sen(x), z = cos(x) , z = x*x - y*y
Jun 16 12:52:07 1998

6.3 Trazo de funciones complejas

Aquí se muestra como generar los datos y el orden de llamado a las funciones que trazan datos de funciones complejas, en general la gráfica de una función compleja es un subconjunto de R^4 que no podemos visualizar, por lo que se emplean diferentes técnicas que nos muestran el comportamiento de una función compleja en R^2 (mapeo comforme) o en R^3 (superficie de valor absoluto y contornos de fase).
La secuencia de llamado típica es:

Generar los datos en un arreglo matricial (superficie de valor absoluto y contornos de fase).
o
Generar los datos en dos arreglos matriciales (mapeo comforme).
o
Obtener los datos desde un archivo y depositarlos en los arreglos correspondientes.

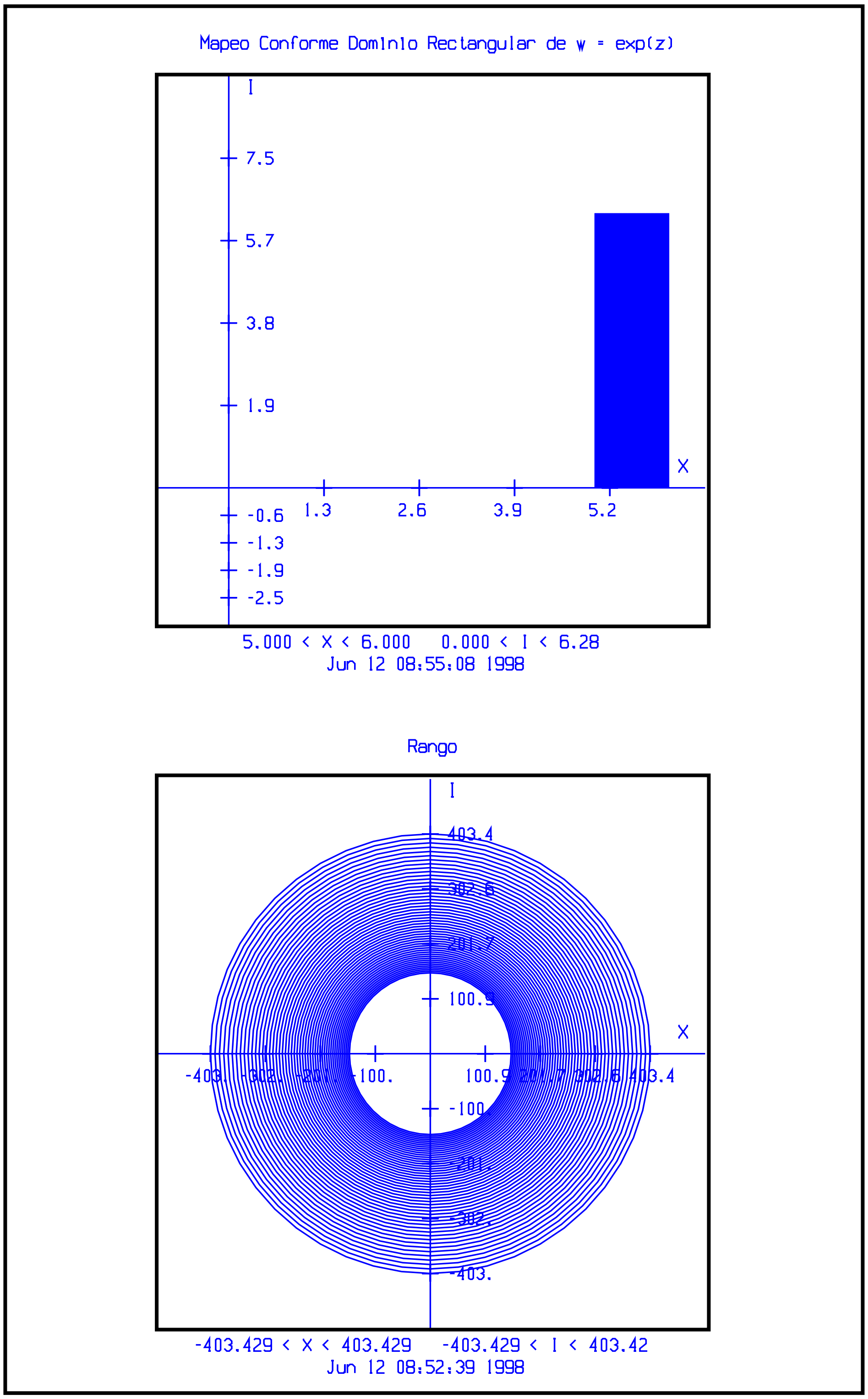
plt00();
pltfr(y);

Generar la superficie de valor absoluto con pltrv, pltiv o visds.
También se pueden generar los contornos de valor absoluto con pltkp.
o
Generar los contornos de fase con pltkc.

pltho();
pltclose();

```c
/*        MAPEO CONFORME DOMINIO RECTANGULAR
              [J.P.H.E.    JULIO-1993]
*/
#include  <stdio.h>
#include  <math.h>
#include  <ctype.h>
#include  <time.h>
#include  "plotp.h"
#include  "ac.h"

#define   npu  41
double    mcwr[npu][npu], mcwi[npu][npu];

main()
{
  double  mipr,  mapr,   mipi,  mapi,  pr,  pi, ex;
  double  x,     y,      z,     dx,    dy;
  double  xmi,   xma,    ymi,   yma;
  int     i,     j,      nci;
  char    bu[] ={"Mapeo Conforme Dominio Rectangular de  w = exp(z)"};
  char    bin[60];
  char    moin,     c;

  xmi  =   2.00;       xma = 10.00;                   /* Dominio del Mapeo */
  ymi  =   0.00;       yma = 3.14159;
  printf("\n\n\n                Mapeo Conforme\n\n");
  printf("\n    Intervalo en X %lf, %lf", xmi, xma);
  printf("\n    Intervalo en Y %lf, %lf", ymi, yma);
  moin = 'E';
  while(!(moin == 'S' || moin == 'N')) {
        printf("\n\n    Modificar los intervalos? (S/N) ");
        scanf("%c", &moin); moin = (islower(moin)) ? _toupper(moin) : moin;
        i=getchar();
  }
  if(moin == 'S') {
     printf("\n    Valor Minimo en X = ");   scanf("%lf", &xmi);
     printf("    Valor Maximo en X = ");     scanf("%lf", &xma);
     printf("    Valor Minimo en Y = ");     scanf("%lf", &ymi);
     printf("    Valor Maximo en Y = ");     scanf("%lf", &yma);
     i = getchar();                                      /* Anular ENTER */
  }
  dx   = fabs(xma-xmi)/(double) (npu-1);          /* Division del dominio */
  dy   = fabs(yma-ymi)/(double) (npu-1);

  mipr = exp(xmi)*cos(ymi); mapr = mipr;
  mipi = exp(xmi)*sin(ymi); mapi = mipi;

  for(i=0, x=xmi; i < npu; i++, x+=dx) {         /* Calculo de la funcion */
      for(j=0, y=ymi; j < npu; j++, y+=dy ) {
           ex  = exp(x);
           pr  = ex*cos(y);
           pi  = ex*sin(y);
           mcwr[i][j] = pr;
           mcwi[i][j] = pi;
```

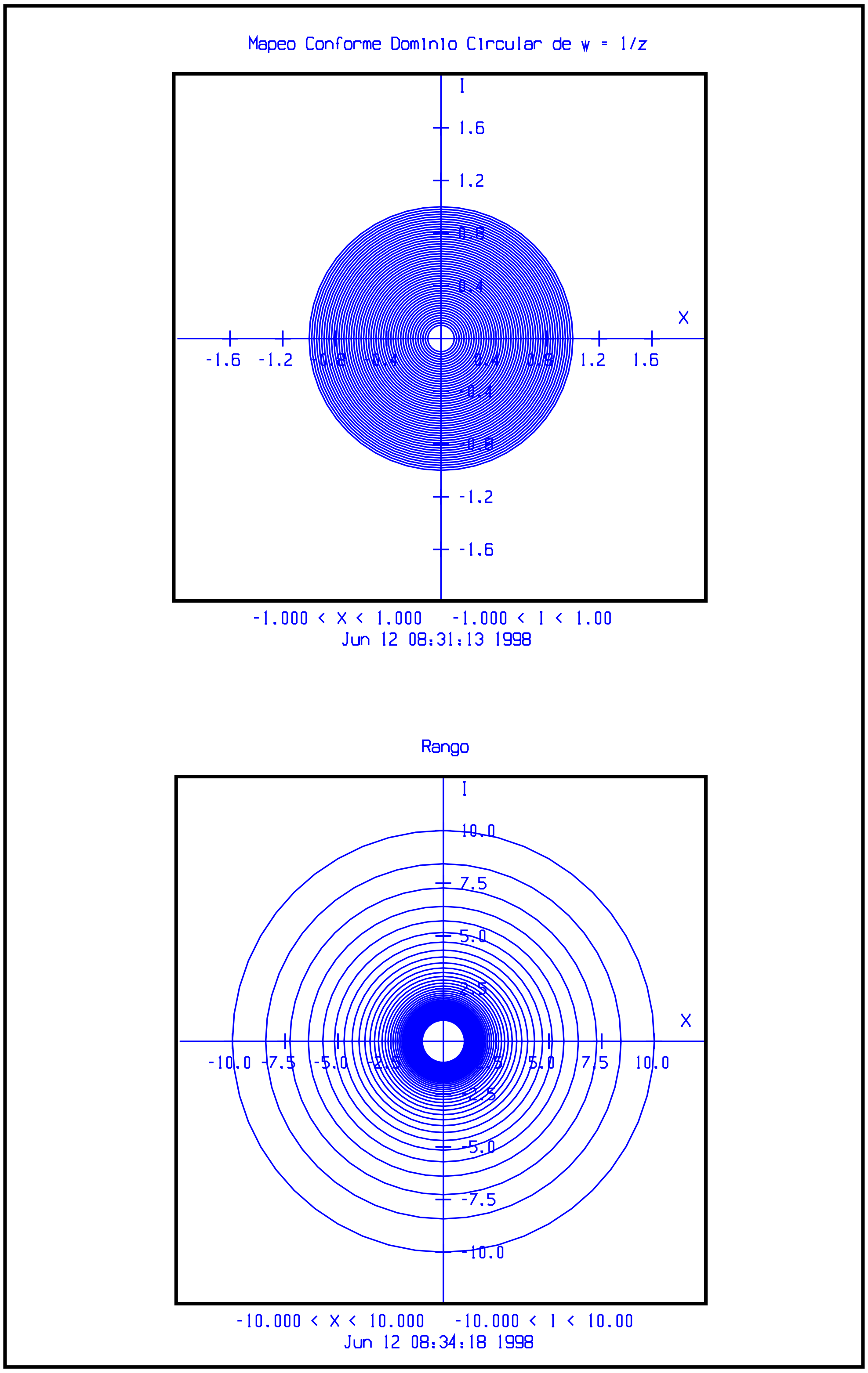
```
         if(pr > mapr) mapr = pr;     /* Calculo del valor min. y max. */
         if(pr < mipr) mipr = pr;
         if(pi > mapi) mapi = pi;
         if(pi < mipi) mipi = pi;
     }
 }

 plt00();
 pltfr(1);
 plot(2.0,  0.5, UP);                            /* Traza el marco Superior */
 plot(-2.0, 0.5, DOWN);    plot(-2.0, 4.5, DOWN);
 plot(2.0,  4.5, DOWN);    plot(2.0,  0.5, DOWN);
 plot(2.0,  -0.5, UP);                           /* Traza el marco Inferior */
 plot(2.0,  -4.5, DOWN);   plot(-2.0, -4.5, DOWN);
 plot(-2.0, -0.5, DOWN);   plot(2.0,  -0.5, DOWN);

 x = fabs(xma-xmi)/2.0;                              /* Grafica el DOMINIO  */
 y = fabs(yma-ymi)/2.0;
 pltso(2.0, 2.0, 0.0,  2.5);
        /* Se ajustan los valores min. y max. para que aparezcan los ejes */
 pltig((xmi-x)>=0.0 ? -1.0 : xmi-x, (ymi-y) >= 0.0 ? -1.0 : ymi-y, 1, pltca);
 pltig((xma+x)<=0.0 ?  1.0 : xma+x, (yma+y) <= 0.0 ?  1.0 : yma+y, 2, pltca);

 for(i=0, x=xmi; i < npu; i++) {
     y = ymi;
     pltig(x, y, 3, pltca);
     for(j=1; j < npu; j++) { y += dy; pltig(x, y, 4, pltca);}
     x += dx;
 }
 pltig(0.0, 0.0, 9, pltca);                                       /* Ejes */
 nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
 plttp(-2.0, -2.3, -0.11, bin, 0.0, nci, 1, pltmv);
 plotr();

 x = fabs(mapr-mipr)/8.0;                             /* Grafica el RANGO  */
 y = fabs(mapi-mipi)/8.0;
 pltso(2.0, 2.0, 0.0, -2.5);
         /* Se ajustan los valores min. y max. para que aparezcan los ejes */
 pltig((mipr-x)>=0.0 ? -1.0 : mipr-x, (mipi-y)>=0.0 ? -1.0 : mipi-y, 1, pltca);
 pltig((mapr+x)<=0.0 ?  1.0 : mapr+x, (mapi+y)<=0.0 ?  1.0 : mapi+y, 2, pltca);
 for(i=0; i < npu; i++) {
     pltig(mcwr[i][0],mcwi[i][0],3,pltca);
     for(j=1; j < npu; j++) pltig(mcwr[i][j],mcwi[i][j],4,pltca);
 }
 pltig(0.0, 0.0, 9, pltca);                                      /* Ejes */
 plotr();

 nci = pltgi(bin, mipr, 'X', mapr, mipi, 'I', mapi);
 pltlla(bu, (sizeof(bu)-1), bin, nci, 1);
 pltho();
 pltclose();
}
```


Mapeo Conforme Dominio Rectangular de w = exp(z)
I
7.5
5.7
3.8
1.9
X
-0.6
-1.3
-1.9
-2.5
1.3
2.6
3.9
5.2
5.000 < X < 6.000   0.000 < I < 6.28
Jun 12 08:55:08 1998
Rango
I
403.4
X
-403.429 < X < 403.429   -403.429 < I < 403.42
Jun 12 08:52:39 1998

```
/*          MAPEO CONFORME DOMINIO CIRCULAR
            [J.P.H.E.          JULIO-1993]
*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "plotp.h"
#include "ac.h"

#define  npu   41
double   mcwr[npu][npu], mcwi[npu][npu];

main()
{
  struct  complex  cx,   cw,    uno;
  double  mipr,  mapr,   mipi,  mapi,    pr,  pi;
  double  x,   y,   xmi, xma, ymi, yma, ri,   rf,  r,  t;
  double  tmi, tma,  xc,  yc, dr,   dt;
  int     i,   j,   nci;
  char    bu[] ={"Mapeo Conforme Dominio Circular de w = 1/z"};
  char    bin[60];
  char    moce, moci, mora;

  tmi  =  0.0;        tma = 6.28318;            /* Dominio del mapeo */
  xc   =  0.0;        yc  = 0.0;
  ri   =  0.01;       rf  = 1.0;

  printf("\n\n\n\n\n           Mapeo Conforme (Dominio Circular)\n\n");
  printf("\n       Centro en X e I      %lf, %lf", xc,  yc);
  printf("\n       Int. del Circ. (rad) %lf, %lf", tmi, tma);
  printf("\n       Radio del Circulo    %lf", rf);
  moce = 'E';
  while(!(moce == 'S' || moce == 'N')) {
        printf("\n\n       Modificar el centro? (S/N) ");
        scanf("%c", &moce); moce = (islower(moce)) ? _toupper(moce) : moce;
        i=getchar();                                /* Anula ENTER */
  }
  if(moce == 'S') {
     printf("\n       Centro (X,I)  = ");   scanf("%lf,%lf", &xc, &yc);
     i = getchar();                                         /* Anula ENTER */
  }
  moci = 'E';
  while(!(moci == 'S' || moci == 'N')) {
        printf("\n\nModificar el Int. del Circ.? (S/N) ");
        scanf("%c", &moci); moci = (islower(moci)) ? _toupper(moci) : moci;
        i=getchar();                                         /* Anula ENTER */
  }
  if(moci == 'S') {
     printf("\nInt. del Circulo (TMI,TMA) = ");   scanf("%lf,%lf", &tmi, &tma);
     i = getchar();                                         /* Anula ENTER */
  }
  mora = 'E';
  while(!(mora == 'S' || mora == 'N')) {
        printf("\n\n          Modificar el Radio? (S/N) ");
```

```
      scanf("%c", &mora); mora = (islower(mora)) ? _toupper(mora) : mora;
}
if(moci == 'S') {
   printf("\nRadio del Circulo (RMI,RMA) = ");   scanf("%lf,%lf", &ri, &rf);
   i = getchar();                                /* Anula ENTER */
}
dr   = fabs(rf - ri)/(double) (npu-1);
dt   = fabs(tma-tmi)/(double) (npu-1);
xmi  = xc - rf;        xma = xc + rf;         r = ri;
ymi  = yc - rf;        yma = yc + rf;         t = tmi;

plt00();
pltfr(1);
plot(2.0,  0.5, UP);                          /* Traza el marco superior */
plot(-2.0, 0.5, DOWN);    plot(-2.0, 4.5, DOWN);
plot(2.0,  4.5, DOWN);    plot(2.0,  0.5, DOWN);
plot(2.0,  -0.5, UP);                         /* Traza el marco inferior */
plot(2.0,  -4.5, DOWN);   plot(-2.0, -4.5, DOWN);
plot(-2.0, -0.5, DOWN);   plot(2.0,  -0.5, DOWN);

x = fabs(xma-xmi)/2.0;                           /* Grafica el DOMINIO  */
y = fabs(yma-ymi)/2.0;
pltso(2.0, 2.0, 0.0,  2.5);
       /* Se ajustan los valores min. y max. para que aparezcan los ejes */
pltig((xmi-x)>=0.0 ? -1.0 : xmi-x, (ymi-y)>=0.0 ? -1.0 : ymi-y, 1, pltca);
pltig((xma+x)<=0.0 ?  1.0 : xma+x, (yma+y)<=0.0 ?  1.0 : yma+y, 2, pltca);

x    = r*cos(t) + xc;
y    = r*sin(t) + yc;
uno  = cmplx(1.0, 0.0);
cw   = cmplx(x, y);
cx   = acdiv(uno, cw);
mipr = cx.x;           mapr = mipr;
mipi = cx.y;           mapi = mipi;

for(i=0; i < npu; i++) {                      /* Calculo de la funcion */
    t  =  tmi;
    x  =  r*cos(t) + xc;                  /* X,Y elementos del dominio */
    y  =  r*sin(t) + yc;
    pltig( x, y, 3, pltca);
    for(j=0; j < npu; j++) {
         x  =  r*cos(t) + xc;
         y  =  r*sin(t) + yc;
         pltig( x, y, 4, pltca);         /* Traza un punto del dominio */
         cw  = cmplx(x, y);
         cx  = acdiv(uno, cw);
         pr  = cx.x;
         pi  = cx.y;
         mcwr[i][j] = pr;
         mcwi[i][j] = pi;
         if(pr > mapr) mapr = pr;    /* Calculo del valor min. y max. */
         if(pr < mipr) mipr = pr;
         if(pi > mapi) mapi = pi;
         if(pi < mipi) mipi = pi;
```

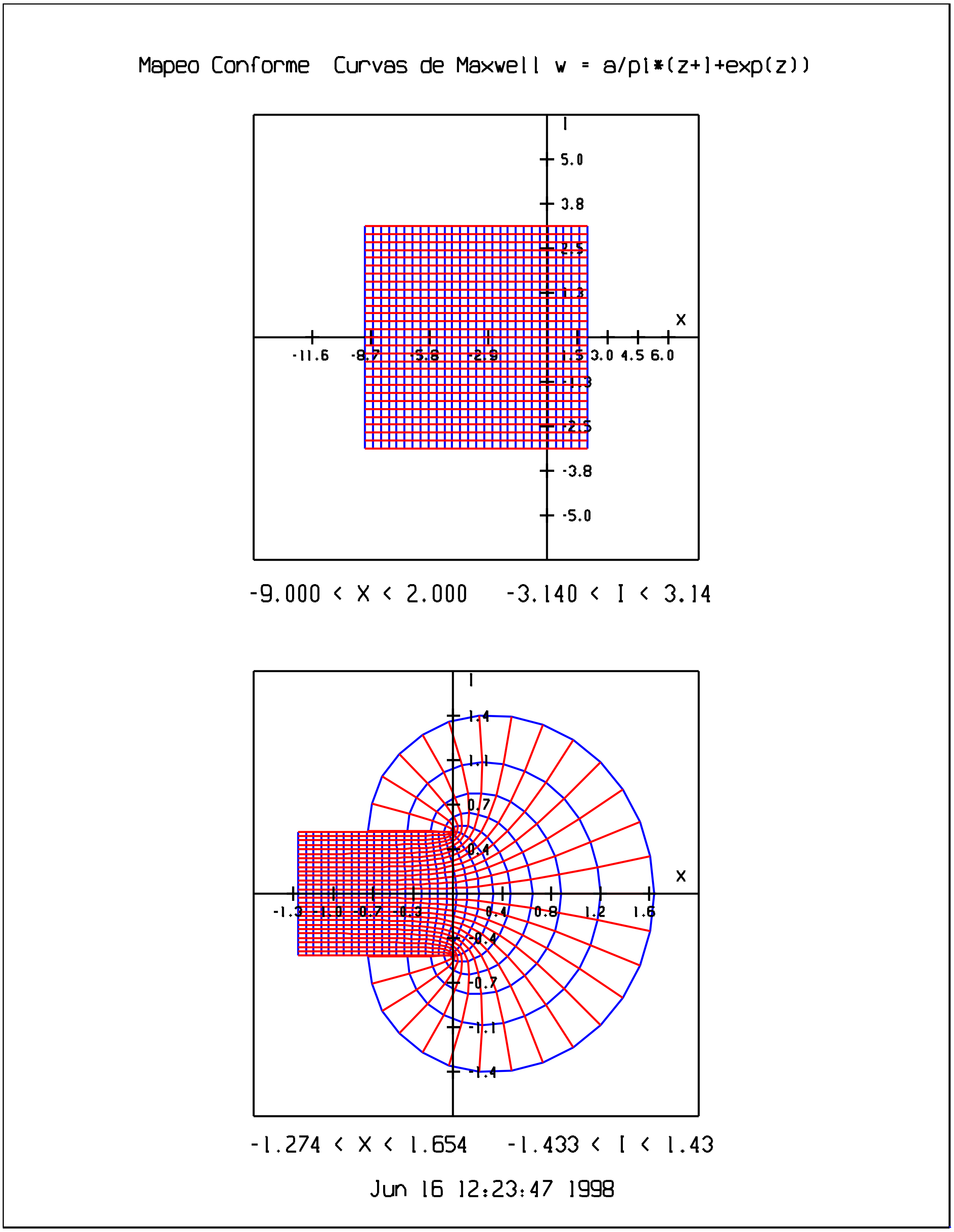
```
        t  +=  dt;
    }
    r +=  dr;
  }
  pltig(0.0, 0.0, 9, pltca);                                       /* Ejes */
  nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
  plttp(-2.0, -2.3, -0.11, bin, 0.0, nci, 1, pltmv);
  plotr();

  x = fabs(mapr-mipr)/8.0;                          /*  Graficar el RANGO  */
  y = fabs(mapi-mipi)/8.0;
  pltso(2.0, 2.0, 0.0, -2.5);
       /* Se ajustan los valores min. y max. para que aparezcan los ejes */
  pltig((mipr-x)>=0.0 ? -1.0:mipr-x, (mipi-y)>=0.0 ? -1.0:mipi-y, 1, pltca);
  pltig((mapr+x)<=0.0 ?  1.0:mapr+x, (mapi+y)<=0.0 ?  1.0:mapi+y, 2, pltca);
  for(i=0; i < npu; i++) {
    pltig(mcwr[i][0],mcwi[i][0],3,pltca);
    for(j=1; j < npu; j++) pltig(mcwr[i][j],mcwi[i][j],4,pltca);
  }
  pltig(0.0, 0.0, 9, pltca);                                      /* Ejes */
  plotr();

  nci = pltgi(bin, mipr, 'X', mapr, mipi, 'I', mapi);
  pltlla(bu, (sizeof(bu)-1), bin, nci, 1);
  pltho();
  pltclose();
}
```


Mapeo Conforme Dominio Circular de w = 1/z
I
X
1.6
1.2
-1.2
-1.6
-1.6
-1.2
1.2
1.6
-1.000 < X < 1.000   -1.000 < I < 1.00
Jun 12 08:31:13 1998
Rango
I
X
10.0
7.5
-5.0
-7.5
-10.0
7.5
10.0
-10.000 < X < 10.000   -10.000 < I < 10.00
Jun 12 08:34:18 1998

```
/*         MAPEO CONFORME DOMINIO RECTANGULAR
           [J.P.H.E.             ABRIL-1994]
*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <time.h>
#include "plotp.h"
#include "ac.h"

#define  npu  29
double   mcwr[npu][npu], mcwi[npu][npu];

main()
{
  struct  complex  cz,  cw;
  double  mipr,  mapr,   mipi,  mapi,    pr,  pi, ex;
  double  x,   y,   dx,  dy;
  double  xmi, xma, ymi, yma;
  int     i,   j,   nci;
  char    bu[] ={"Mapeo Conforme  Curvas de Maxwell w = a/pi*(z+1+exp(z))"};
  char    bin[60];
  char    moin,     c;

  xmi  =   -9.0;       xma = 2.0;
  ymi  =   -3.14;      yma = 3.14;
  printf("\n\n\n                Mapeo Conforme\n\n");
  printf("\n    Intervalo en X %lf, %lf",xmi, xma);
  printf("\n    Intervalo en Y %lf, %lf", ymi, yma);
  moin = 'E';
  while(!(moin == 'S' || moin == 'N')) {
        printf("\n\n    Modificar los intervalos? (S/N) ");
        scanf("%c", &moin); moin = (islower(moin)) ? _toupper(moin) : moin;
        i=getchar();
  }
  if(moin == 'S') {
     printf("\n    Valor Minimo en X = ");   scanf("%lf", &xmi);
     printf("    Valor Maximo en X = ");     scanf("%lf", &xma);
     printf("    Valor Minimo en Y = ");     scanf("%lf", &ymi);
     printf("    Valor Maximo en Y = ");     scanf("%lf", &yma);
     i = getchar();                                  /* Para anular ENTER */
  }
  dx   = fabs(xma-xmi)/(double) (npu-1);
  dy   = fabs(yma-ymi)/(double) (npu-1);

  cz   = cmplx(xmi, ymi);
  cw = acmul(acsum(acexp(cz),acsum(cmplx(1.0,0.0),cz)),cmplx(1.0/6.28,0.0));
  mipr = cw.x;              mapr = mipr;
  mipi = cw.y;              mapi = mipi;
  x  =  xmi;
  for(i=0; i < npu; i++) {
      y  =  ymi;
      for(j=0; j < npu; j++) {
```

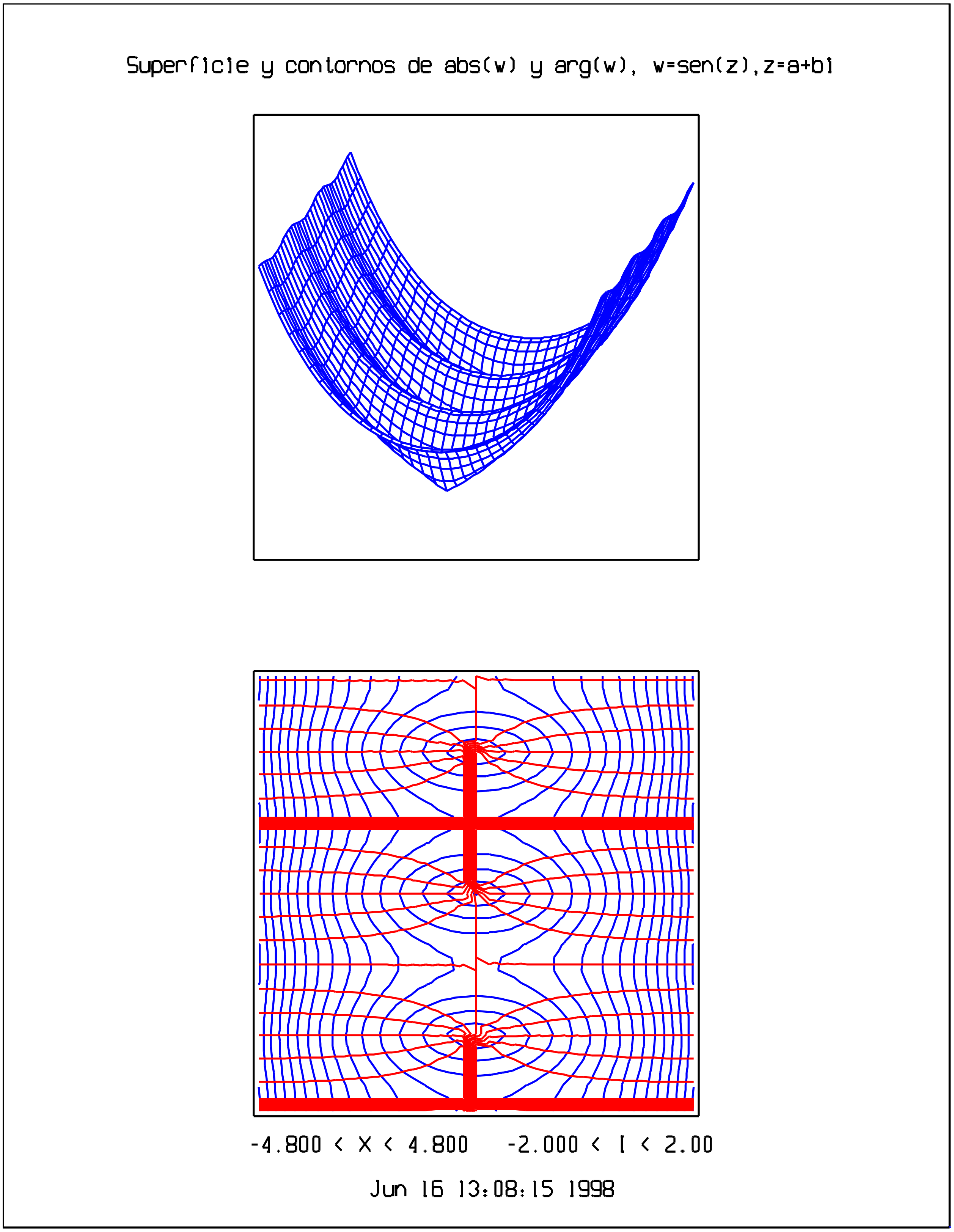
```
        cz  = cmplx(x, y);
        cw = acmul(acsum(acexp(cz),acsum(cmplx(1.0,0.0),cz)),cmplx(1.0/6.28,0.0));
        pr  = cw.x;
        pi  = cw.y;
        mcwr[i][j] = pr;
        mcwi[i][j] = pi;
        if(pr > mapr) mapr = pr;
        if(pr < mipr) mipr = pr;
        if(pi > mapi) mapi = pi;
        if(pi < mipi) mipi = pi;
        y  =  y + dy;
    }
    x  =  x + dx;
}

plt00();
plotz("V1 P1 ",6);
pltfr(1);
plot(2.0,  0.5, UP);                                /* Marco Superior */
plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);

plot(2.0,  -0.5, UP);                               /* Marco Inferior */
plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);


pltcolor(3);                  plotz("P2 ",3);
x = fabs(xma-xmi)/2.0;                                   /*  DOMINIO  */
y = fabs(yma-ymi)/2.0;
pltso(2.0, 2.0, 0.0,  2.5);
pltig((xmi-x) >= 0.0 ? -1.0 : xmi-x, (ymi-y) >= 0.0 ? -1.0 : ymi-y, 1, pltca);
pltig((xma+x) <= 0.0 ? 1.0 : xma+x, (yma+y) <= 0.0 ? 1.0 : yma+y, 2, pltca);
x  = xmi;
for(i=0; i < npu; i++) {
    y = ymi;
    pltig(x, y, 3, pltca);
    for(j=1; j < npu; j++) { y += dy; pltig(x, y, 4, pltca);}
    x += dx;
}
pltig(0.0, 0.0, 9, pltca);                                   /* Ejes */
nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
plttp(-2.0, -2.3, -0.11, bin, 0.0, nci, 1, pltmv);

y  = ymi;    pltcolor(4);    plotz("P3 ",3);
for(i=0; i < npu; i++) {
    x = xmi;
    pltig(x, y, 3, pltca);
    for(j=1; j < npu; j++) { x += dx; pltig(x, y, 4, pltca);}
    y += dy;
}
plotr();

x = fabs(mapr-mipr)/8.0;                                    /*  RANGO  */
```

```
 y = fabs(mapi-mipi)/8.0;
 pltcolor(3);   plotz("P2 ",3);
 pltso(2.0, 2.0, 0.0, -2.5);
 pltig((mipr-x) >= 0.0 ? -1.0 : mipr-x, (mipi-y) >= 0.0 ? -1.0 : mipi-y, 1, pltca);
 pltig((mapr+x) <= 0.0 ? 1.0 : mapr+x, (mapi+y) <= 0.0 ? 1.0 : mapi+y, 2, pltca);
 for(i=0; i < npu; i++) {
     pltig(mcwr[i][0],mcwi[i][0],3,pltca);
     for(j=1; j < npu; j++) pltig(mcwr[i][j],mcwi[i][j],4,pltca);
 }
 pltig(0.0, 0.0, 9, pltca);                                  /* Ejes */
 pltcolor(4);   plotz("P3 ",3);
 for(i=0; i < npu; i++) {
     pltig(mcwr[0][i],mcwi[0][i],3,pltca);
     for(j=1; j < npu; j++) pltig(mcwr[j][i],mcwi[j][i],4,pltca);
 }
 plotr();
 pltcolor(2);  plotz("P4 ",3);
 nci = pltgi(bin, mipr, 'X', mapr, mipi, 'I', mapi);
 pltlla(bu, (sizeof(bu)-1), bin, nci, 1);
 pltho();
 pltclose();
}
```


Mapeo Conforme Curvas de Maxwell w = a/pi*(z+1+exp(z))
I
X
5.0
3.8
2.5
1.3
-1.3
-2.5
-3.8
-5.0
-11.6 -8.7 -5.8 -2.9 1.5 3.0 4.5 6.0
-9.000 < X < 2.000 -3.140 < I < 3.14
1.4
1.1
0.7
0.4
-0.4
-0.7
-1.1
-1.4
-1.3 -1.0 -0.7 -0.3 0.4 0.8 1.2 1.6
-1.274 < X < 1.654 -1.433 < I < 1.43
Jun 16 12:23:47 1998

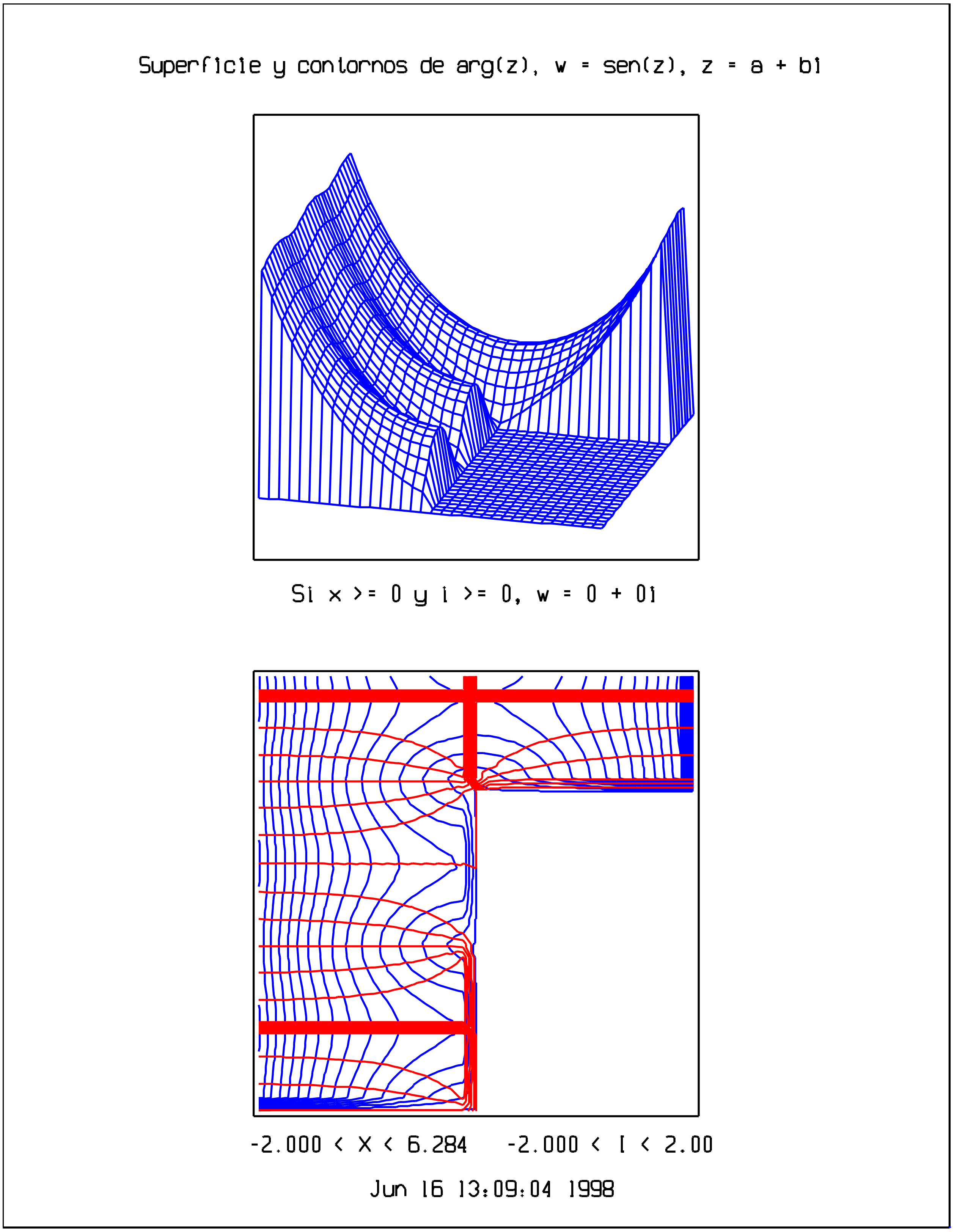
```
/* SUPERFICIE Y CONTORNOS DE ABS(W) Y ARG(W) de  W=SEN(Z), Z = a + bi
                     [J.P.H.E   JUL-1993]
*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "plotp.h"
#include "ac.h"

#define  NX   35            /* Dimension de las matrices de valores para el */
#define  NY   35            /* contorno o la superficie                     */
double  WY[NX][NY], WA[NX][NY];

main()
{
  int     i, j, nci;
  double  x, y, xmi, xma, ymi, yma;
  double  z, w, zmi, zma, zi, za, dx, dy;
  struct  complex cx, cy, cw;
  char    bu[]={"Superficie y contornos de abs(w) y arg(w), w=sen(z), z=a+bi"};
  char    bin[60];
  char    moin;

  xmi  = -4.8;       xma = 4.8;           /* Dominio de la funcion compleja */
  ymi  = -2.0;       yma = 2.0;

  for(;;) {
     printf("\n           Seno Complejo\n\n");

    printf("\n Intervalo en X %lf, %lf\n Intervalo en Y %lf, %lf",xmi,xma,ymi,yma);
    moin = 'E';
    while(!(moin=='S' || moin=='N')) {
          printf("\n\n Modificar los intervalos? (S/N) ");
          scanf("%c", &moin); moin = (islower(moin)) ? _toupper(moin) : moin;
          i=getchar();                                   /* Anular ENTER  */
    }
    if(moin=='S') {
       printf("\n Valor Minimo en X = ");       scanf("%lf", &xmi);
       printf(" Valor Maximo en X = ");         scanf("%lf", &xma);
       printf(" Valor Minimo en Y = ");         scanf("%lf", &ymi);
       printf(" Valor Maximo en Y = ");         scanf("%lf", &yma);
       i = getchar();                                    /* Anular ENTER  */
    }

    dx   = fabs(xma-xmi)/(double) (NX-1);
    dy   = fabs(yma-ymi)/(double) (NY-1);

    cw   = cmplx(xmi, ymi);
    cx   = acsin(cw);                                        /* Funcion */
    zmi  = cabs(cx);
    zi   = acarc(cx);
    zma  = zmi;
    za   = zi;
    y    = ymi;          /* Calculo de los valores de la funcion compleja */
```

```
    for(i=0; i < NY; i++) {
        x = xma;
        for(j=0; j < NX; j++) {
            cx = cmplx(x, y);
            cw = acsin(cx);                                     /* Funcion */
            z  = cabs(cw);                               /* Valor Absoluto */
            w  = acarc(cw);                              /* Fase           */
            WY[i][j] = (double) z;
            WA[i][j] = (double) w;
            zmi = (zmi > z) ? z   : zmi;  /* Calculo del valor min. y max */
            zma = (zma > z) ? zma : z;
            zi  = (zi  > w) ? w   : zi;
            za  = (za  > w) ? za  : w;
            x  -= dx;
        }
        y += dy;
    }
    x   = fabs(zma-zmi)/8.0;
    y   = fabs(za - zi)/8.0;
    zmi = zmi - x;     zma = zma + x;
    zi  = zi  - y;     za  = za  + y;

    plt00();
    plotz("V1 ",3);                                           /* DMP-29  */
    pltfr(1);
    plot(2.0,  0.5, UP);                               /*   Marco Superior  */
    plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
    plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);

    plot(2.0,  -0.5, UP);                              /*   Marco Inferior */
    plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
    plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);

    pltso(1.95, 1.95, 0.0,  2.5);
    pltrv(zmi, &WY[0][0], zma, NX, NY, -15.0, pltca);
    plotr();


    pltso(1.95, 1.95, 0.0, -2.5);

    pltkp(zmi, &WY[0][0], zma, 20, 2, NX, 2, NY, pltca);
    plotz("P2 ",3);
    pltkc(zi,  &WA[0][0], za,  20, 2, NX, 2, NY, pltca);
    plotr();


    plotz("P1 ",3);
    nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
    pltlla(bu, sizeof(bu)-1, bin, nci, 1);

    pltho();
    pltclose();
 }
}
```


Superficie y contornos de abs(w) y arg(w), w=sen(z),z=a+bi
-4.800 < X < 4.800   -2.000 < I < 2.00
Jun 16 13:08:15 1998

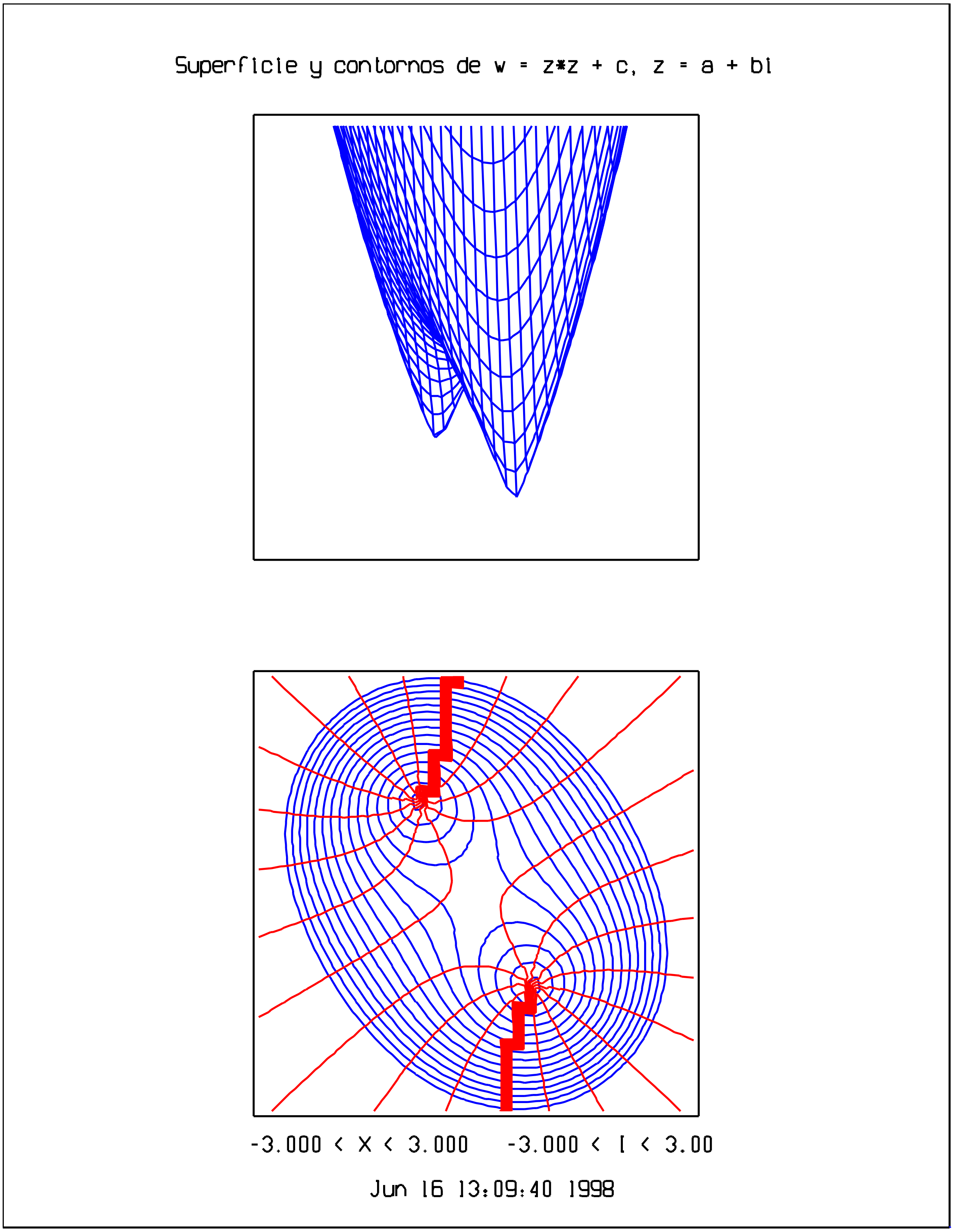
```
/* SUPERFICIE Y CONTORNOS  de  W=SEN(Z), Z = a + bi,   CON CORTE
                  [J.P.H.E.           JUL-1993]
*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "plotp.h"
#include "ac.h"
#define  NX   35              /* Dimension de las matrices de valores para */
#define  NY   35              /* el contorno o la superficie               */
double  WY[NX][NY], WA[NX][NY];

main()
{ int     i, j, nci;
  double  x, y, xmi, xma, ymi, yma;
  double  z, w, zmi, zma, zi, za, dx, dy;
  struct  complex cx, cy, cw;
  char    bu[] ={"Superficie y contornos de arg(z), w = sen(z), z = a + bi"};
  char    bu1[]={"Si x >= 0 y i >= 0, w = 0 + 0i"};
  char    bin[60];
  char    moin;

  xmi     = -2.0;       xma = 6.2840;    /* Dominio de la funcion compleja */
  ymi     = -2.0;       yma = 2.0;

  for(;;) {
     printf("\n           Seno Complejo\n\n");
    printf("\n Intervalo en X %lf, %lf\n Intervalo en Y %lf, %lf",xmi,xma,ymi,yma);
    moin = 'E';
    while(!(moin=='S' || moin=='N')) {
          printf("\n\n Modificar los intervalos? (S/N) ");
          scanf("%c", &moin); moin = (islower(moin)) ? _toupper(moin) : moin;
          i=getchar();
    }
    if(moin=='S') {
       printf("\n Valor Minimo en X = ");  scanf("%lf", &xmi);
       printf(" Valor Maximo en X = ");    scanf("%lf", &xma);
       printf(" Valor Minimo en Y = ");    scanf("%lf", &ymi);
       printf(" Valor Maximo en Y = ");    scanf("%lf", &yma);
       i = getchar();                                      /* Anular ENTER */
    }
    dx   = fabs(xma-xmi)/(double) (NX-1);
    dy   = fabs(yma-ymi)/(double) (NY-1);

    cw   = cmplx(xmi, ymi);
    cx   = acsin(cw);                                           /* Funcion */
    zmi  = cabs(cx);
    zi   = acarc(cx);
    zma  = zmi;
    za   = zi;
    y    = ymi;          /* Calculo de los valores de la funcion compleja */
    for(i=0; i < NY; i++, y+=dy) {
        for(j=0, x=xma; j < NX; j++, x-=dx) {
```

```
            cx = cmplx(x, y);
            cw = acsin(cx);                                   /* Funcion */
            z  = cabs(cw);                              /* Valor Absoluto */
            w  = acarc(cw);                             /* Fase           */
            if( x >= 0.0 && y >= 0.0 ) { z = 0.0; w = 0.0; }     /* CORTE */
            WY[i][j] = (double) z;
            WA[i][j] = (double) w;
            zmi = (zmi > z) ? z   : zmi;  /* Calculo del valor min. y max */
            zma = (zma > z) ? zma : z;
            zi  = (zi  > w) ? w   : zi;
            za  = (za  > w) ? za  : w;
        }
    }

    for(i = 0; i < NY; i++) WY[i][0]    = 0.0;       /* Paredes laterales */
    for(i = 0; i < NX; i++) WY[NY-1][i] = 0.0;

    x   = fabs(zma-zmi)/8.0;             /* Ajuste del valor min. y max. */
    y   = fabs(za - zi)/8.0;
    zmi = zmi - x;     zma = zma + x;
    zi  = zi  - y;     za  = za  + y;

    plt00();
    plotz("V1 ",3);                   /* DMP-29  */
    pltfr(1);
    plot(2.0,  0.5, UP);                             /*   Marco Superior  */
    plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
    plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);
    plot(2.0,  -0.5, UP);                            /*   Marco Inferior */
    plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
    plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);

    pltso(1.95, 1.95, 0.0,  2.5);
    pltrv(zmi, &WY[0][0], zma, NX, NY, -15.0, pltca);
    plotr();


    pltkp(zmi, &WY[0][0], zma, 20, 2, NX, 2, NY, pltca);
    pltso(1.95, 1.95, 0.0, -2.5);

    plotz("P2 ",3);
    pltkc(zi,  &WA[0][0], za,  20, 2, NX, 2, NY, pltca);
    plotr();


    plotz("P1 ",3);
    nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
    pltlla(bu, sizeof(bu)-1, bin, nci, 1);
    plttp(-1.6, -2.3, -0.11, bu1, 0.0, sizeof(bu1)-1, 1, pltmv);
    pltho();
    pltclose();
 }
}
```


Superficie y contornos de arg(z), w = sen(z), z = a + bi
Si x >= 0 y i >= 0, w = 0 + 0i
-2.000 < X < 6.284
-2.000 < I < 2.00
Jun 16 13:09:04 1998

```
/*  SUPERFICIE Y CONTORNOS DE W = Z*Z + C,   Z = a + bi  (PLTRV, PLTKP y PLTKC)
    [J.P.H.E.   JUL-1993]
*/
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include "plotp.h"
#include "ac.h"

#define  NX   37          /* Dimension de las matrices de valores para el */
#define  NY   37          /* contorno o la superficie                     */
double   WY[NX][NY], WA[NX][NY];


main()
{
  int     i, j, nci;
  double  x, y, xmi, xma, ymi, yma;
  double  z, w, zmi, zma, zi, za, dx, dy;
  struct  complex cx, cy, cw;
  char    bu[] ={"Superficie y contornos de w = z*z + c, z = a + bi "};
  char    bin[60];

  xmi     = -3.0;       xma = 3.0;    /* Dominio de la funcion compleja */
  ymi     = -3.0;       yma = 3.0;

  for(;;) {
     printf("\n         w = z*z + c\n\n");

    printf("\n Intervalo en X %lf, %lf\n Intervalo en Y %lf, %lf",xmi,xma,ymi,yma);
    moin = 'E';
    while(!(moin=='S' || moin=='N')) {
          printf("\n\n Modificar los intervalos? (S/N) ");
          scanf("%c", &moin); moin = (islower(moin)) ? _toupper(moin) : moin;
          i=getchar();
    }
    if(moin=='S') {
       printf("\n Valor Minimo en X = ");        scanf("%lf", &xmi);
       printf(" Valor Maximo en X = ");          scanf("%lf", &xma);
       printf(" Valor Minimo en Y = ");          scanf("%lf", &ymi);
       printf(" Valor Maximo en Y = ");          scanf("%lf", &yma);
       i = getchar();                                     /* Para anular ENTER */
    }
    dx   = fabs(xma-xmi)/(double) (NX-1);
    dy   = fabs(yma-ymi)/(double) (NY-1);

    cy   = cmplx(1.0, 2.0);
    cw   = cmplx(xmi, ymi);
    cx   = acsum(acmul(cw, cw), cy);                                /* Funcion */
    zmi  = cabs(cx);
    zi   = acarc(cx);
    zma  = zmi;
    za   = zi;
```

```
    x    = xmi;             /* Calculo de los valores de la funcion compleja */
    for(i=0; i < NY; i++) {
        y = xmi;
        for(j=0; j < NX; j++) {
            cx = cmplx(x, y);
            cw = acsum(acmul(cx, cx), cy);                       /* Funcion */
            z  = cabs(cw);                                /* Valor Absoluto */
            w  = acarc(cw);                                         /* Fase */
            WY[i][j] = (double) z;
            WA[i][j] = (double) w;
            zmi = (zmi > z) ? z   : zmi;    /* Calculo del valor min. y max */
            zma = (zma > z) ? zma : z;
            zi  = (zi  > w) ? w   : zi;
            za  = (za  > w) ? za  : w;
            y  += dy;
        }
        x += dx;
    }
    x   = fabs(zma-zmi)/8.0;                /* Ajuste del valor min. y max. */
    y   = fabs(za - zi)/8.0;
    zmi = zmi - x;     zma = zma - 5.0*x;
    zi  = zi  - y;     za  = za  - 5.0*y;

    plt00();
    plotz("V1 ",3);                                              /* DMP-29  */
    pltfr(1);
    plot(2.0,  0.5, UP);                                /*   Marco Superior  */
    plot(-2.0, 0.5, DOWN);      plot(-2.0, 4.5, DOWN);
    plot(2.0,  4.5, DOWN);      plot(2.0,  0.5, DOWN);

    plot(2.0,  -0.5, UP);                               /*   Marco Inferior */
    plot(2.0,  -4.5, DOWN);     plot(-2.0, -4.5, DOWN);
    plot(-2.0, -0.5, DOWN);     plot(2.0,  -0.5, DOWN);


    pltso(1.95, 1.90, 0.0,  2.5);
    pltrv(zmi, &WY[0][0], zma, NX, NY, -8.0, pltca);
    plotr();

    pltso(1.95, 1.95, 0.0,  -2.5);
    pltkp(zmi, &WY[0][0], zma, 20, 2, NX, 2, NY, pltca);
    plotz("P2 ",3);
    pltkc(zi,  &WA[0][0], za,  20, 2, NX, 2, NY, pltca);
    plotr();

    nci = pltgi(bin, xmi, 'X', xma, ymi, 'I', yma);
    pltlla(bu, sizeof(bu)-1, bin, nci, 1);
    pltho();
    pltclose();
  }
}
```



Superficie y contornos de w = z*z + c, z = a + bi
-3.000 < X < 3.000
-3.000 < I < 3.00
Jun 16 13:09:40 1998

Bibliografía

[1] Licona, Alberto, Silvestre Angoa, Miguel Angel Burgos *Las Gráficas como Complemento del Lenguaje Oral y Escrito* ELEMENTOS, Año 2, Vol. 2, No. 8, Julio-Septiembre de 1986, UAP Puebla, México.

[2] McIntosh, Harold V. *PLOT75* INEN Salazar, Edo. de México, México.

[3] Beebe, Nelson H. F. *A User's Guide to* $< PLOT79 >$ Departments of Physics and Chemistry, University of Utah Salt Lake City, Utah, USA.

[4] Gil Zepeda, Francisco y Harold V. McIntosh *PLOT84* Departamento de Aplicación de Microcomputadoras, ICUAP, UAP Puebla, México.

Indice