FISICA COMPUTACIONAL

UNIVERSIDAD DE MURCIA
Miguel Albaladejo Serrano
Licenciatura en Física
mas4@alu.um.es

Índice general

Índice de figuras

Tema 0

Introducción a la programación en FORTRAN

0.10. Frecuencia de radiación en una transición electrónica

Se trata de escribir un programa que permita calcular las frecuencias de la radiación emitida por una transición electrónica entre dos estados con números cuánticos n_1 y n_2, que viene dada por:

$$\nu = \frac{me^4Z^2}{8\varepsilon_0^2 h^3}\left(\frac{1}{n_1} - \frac{1}{n_2}\right) \tag{1}$$

donde Z es el número atómico, m y e son la masa y la carga del electrón, ε_0 es la permitividad del vacío y h es la constante de Planck (no racionalizada). Según el valor de n_1, las series tienen distintos nombres.

Lo primero que hay que pensar es si la constante que aparece al principio puede generar problemas de `underflow` o de `overflow`. Quizá con otro compilador sí, pero este que usamos nosotros está preparado para dar preferencia a los cálculos para evitar problemas de este tipo.

Para generar los valores deseados de las frecuencias, declaramos varias variables dimensionales (mediante `dimension`) para representar los distintos valores que puede tomar n_2 (`N2(I)`), así como el valor de la frecuencia ν (`V(I)`) asociada a n_2, una vez que se fija n_1 (mediante la constante `N1`). Esto se realiza en las líneas (5-6). Lo siguiente que debemos hacer es definir todas las constantes que aparecen en la ecuación (1), y para ello se escriben las líneas (8-12).

En las líneas (13-25) se realiza la presentación, en la que se escribe una cabecera y se presentan los distintos nombres de las series, según el estado energético inicial (Lyman, Balmer, etc...). Se da entonces al usuario la posibilidad de elegir una serie concreta o una "sin nombre". Lo único que se hace, en realidad es leer (`read`) el valor de `N1`, así como, después, los valores de `Z` y `N2MAX`, que representa el nivel energético máximo al que se quiere llegar (líneas (26-34)). A continuación se inicia un bloque condicional `if`

para asegurarnos de que los valores dados cumplen la restricción $n_2 > n_1$, y también que $n_{2,max} < 50$, ya que ese es el valor máximo que hemos establecido (es un valor bastante grande).

Las líneas (45-49) constituyen el núcleo central del programa, pues en ellas se hacen los cálculos de los valores de ν para cada nivel n_2. Es importante notar que, tal como está definida la variable dimensional `N2(I)`, debemos empezar a hacer los cálculos (el ciclo `do`) a partir de `I = N1 +1` hasta `N2MAX`. Una vez que el ciclo ha terminado, ya tenemos toda una colección de parejas de valores (n_2,ν).

Por otro lado, sobre la finalidad del programa, vamos a intentar que éste proporcione los resultados de tres formas:

- Escribiéndolos en pantalla. Esto se consigue con las líneas (53) y (56), que escriben la cabecera, y la (60), esta dentro de un ciclo para escribir toda la serie de datos.
- Escribiendo todos los valores de n_2 y ν para un determinado átomo (Z) y un determinado estado inicial (n_1). El archivo se abre en la línea (51) y se cierra en (64). La escritura se realiza con las líneas (54), (57) y (61). Todo ello se escribe en el archivo *0-10-mas.dat*.
- Escribiendo muchas series de datos, correspondientes a distintos valores del estado inicial (n_1) y para distintos átomos.Esto permitirá representar comparativamente distintas series de datos. Para ello, aunque el archivo se abre justo antes del bucle que escribe los datos, línea (52), no se cierra hasta que se termina el programa, mientras que el archivo anterior se cerraba justo después del bucle. Este otro archivo se llama *0-10ii-mas.dat*.

El hecho de poder escribir varias series de datos se debe a que no cerramos el programa nada más terminar el bloque `do`, sino que, en las líneas (83-93), se ofrece la posibilidad de terminar o reiniciar el programa, con un bloque condicional `if`. Así, si decidimos continuar, el programa se reinicia, y el archivo (2) sigue abierto, con lo cual se escriben más datos detrás de los anteriores. El archivo (1), como ya ha sido cerrado, es borrado y vuelto a escribir. Una variante sería dejar solo un archivo, ya que rara vez pueden buscarse los dos objetivos que se persiguen con los archivos (1) y (2) a la vez.

Por último, comentar que en las líneas (67-81) se describen los diferentes formatos que se utilizan, y que los archivos obtenidos, *0-10-mas.dat* y/o *0-10ii-mas.dat* pueden ser graficados con GNUPLOT, obteniéndose los gráficos dados por *0-10-mas.plt* y/o *0-10ii-mas.plt* respectivamente. A continuación se ofrece un listado completo del código FORTRAN.

```
1  C     0-10-mas.f
2  C     Vamos a escribir un programa para calcular las frecuencias de la
3  C     la radiacion en las transiciones electronicas en el atomo
4  C     Miguel Albaladejo Serrano 20/02/04 17:20
```

```
5         DIMENSION N2(50),V(50)
6         REAL*8 N1,N2,N2MAX,ELM,HPLANCK,V,Q,EPSI0,C,Z
7
8         ELM=9.109534D-31
9         HPLANCK=6.626176D-34
10        Q=1.602189D-19
11        EPSI0=8.854D-12
12        C=(ELM*Q**4)/(8*(EPSI0**2)*(HPLANCK**3))
13     10 WRITE(*,100) !Presentacion
14        WRITE(*,150) !Presentacion
15        WRITE(*,100) !Presentacion
16        WRITE(*,*)'
17       &
18       &                                                              '
19  C     Aqui se dicen los nobres de la serie
20        WRITE(*,310)
21        WRITE(*,320)
22        WRITE(*,330)
23        WRITE(*,340)
24        WRITE(*,350)
25        WRITE(*,360)
26  C     Input de datos necesarios: N1, N2MAX, Z
27        WRITE(*,200)
28        READ(*,*)N1
29
30        WRITE(*,210)
31        READ(*,*)Z
32
33        WRITE(*,220)
34        READ(*,*)N2MAX
35  C     Calculos
36        IF (N2MAX.LE.N1) THEN
37            WRITE(*,500)
38            GOTO 600
39          ELSEIF (N2MAX.GE.50) THEN
40            WRITE(*,*)'El programa esta preparado solo para N2MAX < 51'
41          ELSE
42            CONTINUE
43        ENDIF
44  C     Constante del calculo
45        C=(ELM*Q**4)/(8*(EPSI0**2)*(HPLANCK**3))
46        DO I=N1 + 1 , N2MAX
47           N2(I) = I
48           V(I) = C*(Z**2)*(1/N1-(1/N2(I)))
49        END DO
50  C     Escribe los calculos, en pantalla y en dos ficheros
```

```
51        OPEN(1,FILE='0-10-mas.dat')
52        OPEN(2,FILE='0-10ii-mas.dat')
53        WRITE(*,390)INT(Z),INT(N2MAX),INT(N1)
54        WRITE(1,390)INT(Z),INT(N2MAX),INT(N1)
55        WRITE(2,390)INT(Z),INT(N2MAX),INT(N1)
56        WRITE(*,400)
57        WRITE(1,400)
58        WRITE(2,400)
59        DO K=N1 + 1, N2MAX
60           WRITE(*,410)NINT(N2(K)),V(K)
61           WRITE(1,410)NINT(N2(K)),V(K) !Se cierra. Representacion
62           WRITE(2,410)NINT(N2(K)),V(K) !No se cierra. Para datos
63        END DO
64        CLOSE(1)
65
66
67   100  FORMAT(15x,'************************************************')
68   150  FORMAT(15x,'** FRECUENCIAS DE LAS TRANSICIONES ELECTRONICAS **')
69   200  FORMAT(10x,'ÍQue serie quieres? (N1 =)')
70   210  FORMAT(10x,'ÍQue atomo quieres? (Z =)')
71   220  FORMAT(10x,'ÍHasta que nivel quieres? (N2MAX =)')
72   310  FORMAT(10x,'Serie de LYMAN (N1 = 1)')
73   320  FORMAT(10x,'Serie de BALMER (N1 = 2)')
74   330  FORMAT(10x,'Serie de PASCHEN (N1 = 3)')
75   340  FORMAT(10x,'Serie de BRACKETT (N1 = 4)')
76   350  FORMAT(10x,'Serie de PFUND (N1 = 5)')
77   360  FORMAT(10x,'Otra (N1 > 5)')
78   390  FORMAT('#',3x,'Z=',I4,4x,'N2MAX=',I3,4x,'N1=',I3)
79   400  FORMAT('#',7x,'N2',8x,'frecuencia')
80   410  FORMAT(8x,I2,8x,D9.3)
81   500  FORMAT(16x,'ESO NO TIENE MUCHO SENTIDO, NO?')
82  C     Fin de programa
83   600  WRITE(*,*)'PROGRAMA TERMINADO'
84        WRITE(*,*)'ÍQUIERES CONTINUAR? (1==SI) (2==NO)'
85        READ(*,*)CONT
86        IF (INT(CONT).EQ.1) THEN
87           GOTO 10
88           ELSEIF (INT(CONT).EQ.2) THEN
89                  GOTO 1000
90           ELSE
91                  GOTO 600
92           ENDIF
93        CLOSE(2)
94   1000 END
```

0.11. Potencial de Lennard-Jones

En este problema se estudia el potencial de Lenard-Jones, que describe la interacción entre dos átomos de un gas noble, para los casos del Ne, Ar, Kr y Xe. Dicha energía potencial de interacción viene dada por:

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] \tag{2}$$

donde ε y *sigma* son parámetros de cada gas, cuyo significado hay que deducir y estudiar. Dichos parámetros son:

	Ne	Ar	Kr	Xe
ε (eV)	0.0031	0.0104	0.0140	0.0200
σ	2.64	3.40	3.65	3.98

Cuadro 1: Valores de las constantes para los gases

Lo primero que hacemos es graficar dichos potenciales (con GNUPLOT) para los cuatro gases, teniendo en cuenta que vamos a tener problemas de escala: las energías son claramente divergentes cerca del origen: $\lim_{r\to 0} V(r) = +\infty$. Como este comportamiento es fácil de ver, sin necesidad de representar la energía, lo que haremos será limitar la escala a valores de la energía que estén por debajo de un cierto límite que nos interese (este valor será ε). Haciendo esto, observamos la forma del potencial dado por la representación (ver figura 1). De esta representación podemos deducir varias cosas (algunas de ellas deducibles con solo observar la forma analítica del potencial), como que $\lim_{r\to\infty} V(r) = 0^-$. Además, podemos darnos cuenta de lo que representan matemáticamente ε y σ: está claro que σ es el valor donde el potencial se hace cero, se anula. Por contra, $-\varepsilon$ representa un mínimo en la energía de interacción. Estas dos cosas se pueden también comprobar analíticamente:

$$\lim_{r\to\sigma} V(r) = 4\varepsilon \left[\left(\frac{\sigma}{\sigma}\right)^{12} - \left(\frac{\sigma}{\sigma}\right)^{6} \right] = 4\varepsilon(1-1) = 0 \tag{3}$$

$$\frac{\mathrm{d}V(r)}{\mathrm{d}r} = \frac{4\varepsilon}{r} \left[\left(-12 \left(\frac{\sigma}{r}\right)^{12} + 6 \left(\frac{\sigma}{r}\right) \right)^{6} \right] = 0 \Leftrightarrow \left(\frac{\sigma}{r}\right)^{6} = \frac{1}{2} \Leftrightarrow \boxed{r = \sigma\sqrt[6]{2}} \tag{4}$$

Sustituyendo en la expresión de $V(r)$ se obtiene:

$$V(r)|_{r=\sigma\sqrt[6]{2}} = 4\varepsilon \left[\frac{1}{4} - \frac{1}{2} \right] = -\varepsilon \tag{5}$$

Así pues, σ representa la distancia mínima entre los átomos cuando la energía potencial de interacción es menor que cero (cuando hay unión). El valor de $-\sigma$ indica el valor de la energía potencial a partir del cuál esta pasa de ser repulsiva a ser atractiva.

También hicimos un pequeño programa (apenas 30 líneas) en FORTRAN para calcular el potencial en cada punto para cada uno de los gases. Primero declaramos las variables dimensionales y parámetros a usar (líneas (5-10)). El parámetro `N` nos indicará el número de puntos, las variables `EPSI` y `SIGMA` representarán las constantes de cada gas, y `R(I) POTEN(I)` serán la posición y el potencial en dicho punto. Abrimos varios archivos (líneas (11-14)), en los que guardaremos los datos correspondientes a cada gas. Entonces realizamos dos ciclos anidados: el primero indicará el gas sobre el que se están haciendo los cálculos, mientras que el segundo realiza las `2N` operaciones necesarias, y escribe los datos en el archivo adecuado. A continuación se incluye un listado del código.

```
1  C NOMBRE: 0-11-mas.f
2  C AUTOR: Miguel Albaladejo
3  C DESCRIPCION: Energia interaccion atomos gas noble (Lennard-Jones)
4  C FECHA: 22/02/04 16:03
5        PARAMETER(N=10000)
6        DIMENSION R(N), EPSI(4), SIGMA(4),DR(4),POTEN(N)
7  C     Definiciones de los datos epsilon y sigma
8  C     1 es Ne, 2 es Ar, 3 es Kr, 4 es Xe
9        DATA EPSI/0.0031,0.0104,0.014,0.02/SIGMA/2.64,3.4,3.65,3.98/
10 C     Archivos donde se guardaran los datos
11       OPEN(1,FILE='0-11.mas-Ne.dat',STATUS='UNKNOWN')
12       OPEN(2,FILE='0-11.mas-Ar.dat',STATUS='UNKNOWN')
13       OPEN(3,FILE='0-11.mas-Kr.dat',STATUS='UNKNOWN')
14       OPEN(4,FILE='0-11.mas-Xe.dat',STATUS='UNKNOWN')
15 C     Ahi van los calculos...
16         DO I=1, 4 !Contador para los gases
17           DR(I) = 2.1*SIGMA(I)/REAL(N)
18           WRITE(I,50)EPSI(I),SIGMA(I),DR(I),N
19           DO J=1, N !Contador para los puntos
20           R(J) = 0.9*SIGMA(I) + REAL(J)*DR(I)
21           POTEN(J) = 4.0*EPSI(I)*((SIGMA(I)/R(J))**12 -
22      &       (SIGMA(I)/R(J))**6)
23           WRITE(I,100)R(J),POTEN(J)
24         END DO
25       CLOSE(I)
26       END DO
27    50 FORMAT('#',3X,'EPSILON = ',E9.3,4X,'SIGMA = ',E9.3,4X,'DR = ',
28      &E9.3,5X,'NUMERO DE PUNTOS N = ',I7)
29   100 FORMAT(6X,E9.3,6X,E9.3)
30       END
31
32
33
34
```

0.12. Regresión lineal por mínimos cuadrados

El objetivo de este problema es preparar un programa para calcular los parámetros del ajuste de n datos (x_i, y_i) a una recta $y = ax + b$. Las fórmulas necesarias son:

$$a = \frac{1}{D}\sum_{i=1}^{n}(x_i - \langle x\rangle)y_i \tag{6a}$$

$$b = \langle y\rangle - a\langle x\rangle \tag{6b}$$

$$\langle x\rangle = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6c}$$

$$\langle y\rangle = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{6d}$$

$$\varepsilon_a \approx \sqrt{\frac{1}{D}\frac{\sum_{i=1}^{n} d_i^2}{n-2}} \tag{6e}$$

$$\varepsilon_b \approx \sqrt{\left(\frac{1}{n} + \frac{\langle x\rangle^2}{D}\right)\frac{\sum_{i=1}^{n} d_i^2}{n-2}} \tag{6f}$$

$$d_i = y_i - ax_i - b \tag{6g}$$

$$D = \sum_{i=1}^{n}(x_i - \langle x\rangle)^2 \tag{6h}$$

Para compactar más los cálculos, hemos hecho las siguientes agrupaciones:

$$D = \sum_i {x_i}^2 - \frac{\left(\sum_i x_i\right)^2}{n} \tag{7a}$$

$$\langle x\rangle = \frac{1}{n}\sum_i x_i \tag{7b}$$

$$\langle y\rangle = \frac{1}{n}\sum_i y_i \tag{7c}$$

$$a = \frac{1}{D}\left(\sum_i x_i y_i - \langle x\rangle \sum_i y_i\right) \tag{7d}$$

$$b = \langle y\rangle - a\langle x\rangle \tag{7e}$$

$$y_{teo,i} = ax_i - b \tag{7f}$$

$$d_i = y_i - y_{teo,i} \tag{7g}$$

Algunas fórmulas de las que están en el párrafo anterior –y otras que no están– no las hemos cambiado respecto de las originales. Todas estas fórmulas, recordemos, han sido

introducidas para reducir el código FORTRAN al mínimo, y dan lugar a las fórmulas que están diseminadas a lo largo de todo el código (todo esto en las líneas (7-24))

La estructura del programa es simple: comienza con la declaración de los vectores, la inicialización de los sumatorios a cero, la apertura de los archivos que se leerán/escribirán y la lectura de los datos del archivo *datos_ exp.in*. La estructura posterior es muy mecánica: ciclo `DO` de cálculo, (25-30) (los sumatorios `SUMX`, `SUMXX`, `SUMXY` y `SUMY`), asignación de valores, (32-36) (`D`,`XMED`,`YMED`, `A` y `B`), otro ciclo de cálculos, (38-42) (`YT`,`DI` y `SUMDI`), y la asignación de valores de los errores, (44-45) (`EA` y `EB`). Los nombres de las variables son evidentes. `YT(I)` son los valores teóricos que se obtienen con los parámetros a y b de la recta para cada valor x_i.

Después se imprimen, con las líneas (46-49), los valores de x_i (`X(I)`) e $y_{teo,i}$ (`YT(I)`) en un archivo *datos_ ajuste.out*. Esto permitirá representar los datos experimentales y los teóricos con GNUPLOT (esto se encuentra realizado en el archivo *0-12-mas.plt*, de modo que los datos teóricos aparecen con una línea (ya que es una recta)). Una vez escritos los datos en el fichero, se imprime la ecuación de la recta en pantalla con los valores a, b, ε_a y ε_b (dados respectivamente por `A`, `B`, `EA`, y `EB`).

Las últimas líneas solo contienen adornos y formatos.

En la figura 2 podemos ver un ejemplo de representación mediante GNUPLOT los datos de un ajuste hecho con este programa de FORTRAN.

A continuación incluimos el código utilizado.

```
1  C
2  C NOMBRE: 0-12-mas.f
3  C AUTOR: Miguel Albaladejo Serrano
4  C DESCRIPCION: Regresion lineal de n parejas de datos
5  C FECHA: 24/02/04 22:18
6  C
7        REAL X,Y,D
8        DIMENSION X(10000),Y(10000),YT(10000),DI(10000)
9        SUMX = 0.0
10       SUMY = 0.0
11       SUMXX = 0.0
12       SUMXY = 0.0
13       SUMDI = 0.0
14       WRITE(*,100)
15       WRITE(*,200)
16       WRITE(*,250)
17       OPEN(1,FILE='datos_exp.in',STATUS='OLD')
18       OPEN(2,FILE='datos_ajuste.out',STATUS='UNKNOWN')
19       READ(*,*)N
20
21       DO I = 1, N
22         READ(1,*)X(I),Y(I)
```

```
23        END DO
24        CLOSE(1)
25        DO I = 1, N
26           SUMX = SUMX + X(I)
27           SUMY = SUMY + Y(I)
28           SUMXX = SUMXX + X(I)**2
29           SUMXY = SUMXY + X(I)*Y(I)
30        END DO
31
32           D = SUMXX - (SUMX**2)/REAL(N)
33           XMED = SUMX/REAL(N)
34           YMED = SUMY/REAL(N)
35           A = (SUMXY - XMED*SUMY)/D
36           B = YMED - A*XMED
37
38        DO I=1,N
39           YT(I) = A*X(I) + B
40           DI(I) = Y(I) - YT(I)
41           SUMDI = SUMDI + (DI(I))**2
42        END DO
43
44           EA = SQRT((1.0/D)*(SUMDI/(REAL(N)-2.0)))
45           EB = SQRT(((1.0/REAL(N))+(XMED**2/D))*(SUMDI/(REAL(N)-2.0)))
46        WRITE(2,350)N
47        DO I =1, N
48        WRITE(2,300)X(I),YT(I)
49        END DO
50
51        WRITE(*,*)'La recta es: y =(',A,'+-',EA,')','x + (',B,'+-',EB,')'
52   100  FORMAT (20X,'AJUSTE DE PARES DE DATOS A UNA RECTA')
53   200  FORMAT (20X,36('*'))
54   250  FORMAT (10X,'dime el numero EXACTO de datos N=')
55   300  FORMAT (F9.3,1X,F9.3)
56   350  FORMAT ('#',' Ajuste de',I3,' datos a una recta. Datos y para los'
57       &'puntos x # obtenidos a partir de la recta del ajuste')
58        CLOSE(2)
59        END
```

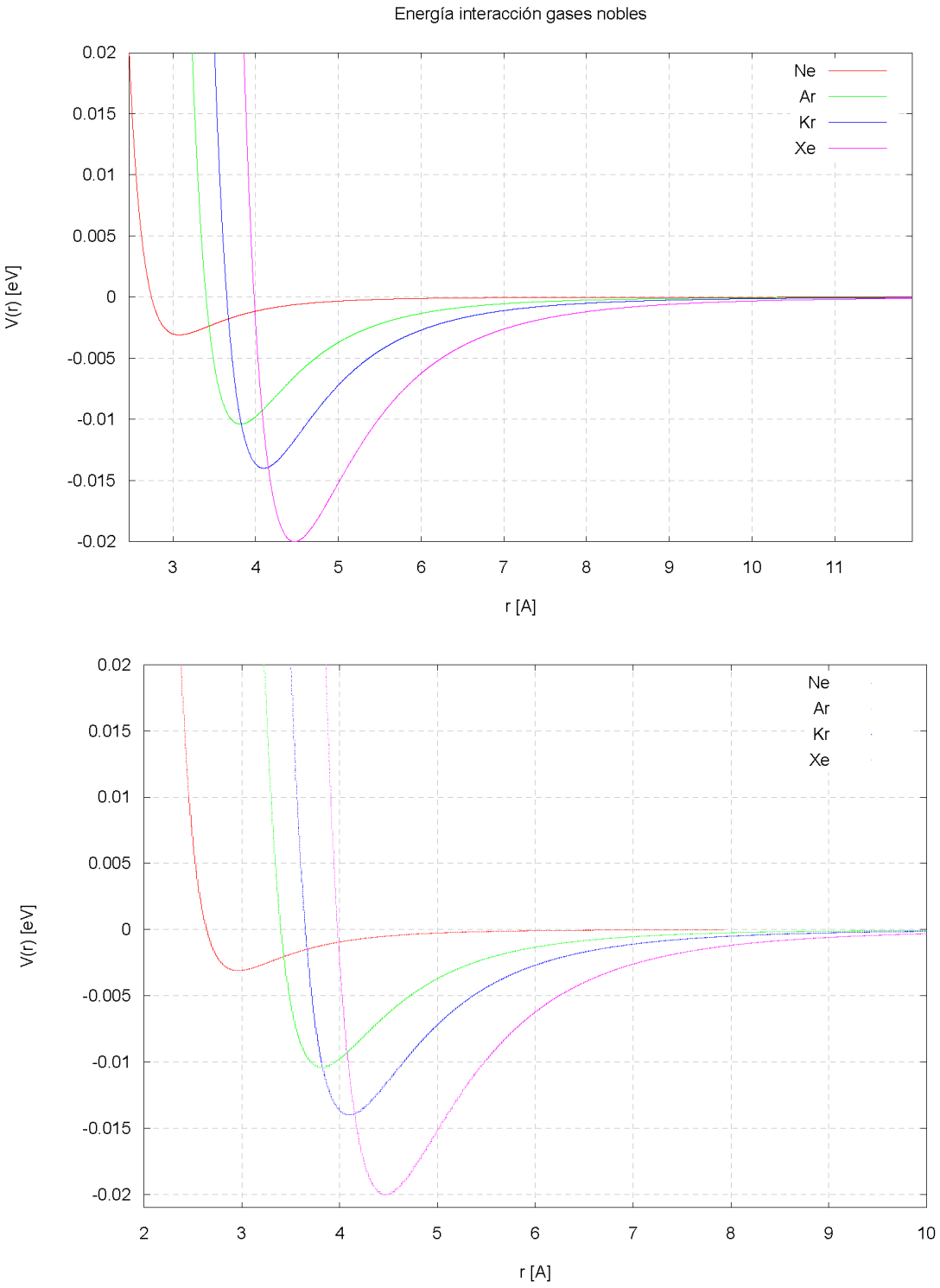

Figura 1: Representación analítica (arriba) y mediante puntos (abajo) de la energía potencial de interacción de Lenard-Jones

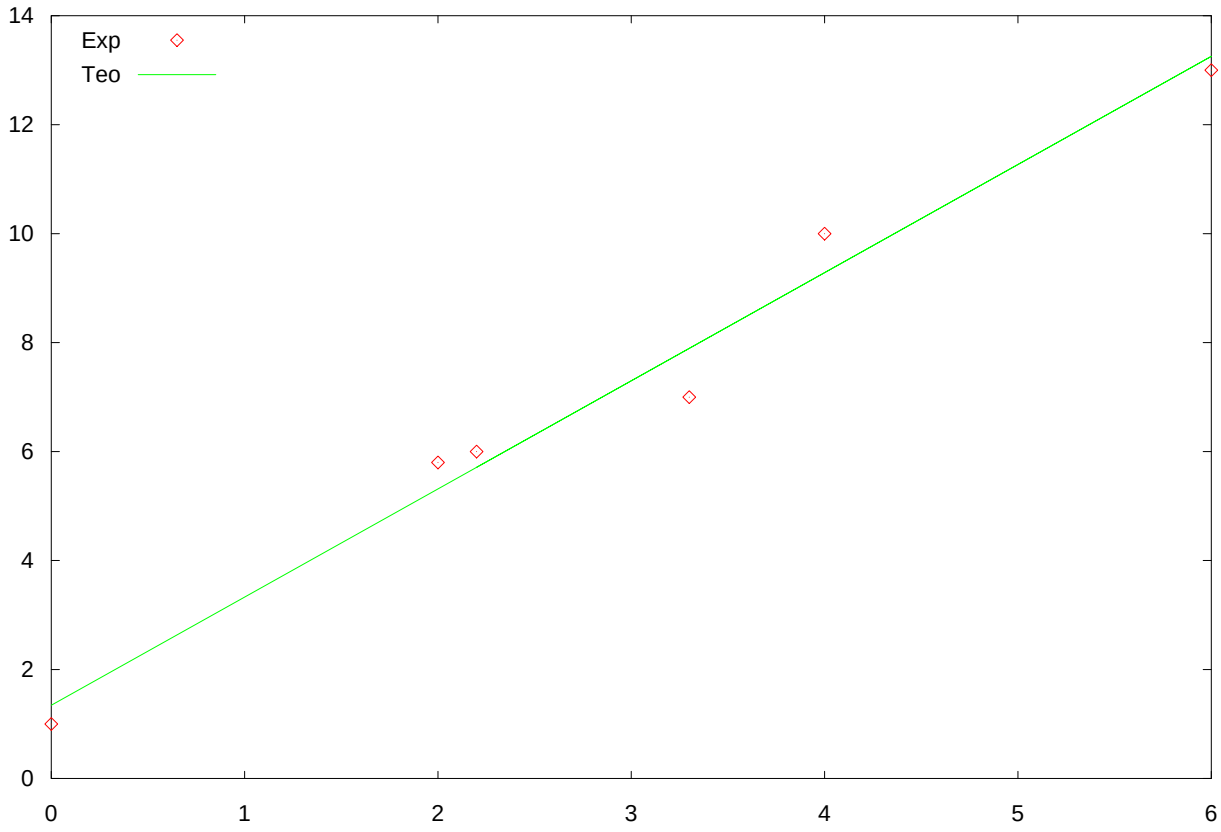

Figura 2: Ejemplo de una gráfica de GNUPLOT a partir del ajuste obtenido con el programa

Tema 1

Ceros de una función

1.6. Ecuación de van der Waals. Coordenadas críticas

1.6.1. Gráficas y cálculos analíticos

La ecuación de estado de VAN DER WAALS para un mol de gas es:

$$\left(p + \frac{a}{v^2}\right)(v - b) = RT \tag{1.1}$$

Donde a y b (covolumen) son parámetros propios de cada gas. Por ejemplo, en la figura 1.2 se puede observar las distintas isotermas que se obtienen para el CO_2, cuyos parámetros son $a = 8.856 \cdot 10^{-4} \text{Jm}^3 \cdot \text{mol}^2 = 3.59 \frac{\text{atmL}^2}{\text{mol}^2}$ y $b = 0.043 \frac{\text{L}}{\text{mol}}$. En la figura 1.1 se representan también las isotermas correspondientes al gas N_2, de parámetros $a == 1.390 \frac{\text{atmL}^2}{\text{mol}^2}$ y $b = 0.03913 \frac{\text{L}}{\text{mol}}$. Los valores de las coordenadas críticas se muestran en la tabla .

La deducción de las coordenadas críticas de forma analítica es sencilla, partiendo del hecho de que en (p_c, v_c) se tiene un punto de tangente horizontal, es decir, se cumple:

$$\left.\frac{\partial p}{\partial v}\right|_{v_c,\ T_c} = \left.\frac{\partial^2 p}{\partial v^2}\right|_{v_c,\ T_c} = 0 \tag{1.2}$$

Esto implica que:

$$\begin{aligned} \frac{\partial p}{\partial v} &= -\frac{RT}{(v-b)^2} + \frac{2a}{v^3} = 0 \\ \frac{\partial^2 p}{\partial v^2} &= -\frac{2RT}{(v-b)^3} - \frac{6a}{v^4} = 0 \end{aligned}$$

De las anteriores ecuaciones, mediante combinaciones adecuadas, se llega a:

$$RT\left(-\frac{3}{v(v-b)^2} + \frac{2}{(v-b)^3}\right) = 0 \Rightarrow \boxed{v_c = 3b}$$

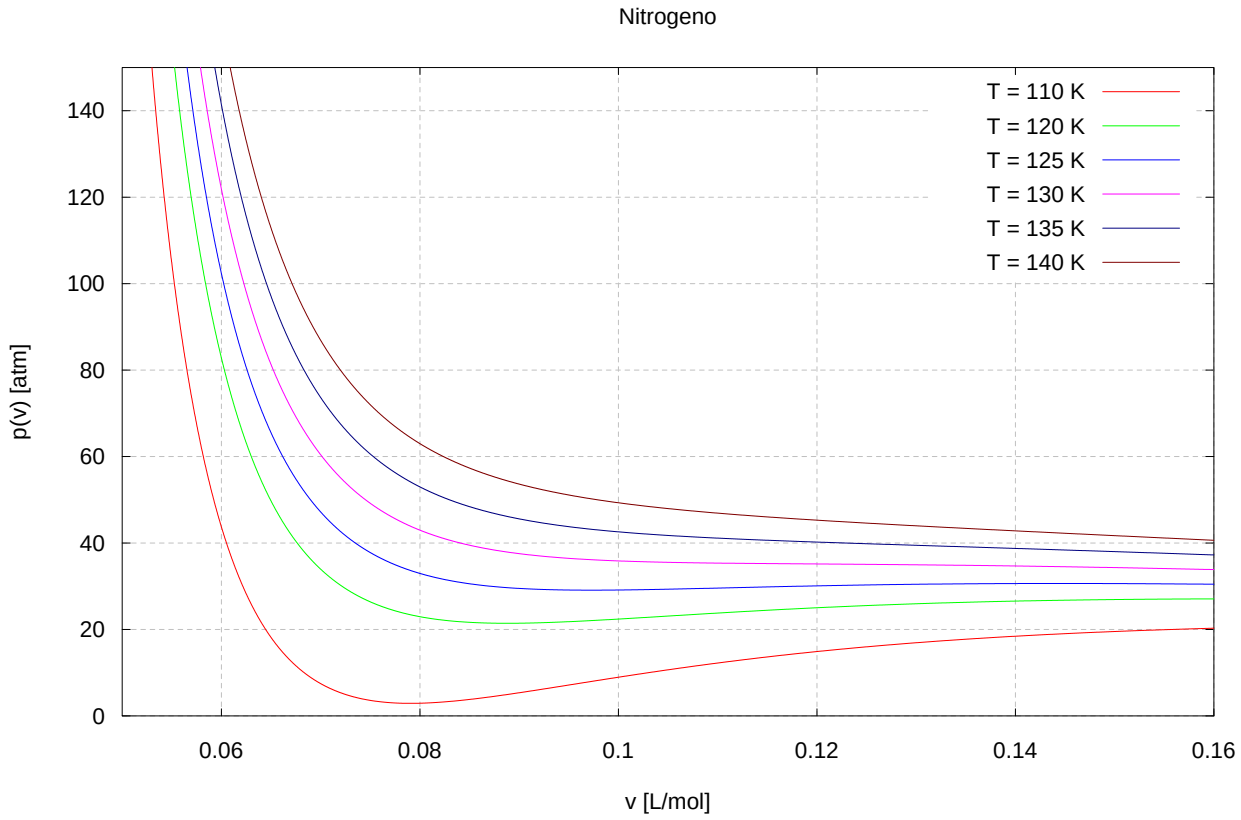

Figura 1.1: Isotermas de van der Waals para el N_2

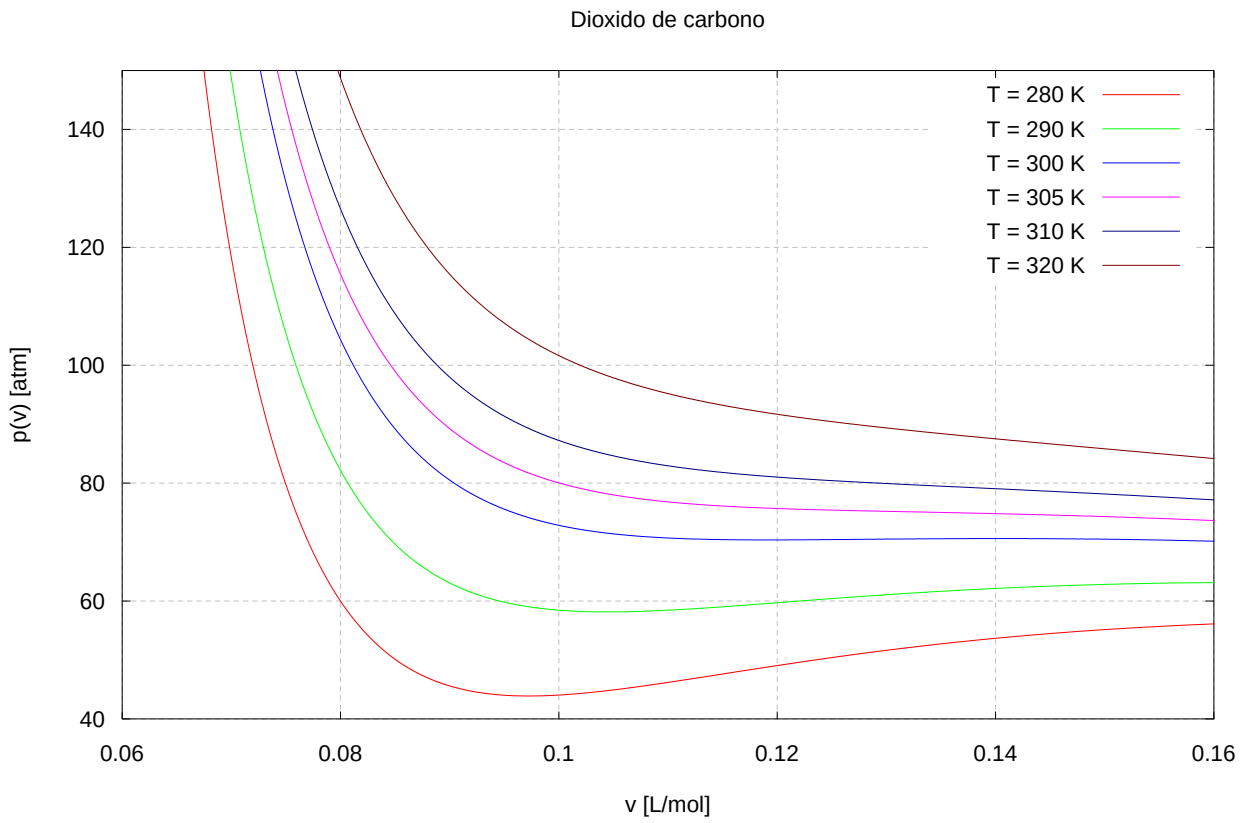

Figura 1.2: Isotermas de van der Waals para el CO_2

A partir de aquí, es fácil deducir los valores de las otras coordenadas críticas, que resumimos a continuación:

$$v_c = 3b \tag{1.3a}$$
$$T_c = \frac{8a}{27Rb} \tag{1.3b}$$
$$p_c = \frac{a}{27b^2} \tag{1.3c}$$

Es fácil comprobar que las coordenadas críticas cumplen la relación:

$$v_c = \frac{3}{8}\frac{RT_c}{p_c} \tag{1.4}$$

Esta relación podría servir para calcular v_c una vez conocidos p_c y T_c, sin necesidad de saber los parámetros, por otro lado de determinación experimental, de la ecuación de van der Waals, a y b. Pero la última parte del problema consiste en desarrollar algún método de cálculo numérico que permita obtener v_c a partir de p_c y T_c con lo que la relación 1.4 nos servirá de comparación.

La primera aproximación al problema que hemos hecho ha sido un pequeño programa que calcule, conocidos los parámetros a y b de algunas sustancias, los valores de las coordenadas críticas, para tener ciertas aproximaciones y comparar después. Este sencillo programa, que no merece mayor comentario, se hizo con las siguientes líneas de código[1]:

```
1  C
2  C NOMBRE: 1-5-masii.f
3  C AUTOR: Miguel Albaladejo Serrano mikiman@fisimur.org
4  C DESCRIPCION: Programita para calcular coordenadas crąticas de vdWaals
5  C Voy a hacer una versićn que permita obtener tablas para LaTeX
6  C 02/03/04 23:17
7  C
8        parameter (n=14)
9        dimension  a(n),b(n),p(n),t(n),v(n)
10
11       data a/1.33,4.39,4.17,3.59,6.71,1.35,6.50,0.034,0.24,2.25,1.49,
12      &1.39,1.36,5.46/b/0.036,0.051,0.037,0.043,0.056,0.032,0.056,0.024,
13      &0.027,0.043,0.040,0.039,0.032,0.030/
14       open(1,file='1-5-mas-LaTeX.dat',status='unknown')
15       write(1,50)
16       do i=1,n
17         p(i) = a(i)/(27.0*b(i)**2)
18         v(i) = 3.0*b(i)
19         t(i) = 8.0*a(i)/(27.0*0.08205*b(i))
```

[1]Las salidas que el programa da al archivo están preparadas, con un formato especial, para ser incluidas directamente en un archivo para LaTeX, con lo cual no es de extrañar el formato dado por las líneas (50) y (100)

Gas	$a\left(\frac{\text{atmL}^2}{\text{mol}^2}\right)$	$b\left(\frac{\text{L}}{\text{mol}}\right)$	p_c(atm)	v_c(L)	T_c(K)
Aire	1.330	0.036	38.009	0.108	133.413
C_2H_2	4.390	0.051	62.512	0.153	310.844
NH_3	4.170	0.037	112.816	0.111	406.988
CO_2	3.590	0.043	71.911	0.129	301.490
SO_2	6.710	0.056	79.247	0.168	432.695
Ar	1.350	0.032	48.828	0.096	152.346
Cl	6.500	0.056	76.767	0.168	419.153
He	0.034	0.024	2.186	0.072	5.116
H_2	0.240	0.027	12.193	0.081	32.099
CH_4	2.250	0.043	45.069	0.129	188.956
CO	1.490	0.040	34.491	0.120	134.516
N_2	1.390	0.03913	33.847	0.117	128.706
O_2	1.360	0.032	49.190	0.096	153.475
Vapor de agua	5.460	0.030	224.691	0.090	657.232

Cuadro 1.1: Parámetros a y b de algunos gases y sus correspondientes coordenadas críticas calculadas según (1.3). Valores de a y b tomados del libro *Lecciones de Física, Termología 1*, del profesor J.A. Ibáñez.

```
20          write(1,100)a(i),b(i),p(i),v(i),t(i)
21        end do
22        close(1)
23    50  format('$a$ &',8x,'$b$ &',8x,'$p_c$ &',8x,'$v_c$ &',8x,'$T_c$ &')
24   100  format (5('$',f9.3,'$ &',2x))
25        end
26
```

Para desarrollar un método de cálculo necesitaremos conseguir una sucesión que involucre al volumen crítico v_c. Para ello partimos de la ecuación de van der Waals para un mol, (1.1), para escribir:

$$v_c = b + \frac{RT_c}{P_c + \frac{a}{v_c^2}}$$

Ahora bien, no queremos que aparezcan ni a ni b, luego hacemos las sustituciones $b = \frac{v_c}{3}$ y $a = \frac{9}{8}Rv_cT_c$, y obtenemos:

$$\frac{2}{3}v_c = \frac{RT}{P_c + \frac{9Rv_cT_c}{8v_c^2}}$$

Todo esto lo podemos reescribir fácilmente como:

$$v_c = \frac{12RT_c}{8P_c + \frac{9RT_c}{v_c}} \tag{1.5}$$

Si consideramos que la expresión que está despejada es el término v_i de una iteración que calculamos, podemos tomar entonces el término v_c que aparece abajo del todo como

el término anterior, v_{i-1} y definir mediante esta serie una recurrencia que nos lleve al valor de v_c deseado, mediante el método de sustitución repetida.

1.6.2. Uso de la funciones `zbrent`, `bisecc` y `sustit` para el cálculo de v_c

El programa comienza con las habituales declaraciones y con algunos cálculos ((1-22)) que, salvo modificaciones posteriores, no se cambiarán. En éstas líneas también se pide al usuario que declare el valor de p_c y T_c. Los cálculos realizados son dos valores para acotar los ceros de la función de van der Waals, y el valor teórico del volumen crítico v_c a partir de la ecuación 1.4.

Después se pide al usuario que especifique si desea introducir el mismo los valores de a y b o si prefiere que los calcule el ordenador a partir de las ecuaciones 1.3, despejando a y b. Esto se realiza con un bloque `if...end if` en las líneas (24-37). Debe tenerse en cuenta que la **precisión de los cálculos** no solo se verá limitada por la tolerancia (`tol` que introduzcamos al principio, sino también por la precisión de los valores de p_c, T_c, a y b que introduzcamos. Posteriormente se pide al usuario que decida si acotar el mismo el valor de v_c por dos valores para pasarle a las subrutinas (funciones, en realidad) `zbrent` o `bisecc`. Esta opción no es la recomendada, ya que es mejor que el ordenador use los que él mismo calcula, ya que está comprobado analíticamente que la función en esos dos puntos tiene distinto signo.

Tras esto, el programa ya tiene todos los datos que necesita para calcular el valor de v_c por varios métodos. El más sencillo es el de la sustitución, `sustit`, definido en las líneas (110-124), en los que se aplica la ecuación 1.5. Lo más interesante de este método es que el valor que se utiliza como valor inicial es el definido como `v2`, que no es otro que el valor del volumen que tendría un mol de gas ideal en las condiciones (p_c, T_c). Es decir, partimos del valor de la ecuación de estado de gases ideales y llegamos al valor crítico de van der Waals, lo que resulta interesante. Es fácil comprobar analíticamente que este método converge siempre a partir de este valor.

La función `zbrent` no la explicamos, lógicamente, y se encuentra en las líneas (126-199). La otra función que se utiliza es `bisecc`, líneas (81-106), hecha por nosotros mismos, cuyo esque de petición es el mismo que el de `zbrent`, y sólo consiste en aplicar el método de bisección visto en clase.

El núcleo del programa son las líneas (52-54), en las que se llama a las funciones anteriormente descritas para hallar ceros de distintas funciones. Las líneas (56-60) sólo se encargan de escribir los valores obtenidos mediante estos métodos, y de escribir el valor teórico para poder comparar.

Como comentario final, notar que el programa parece dar un valor más *fino* cuando no se introducen los parámetros a y b que cuando se introducen. Esto se debe a que los parámetros reales, experimentales, a y b pueden no coincidir con los que se calculan

a partir de la ecuación de van der Waals. Esto es lógico porque la adecuación de la ecuación a la realidad es buena cualitativamente aunque a veces falla en sus predicciones cuantitativas. Por otro lado, es lógico que el resultado concuerde más con el valor teórico cuando no se introducen a y b, ya que entonces el ordenador los calcula por sí mismo mediante ecuaciones deducidas mediante la propia ecuación de van der Waals, es decir, siguen la propia coherencia matemática de la ecuación.

A continuación se detalla todo el código FORTRAN usado.

```
C
C NOMBRE: 1-5-mas.f
C AUTOR: Miguel Albaladejo
C DESCRIPCION: Estudio de los puntos criticos de la ec vdW
C FECHA: 5 de marzo de 2004
C
      external vdw,g
      common /eq/a,b,pc,tc,R
      dimension vc(3)
      write(*,100)
      write(*,150)
      write(*,100)

      R = 0.08205
      write(*,200)
      read(*,*)pc,tc
      write(*,250)
      read(*,*)tol

      v1 = (1.0/8.0)*(tc*R)/(pc)
      v2 = (R*tc/pc)
      vt = v2*(3.0/8.0)

  10  write(*,*)'Quieres introducir los valores de a y b?(1==si, 2==no)'
      read(*,*)elec
      if (int(elec).eq.1) then
         write(*,*)'a = '
         read(*,*),a
         write(*,*)'b = '
         read(*,*),b
       else if (int(elec).eq.2) then
         a = (27.0/64.0)*(tc*R)**2/(pc)
         b = (1.0/8.0)*(tc*R)/(pc)
         continue
       else
         goto 10
       end if
```

```
38    20  write(*,*)'Quieres escribir tu valores para acotar o los elijo yo'
39       &,' mismo? (1==si [no recomendado], 2==no)'
40        read(*,*)elec
41        if (int(elec).eq.1) then
42           write(*,*)'v1 (L.) = '
43           read(*,*),v1
44           write(*,*)'v2 (L.) = '
45           read(*,*),v2
46         else if (int(elec).eq.2) then
47           continue
48         else
49           goto 20
50         end if
51
52        vc(1) = zbrent(vdw,v1,v2,tol)
53        vc(2) = bisecc(vdw,v1,v2,tol)
54        vc(3) = sustit(g,v2,tol)
55
56        write(*,*)'ZBRENT -> vc (L) = ',vc(1)
57        write(*,*)'BISECC -> vc (L) = ',vc(2)
58        write(*,*)'SUSTIT -> vc (L) = ',vc(3)
59        write(*,*) 'Su valor teorico es vc_teorico = (3/8)* (R*tc/pc) = ',
60       &vt,' L.'
61    100 format (20x,25('*'))
62    150 format (20x,'ECUACION DE VAN DER WAALS')
63    200 format ('Introduce los valores de p_c (atm) y T_c (K)')
64    250 format ('Escribe el măximo error en vc que pretendes')
65        end
66  ****** funcion para la ecuacion de van der waals
67
68        function vdw(vc)
69        common /eq/a,b,pc,tc,R
70         vdw = (pc + a/(vc**2) )*(vc - b) - R*tc
71        return
72        end
73  ****** funcion para el metodo de sustitucion
74
75        function g(vc)
76        common /eq/a,b,pc,tc,R
77         g = (12.0*R*tc)/(8.0*pc + (9.0*R*tc)/(vc) )
78        return
79        end
80  ****** funcion para la biseccion
81        function bisecc(f,xlo,xro,tol)
82        xl = xlo
83        xr = xro
```

```
84        do i=1,1000
85        if (f(xl)*f(xr).gt.0) then
86          bisecc = -1.0
87          write(*,*)'Has elegido malos valores. Te lo adverti'
88          return
89         else
90          continue
91        end if
92        xm = (xl + xr)/2.0
93        error = abs( xl - xr)
94
95        if (error.gt.(2.0*tol)) then
96              if ((f(xm)*f(xl)).gt.0.0) then
97                xl = xm
98               else
99                xr = xm
100             end if
101        else
102            bisecc = xm
103       end if
104       end do
105       return
106       end
107
108 ****** funcion para el mĆtodo de sustitucićn
109
110       function sustit(f,x1,tol)
111       x = x1
112       do i=1,10000
113        xold = x
114        x = f(x)
115        if ((abs(xold-x)).gt.tol) then
116         continue
117        else
118         sustit = x
119         return
120        end if
121       end do
122
123       return
124       end
125 ****** funcion/subrutina zbrent
126       FUNCTION zbrent(func,x1,x2,tol)
127
128       INTEGER ITMAX
129       REAL zbrent,tol,x1,x2,func,EPS
```

```
130       EXTERNAL func
131       PARAMETER (ITMAX=100,EPS=3.e-8)
132       INTEGER iter
133       REAL a,b,c,d,e,fa,fb,fc,p,q,r,s,tol1,xm
134       a=x1
135       b=x2
136       fa=func(a)
137       fb=func(b)
138       if((fa.gt.0..and.fb.gt.0.).or.(fa.lt.0..and.fb.lt.0.))pause
139      *'root must be bracketed for zbrent'
140       c=b
141       fc=fb
142       do 11 iter=1,ITMAX
143         if((fb.gt.0..and.fc.gt.0.).or.(fb.lt.0..and.fc.lt.0.))then
144           c=a
145           fc=fa
146           d=b-a
147           e=d
148         endif
149         if(abs(fc).lt.abs(fb)) then
150           a=b
151           b=c
152           c=a
153           fa=fb
154           fb=fc
155           fc=fa
156         endif
157         tol1=2.*EPS*abs(b)+0.5*tol
158         xm=.5*(c-b)
159         if(abs(xm).le.tol1 .or. fb.eq.0.)then
160           zbrent=b
161           return
162         endif
163         if(abs(e).ge.tol1 .and. abs(fa).gt.abs(fb)) then
164           s=fb/fa
165           if(a.eq.c) then
166             p=2.*xm*s
167             q=1.-s
168           else
169             q=fa/fc
170             r=fb/fc
171             p=s*(2.*xm*q*(q-r)-(b-a)*(r-1.))
172             q=(q-1.)*(r-1.)*(s-1.)
173           endif
174           if(p.gt.0.) q=-q
175           p=abs(p)
```

```
176             if(2.*p .lt. min(3.*xm*q-abs(tol1*q),abs(e*q))) then
177               e=d
178               d=p/q
179             else
180               d=xm
181               e=d
182             endif
183           else
184             d=xm
185             e=d
186           endif
187           a=b
188           fa=fb
189           if(abs(d) .gt. tol1) then
190             b=b+d
191           else
192             b=b+sign(tol1,xm)
193           endif
194           fb=func(b)
195   11    continue
196         pause 'zbrent exceeding maximum iterations'
197         zbrent=b
198         return
199         END
```

Tema 2

Ecuaciones lineales

2.3. Leyes de Kirchhoff I

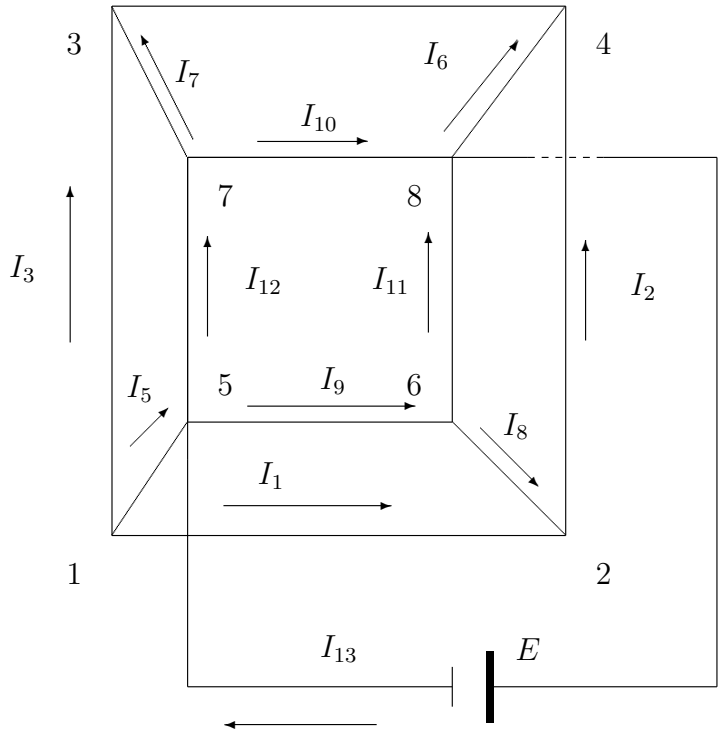

Figura 2.1: Circuito del cubo de resistencias

En este programa hacemos una pequeña variante de otro realizado en clase. Tenemos el circuito de la figura 2.1, que es la proyección bidimensional de un cubo tridimensional con resistencias en sus aristas. Queremos conocer la resistencia equivalente del cubo haciendo variar una de ellas, que llamaremos r, concretamente, la que está entre los puntos 1 y 3. Todas las demás valen $R = 1\ \Omega$. Para ello necesitamos conocer la intensidad que pasa por la rama que está antes y después del cubo, sin incluir a este, pero incluyendo

al generador E, además de conocer el valor de E. Definimos las intensidades con las direcciones que marcan las flechas en la figura, y aplicamos las leyes de Kirchhoff al sistema, para obtener las ecuaciones:

nudo 1	$0 = I_1 + I_3 + I_5 - I_{13}$
nudo 2	$0 = -I_1 + I_2 - I_8$
nudo 4	$0 = -I_2 + I_4 - I_6$
nudo 3	$0 = -I_3 - I_4 - I_7$
nudo 5	$0 = -I_5 + I_9 + I_{12}$
nudo 6	$0 = I_8 - I_9 + I_{11}$
nudo 8	$0 = I_6 - I_{10} - I_{11} + I_{13}$
1562	$0 = -I_1R + I_5R + I_8R + I_9R$
6842	$0 = I_2R - I_6R + I_8R - I_{11}R$
8734	$0 = -I_4R - I_6R + I_7R - I_10R$
1375	$0 = I_3r - I_5R - I_7R - I_{12}R$
1342	$0 = I_1R - I_2R + I_3r - I_4R$

Cuadro 2.1: Ecuaciones para el circuito del problema 2.3.

La resistencia equivalente del sistema vendrá dado por:

$$R_{eq} = \frac{E}{I_{13}} \tag{2.1}$$

En general, cuando todas las resistencias son iguales, de valor R, se obtiene por simetría que:

$$R_{eq} = \frac{5}{6}R \tag{2.2}$$

Para resolver este sistema lineal usamos las subrutinas `ludcmp` y `lubksb`. Para ello tenemos que pasarle al sistema la matriz de coeficientes a y la matriz de términos independientes b. Para ello las escribiremos directamente al sistema, valiéndonos cuando sea necesario de un programa sencillo creado llamado `petmat.f`. El programa `2-3-mas.f` se explica a continuación.

Lo primero es declarar todas las matrices necesarias, así como valores de datos que el problema nos ofrece (como, por ejemplo, el voltaje E, aunque el resultado que a nosotros nos interesa es independiente de este valor), así como abrir el archivo (*2-3-mas-Req.dat*)en el que guardaremos todos los datos que vayamos calculando para representar el variación de la resistencia equivalente (líneas (1-8)). A continuación viene un ciclo muy grande. Lo primero que se hace en este ciclo es poner a cero todos los términos $a(i, j)$ de la matriz de coefcientes, así como los $b(i)$. Después se definen los coeficientes, que, a partir de las ecuaciones del sistema, sabemos que son distintos de cero ((9-69)). También se define la variable r, que va tomando sucesivos valores, es decir, la resistencia variable. Entonces se definen los coeficientes que varían con r ((9-69)). Se resuelve el sistema entonces mediante las subrutinas `ludcmp` y `lubksb`. Como la intesidad que queremos es la I_{13}, y

estas subrutinas devuelven los valores de las incógnitas mediante la propia matriz b, la línea (74) nos permite calcular la resistencia equivalente para cada valor de r. Después se escribe dicha resistencia en pantalla y en archivo, y se cierra el ciclo y el archivo (líneas (76-79)).

Los valores de las resistencias equivalentes obtenidas para cada valor de la resistencia entre los nodos 1 y 3 se representan en la tabla 2.2. Como era de esperar, para $r = 1\ \Omega$, se tiene $R_{eq} = 5/6 = 0.8333\ldots\ \Omega$.

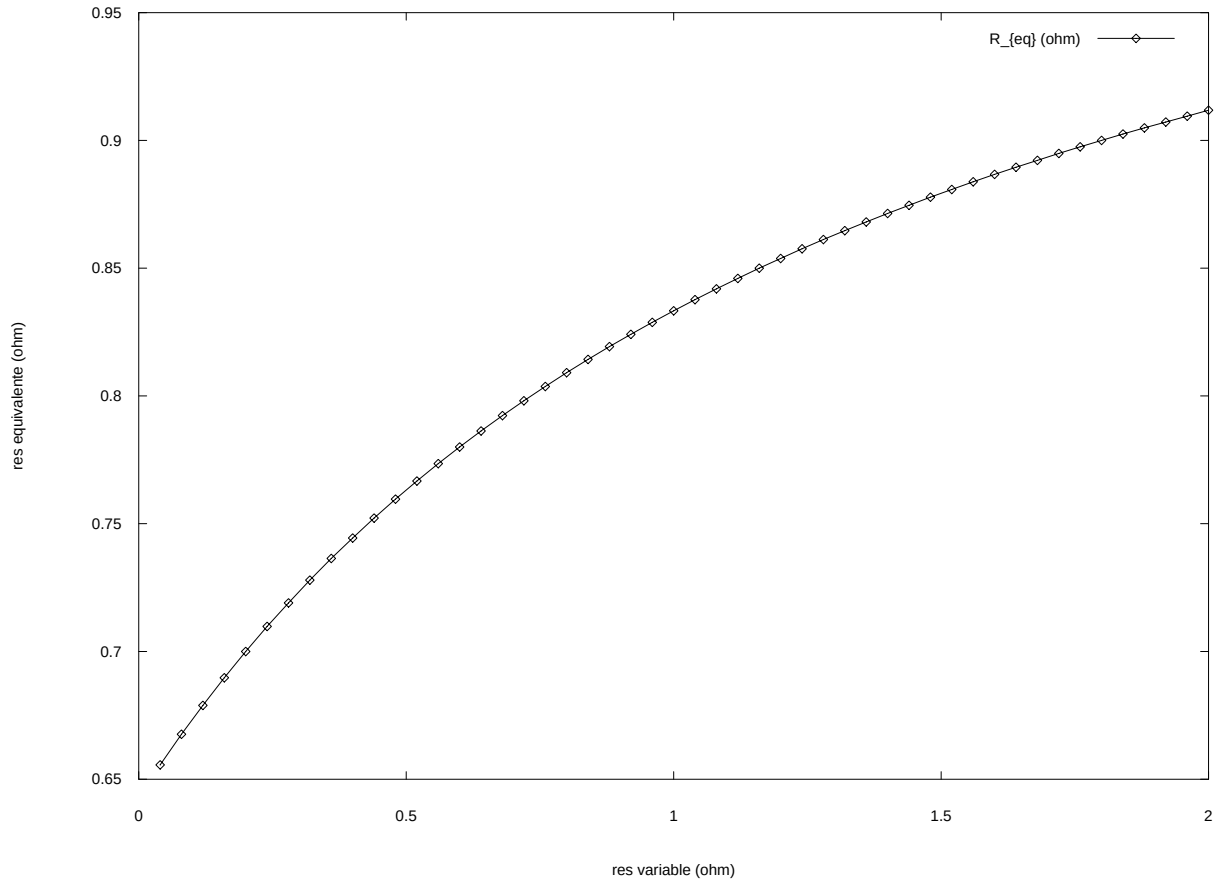

Figura 2.2: Representación de la variación de R_{eq} frente a r

```
1         parameter (n=13)
2         dimension a(1:n,1:n), b(1:n), indx(1:n)
3         E = 1.0
4         m=50
5         rmax = 2.0
6         step = rmax/real(m)
7         open (1,file='2-3-mas-Req.dat',status='unknown')
8         r = 0.0
9         do i=1,m+1
```

```
10
11          do l=1,n
12          do j=1,n
13           b(l) = 0.0
14           a(l,j) = 0.0
15          end do
16          end do
17
18         a( 1, 1) =   1.
19         a( 1, 3) =   1.
20         a( 1, 5) =   1.
21         a( 1, 13) =  -1.
22         a( 2, 1) =  -1.
23         a( 2, 2) =   1.
24         a( 2, 8) =  -1.
25         a( 3, 2) =  -1.
26         a( 3, 4) =   1.
27         a( 3, 6) =  -1.
28         a( 4, 3) =  -1.
29         a( 4, 4) =  -1.
30         a( 4, 7) =  -1.
31         a( 5, 5) =  -1.
32         a( 5, 9) =   1.
33         a( 5, 12) =   1.
34         a( 6, 8) =   1.
35         a( 6, 9) =  -1.
36         a( 6, 11) =   1.
37         a( 7, 6) =   1.
38         a( 7, 10) =  -1.
39         a( 7, 11) =  -1.
40         a( 7, 13) =   1.
41         a( 8, 1) =  -1.
42         a( 8, 5) =   1.
43         a( 8, 8) =   1.
44         a( 8, 9) =   1.
45         a( 9, 2) =   1.
46         a( 9, 6) =  -1.
47         a( 9, 8) =   1.
48         a( 9, 11) =  -1.
49         a( 10, 4) =  -1.
50         a( 10, 6) =  -1.
51         a( 10, 7) =   1.
52         a( 10, 10) =  -1.
53
54         a( 11, 5) =  -1.
55         a( 11, 7) =  -1.
```

```
56          a( 11, 12) = -1.
57
58          a( 12, 1) =  -1.
59          a( 12, 2) =  -1.
60          a( 12, 4) =  -1.
61
62          a( 13, 5) =   1.
63          a( 13, 9) =   1.
64          a( 13, 11) =   1.
65
66          r = r + step
67          a( 11, 3) =  r
68          a( 12, 3) =  r
69          b(13) = E
70
71          call ludcmp(a,n,n,indx,d)
72          call lubksb(a,n,n,indx,b)
73
74          Req = E/b(13)
75
76          write(*,*)'Req = E/I(13) = ',Req
77          write(1,100)r,Req
78          end do
79          close(1)
80
81    100   format(2x,f9.4,2x,f9.4)
82          end
83
```

2.5. Sistema de muelles sometidos a campo gravitatorio

Nuestro sistema está formado por una masa sujetada por dos muelles, sometida a la acción de la gravedad. La energía potencial de este sistema viene dada por:

$$V(x,y) = \frac{1}{2}k_1\left(\sqrt{x^2+y^2} - L_1\right)^2 + \frac{1}{2}k_2\left(\sqrt{(x-D)^2+y^2} - L_2\right)^2 - mgy \qquad (2.3)$$

Una representación analítica de este potencial está representada en la figura 2.3.

Lo que queremos es conocer la posición de equilibrio estático (estable), que se corresponde con un mínimo de potencial. Para ello habría que resolver las ecuaciones:

$$\frac{\partial V}{\partial x}|_{eq} = \frac{\partial V}{\partial y}|_{eq} = 0$$

con la condición adicional de que se anulase la derivada segunda. Las derivadas parciales vienen dadas por:

$$\frac{\partial V}{\partial x} = k_1 \left(\sqrt{x^2+y^2} - L_1\right) \frac{x}{\sqrt{x^2+y^2}} + k_2 \left(\sqrt{(x-D)^2+y^2} - L_2\right) \frac{x-D}{\sqrt{(x-D)^2+y^2}} \tag{2.4a}$$

$$\frac{\partial V}{\partial y} = k_1 \left(\sqrt{x^2+y^2} - L_1\right) \frac{y}{\sqrt{x^2+y^2}} + k_2 \left(\sqrt{(x-D)^2+y^2} - L_2\right) \frac{y}{\sqrt{(x-D)^2+y^2}} - mg \tag{2.4b}$$

Los valores que nos conciernen (DNI par) vienen dados en la tabla 2.2:

Dato	Valor
k_1	10 N/m
k_2	20 N/m
L_1	0.1 m
L_2	0.1 m
D	0.1 m
g	9.81 m/s^2

Cuadro 2.2: Datos para el sistema

Estas ecuaciones parecen altamente no lineales, y no parece que pueda ser linealizadas mediante un cambio de variables. La mejor manera de resolver el problema de hallar un mínimo del potencial sería hallar "ceros" de las ecuaciones dadas por 2.4. Pero esto también puede llegar a ser problemático, ya que requiriría, entre otras cosas, disponer de un método que no sea el de bisección o similares, como los usados por `zbrent` para minimizar, ya que la forma parabólica del potencial hace difícil usar estos métodos, o, cuando menos, calcular también las segundas derivadas, que no sería especialmente complicado, pero tampoco es nuestra intención, ya que queremos resolver el problema usando métodos numéricos. Por todo esto, la solución que planteamos es calcular el mínimo de potencial mediante un método gráfico–numérico. Gráfico no porque nos basemos en dibujos para calcular la posición del mínimo, sino porque nos basaremos en el valor que el potencial toma en cada punto de los accesibles al sistema. Calcularemos el valor del potencial en cada punto de una malla cuadrada, e iremos comparando los valores que vayamos obteniendo sucesivamente, quedándonos, evidentemente, con el valor más pequeño. Para ello realizamos el programa *2-5-mas.f*.

En las líneas (7-19) se preparan los datos y variables del sistema. En las líneas (20-26) se declaran valores necesarios para un ciclo doble `do ... end do`, en el que se recorren las variables x e y, desde su valor mínimo, 0, hasta unos valores máximos, definidos mediante valores físicos lógicos. La x puede llegar, como máximo, hasta el valor dado por D. Por otro lado, el valor mínimo de y, la altura mínima a la que puede estar el sistema, es la altura a la que estaría la masa si pendiese sólo del muelle más elástico. Por consideraciones de fuerzas en equilibrio, dcho valor máximo (como altura, es mínimo)

viene dado por $y = L + mg/k$, referidas L y k al muelle de más elasticidad. El doble ciclo actúa de la siguiente manera. Para cada iteración, se calcula el valor del potencial en el punto correspondiente (determinado por las variables de los ciclos `do`), y si resulta menor que un valor obtenido anteriormente, llamado `vmin` (al que inicialmente se le debe dar un valor cualquiera, suficientemente grande), este nuevo valor más pequeño sustituye al antiguo, de modo que dicha variable `vmin` va tomando el valor más pequeño de todos los que están contenidos en el recuadro en el que se puede encontrar la masa. Las posiciones en la que se encuentra un valor del potencial que se almacena como nuevo `vmin` son guaradas en dos variables `xmin` e `ymin`. Cuando termina el ciclo, sabemos cual es el valor mínimo del potencial y en que posición está dicho valor. Esta posición corresponderá a la de equilibrio. Pero como también queremos poder ver dicho potencial, podemos ir escribiendo las distintas parejas de valores (x, y), así como el valor del potencial en dicho punto, $V(x, y)$, en un archivo *2-5-mas-potencial.dat*, lo que nos servirá para representar el potencial. Dicha representación se encuentra en la figura 2.4. Los valores x e y en los que encontramos el mínimo de potencial, esto es, la posición de equilibrio, que calculamos nosotros, así como el valor del potencial en dicho punto, se recogen en la tabla 2.3. Como se puede observar en las figuras, esta posición se corresponde, efectivamente, a un mínimo de potencial.

En la ejecución del programa es importante destacar el papel del parámetro n. Este señala el número máximo de iteraciones que se deben realizar (se harán n pasos del bucle interior por cada uno de los n pasos del bucle exterior, en total, n^2 pasos). Hay que ser "cuidadosos" a la hora de determinarlo, pues puede ralentizar la ejecución. Por otra parte, el valor de las coordenadas será más preciso también dependiendo de que este parámetro n sea mayor o menor.

Dato	Valor
x	0.0624 m
y	0.1264 m
$V_{min}(x, y)$	-0.1054 J

Cuadro 2.3: Valores del mínimo de potencial

```
1  C
2  C NOMBRE: 2-5-mas.f
3  C AUTOR: Miguel Albaladejo Serrano
4  C DESCRIPCION:
5  C FECHA:
6
7        real k1,k2,l1,l2,m
8        common /poten/k1,k2,l1,l2,d,m,gacc
9        external v
10       write(*,200)
11       write(*,300)
```

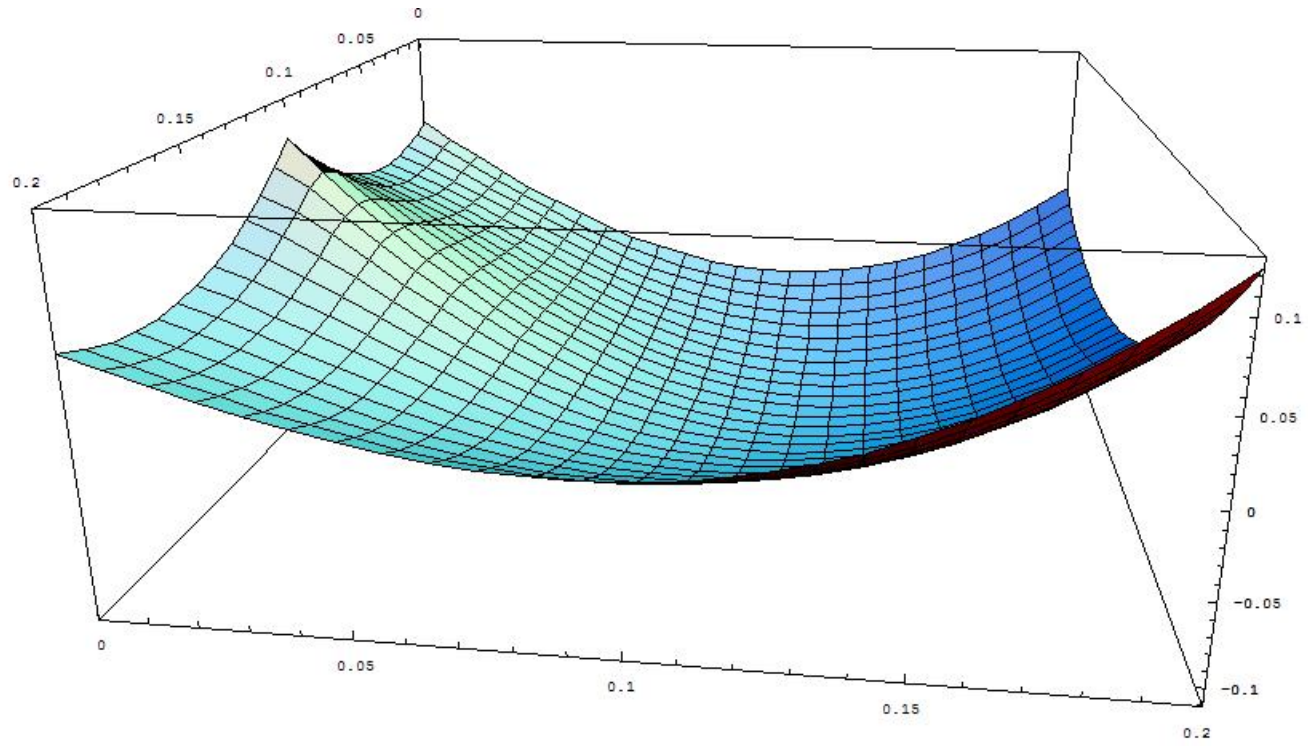

Figura 2.3: Representación analítica del potencial problema

```
12        write(*,200)
13        gacc = 9.81
14        k1 = 10.0
15        k2 = 20.0
16        l1 = 0.1
17        l2 = 0.1
18        d = 0.1
19        m = 0.1
20        open (1,file='potencial.dat',status='unknown')
21        n = 300
22        xmin = 0.05
23        ymin = 0.05
24        vmin = v(xmin,ymin)
25        hx = 0.2/real(n)
26        hy = 0.2/real(n)
27        do i=1,n-1
28        x = x + hx
29        y = 0.0
30        do j=1,n-1
31        y = y + hy
32        compar = v(x,y)
33        write (1,100)x,y,v(x,y)
```

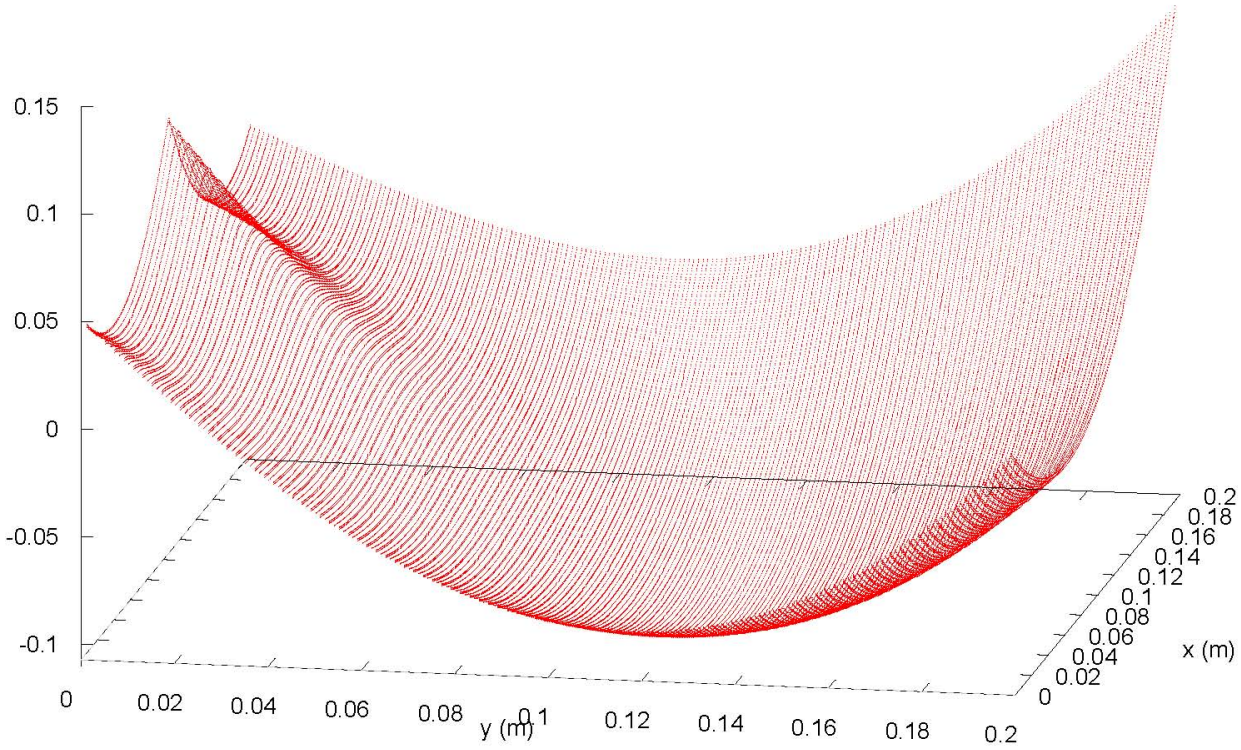

Figura 2.4: Representación mediante puntos (con GNUPLOT) del potencial problema

```
34          if (compar.lt.vmin) then
35             vmin = v(x,y)
36             xmin = x
37             ymin = y
38             continue
39           else
40             continue
41          end if
42          end do
43          end do
44          close (1)
45
46          write(*,*)'La posición de equilibrio parece ser:'
47          write(*,*)'xmin = ',xmin
48          write(*,*)'ymin = ',ymin
49          write(*,*)'Vmin = ',vmin
50    100   format (4x,f9.3,4x,f9.3,4x,f13.5)
51    200   format (20x,47('*'))
52    300   format (20x,'Posicion de equilibrio de un sistema de muelles')
53          end
54
55          function v(x,y)
```

```
56          real k1,k2,l1,l2,m
57          common /poten/k1,k2,l1,l2,d,m,gacc
58          v = 0.5*k1*(sqrt(x**2 + y**2) - l1)**2 + 0.5*k2*
59         &(sqrt((x-d)**2 + y**2) - l2)**2 - m*gacc*y
60          return
61          end
62
63
64
```

2.6. Leyes de Kirchhoff II

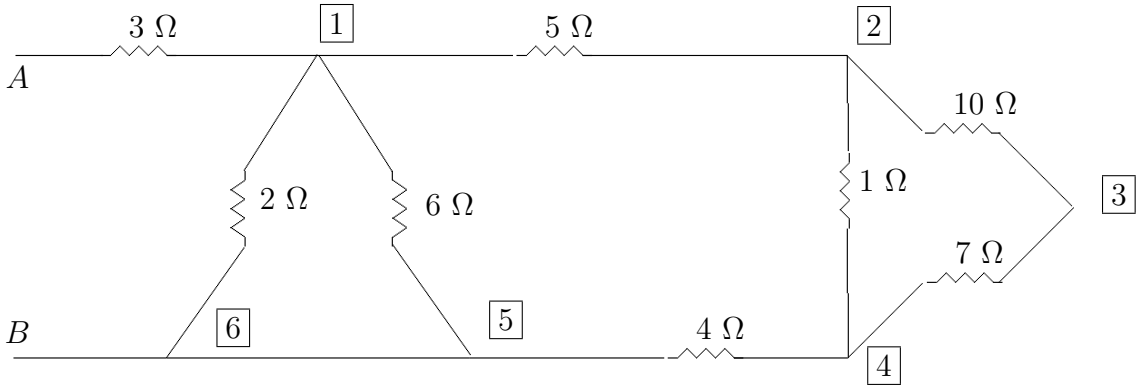

Figura 2.5: Circuito problema

Tenemos el circuito de la figura 2.5, y queremos conocer la intensidad que pasa por cada resistencia y el voltaje en cada uno de los puntos 1–6. Para ello aplicaremos sistemáticamente las leyes de Kirchhoff a los nodos 1,2,4 y a las mallas BA16B, 12451, 123451 y 1561, obteniendo siete ecuaciones, las que necesitamos para conocer las siete intensidades I_i $i = 1, 2, \ldots, 7$. Esto resulta en las ecuaciones:

nudo 1	$0 = I_1 - I_2 - I_6 - I_7$
nudo 2	$0 = I_2 - I_3 - I_5$
nudo 4	$0 = I_3 + I_5 - I_4$
1561	$0 = I_6 R_7 - I_7 R_8$
12451	$0 = I_2 R_2 + I_5 R_6 + I_4 R_5 - I_6 R_7$
123451	$0 = I_2 R_2 + I_3 (R_3 + R_4) + I_4 R_5 - I_6 R_7$
BA16B	$-1 = I_1 R_1 I_7 R_8$

Cuadro 2.4: Ecuaciones para el circuito del problema 2.6.

Los valores de las resistencias vienen recogidos en la tabla 2.5.

R_1	3 Ω
R_2	5 Ω
R_3	10 Ω
R_4	7 Ω
R_5	4 Ω
R_6	1 Ω
R_7	6 Ω
R_8	2 Ω

Cuadro 2.5: Valores de las resistencias para el problema 2.6

Para resolver este sistema, como ya hicimos, hacemos un programa en el que poder aplicar las subrutinas vistas en clase. Con ello resolveremos el problema de las intensidades. Para obtener los potenciales en cada punto los hallaremos teniendo en cuenta que entre cada dos puntos i y j la diferencia de potencial (con el signo correcto) es $V_j - V_i = \sum_n I_n R_n$, donde el sumatorio se extiende sobre las resistencias que están entre i y j.

El núcleo del programa son las subrutinas `ludcmp` y `lubksb`. El programa comienza con las declaraciones y las definiciones de las matrices necesarias, con los elementos, usando para ello en ocasiones el programa *petmat.f* ((5-73)). En las líneas (74-78) se llama a las subrutinas, y en las líneas () se asigna a la variable dimensional `curr` el valor correspondiente de las corrientes incógnitas dadas por la matriz `b`. En las líneas () se calculan los potenciales en los puntos mediante la aplicación de lo dicho anteriormente. Después, estos datos vienen escritos en la pantalla.

Los resultados que obtenemos para las intensidades y los potenciales vienen recogidos en las tablas 2.6 y 2.7 respectivamente.

Intensidad	Valor (A)
I_1	-0.232374504
I_1	-0.0304568596
I_1	-0.00169204467
I_1	-0.0304568596
I_1	-0.0287648141
I_1	-0.0504794121
I_1	-0.151438236

Cuadro 2.6: Valores resultado de las intensidades

```
1  C
2  C NOMBRE: 2-6-mas.f
3  C AUTOR: Miguel Albaladejo Serrano
4  C DESCRIPCION: Aplicacion leyes Kirchhoff
5        parameter (n=7)
```

Potencial	Valor (V)
I_1	1.69712353
I_1	1.84940779
I_1	1.86632824
I_1	1.87817252
I_1	2.0
I_1	2.0
I_1	2.0

Cuadro 2.7: Valores resultado de los potenciales

```
      dimension a(1:n,1:n), b(1:n), r(8), indx(1:n),V(0:7), curr(7)
****** RESISTENCIAS ****************************************************
      r(1) = 3.0
      r(2) = 5.0
      r(3) = 10.0
      r(4) = 7.0
      r(5) = 4.0
      r(6) = 1.0
      r(7) = 6.0
      r(8) = 2.0
****** POTENCIALES EN A y B ***************************************
      V(0) = 1.0
      V(7) = 2.0
****** ELEMENTOS DE LA MATRIZ DE COEFICIENTES **************************
      a( 1, 1) =   1.
      a( 1, 2) =  -1.
      a( 1, 3) =   0.
      a( 1, 4) =   0.
      a( 1, 5) =   0.
      a( 1, 6) =  -1.
      a( 1, 7) =  -1.
      a( 2, 1) =   0.
      a( 2, 2) =   1.
      a( 2, 3) =  -1.
      a( 2, 4) =   0.
      a( 2, 5) =  -1.
      a( 2, 6) =   0.
      a( 2, 7) =   0.
      a( 3, 1) =   0.
      a( 3, 2) =   0.
      a( 3, 3) =   1.
      a( 3, 4) =  -1.
      a( 3, 5) =   1.
      a( 3, 6) =   0.
```

```
40        a( 3, 7) =   0.
41        a( 4, 1) =   0.
42        a( 4, 2) =   0.
43        a( 4, 3) =   0.
44        a( 4, 4) =   0.
45        a( 4, 5) =   0.
46        a( 4, 6) =  r(7)
47        a( 4, 7) =  -r(8)
48        a( 5, 1) =   0.
49        a( 5, 2) =   r(2)
50        a( 5, 3) =   0.
51        a( 5, 4) =   r(5)
52        a( 5, 5) =   1.
53        a( 5, 6) =  -r(7)
54        a( 5, 7) =   0.
55        a( 6, 1) =   0.
56        a( 6, 2) =   r(2)
57        a( 6, 3) =   r(3)+r(4)
58        a( 6, 4) =   r(5)
59        a( 6, 5) =   0.
60        a( 6, 6) =  -r(7)
61        a( 6, 7) =   0.
62        a( 7, 1) =   r(1)
63        a( 7, 2) =   0.
64        a( 7, 3) =   0.
65        a( 7, 4) =   0.
66        a( 7, 5) =   0.
67        a( 7, 6) =   0.
68        a( 7, 7) =   r(8)
69  ****** ELEMENTOS DE LA MATRIZ B ****************************************
70        do i=1,n
71        b(i) = 0
72        end do
73        b(7) = V(0) - V(7)
74  ****** EL ORDENADOR HACE AHORA LOS CALCULOS...
75        call ludcmp(a,n,n,indx,d)
76
77        call lubksb(a,n,n,indx,b)
78  ****** ... Y NOSOTROS LOS USAMOS
79        write(*,*)
80        do i=1,n
81        curr(i) = b(i)
82        write(*,*) 'I(',i,' ) = ',curr(i)
83        end do
84  ****** CALCULO DE LOS POTENCIALES EN CADA PUNTO
85        V(1) = V(0) - curr(1)*r(1)
```

```
86        V(2) = V(1) - curr(2)*r(2)
87        V(3) = V(2) - curr(3)*r(3)
88        V(4) = V(3) - curr(3)*r(4)
89        V(5) = V(1) - curr(6)*r(7)
90        V(6) = V(1) - curr(7)*r(8)
91        do i=1,7
92        write(*,*)'V(',i,') = ',V(i)
93        end do
94        end
95
```

El programa *petmat.f*

Para la introducción cómoda de matrices, muy frecuente en este tema, hemos creado el programa `petmat`, abreviación de *petición de matriz*. Este programa pide la dimensión de la matriz que se quiere introducir, líneas (2-9), y comienza a pedir, uno a uno, los elementos de una matriz que se llama, genéricamente, `a`. Estos elementos de matriz vienen copiados a un archivo, *matrix.dat*. Se puede también incluir las líneas (22-28) para que solo escriba al archivo los elementos distintos de cero. Si no, bastará dejar la línea (24). Es un programa bastante útil para su sencillez, que permite la inclusión cómoda de matrices en un programa. Sólo hay que abrir el archivo escrito, cortar y copiar.

```
1  C     petición de matriz
2         dimension a(100,100)
3         write(*,*)'Escribe la dimensión ( < 100 ) de la matriz (nxn)'
4         write(*,*)'n = '
5         read(*,*)n
6         write(*,*)'m = '
7         read(*,*)m
8         n=int(n)
9         m=int(m)
10        do i=1,n
11        do j=1,m
12        write(*,*)'a(',i,',',j,') = '
13        read(*,*)a(i,j)
14        end do
15        end do
16
17        open (1,file='matrix.dat',status='unknown')
18        do i=1,n
19        do j=1,n
20 C Opcion de escribir solo los coefs distintos de cero. Para escribir to-
21 C dos, quitar lineas 22-27, dejando solo la 24
22        compar = a(i,j)
```

```
23          if (compar.ne.0.0) then
24            write(1,*)'a(',i,',',j,') = ',a(i,j)
25            continue
26           else
27            continue
28          end if
29          end do
30          end do
31
32          end
33
34
```

Tema 3

Derivación e integración

3.7. Determinación de las magnitudes de Planck mediante análisis dimensional

Pretendemos en este documento establecer de forma clara y formal cuales son la longitud y la masa de Planck, es decir, las escalas mínimas que hemos de encontrar al formular una teoría coherente sobre la estructura de la materia. Para ello, usaremos las ecuaciones dimensionales de las tres constantes básicas de la Física Moderna: $\hbar$ (la constante de Planck), G_N (La constante de la Gravitación Universal de Newton) y c (la velocidad de la luz en el vacío).

3.7.1. Introducción

Lo primero que debemos hacer es obtener las ecuaciones dimensionales de las tres constantes que queremos usar, es decir, su expresión en función de las magnitudes (no unidades) fundamentales del Sistema Internacional de Unidades (SI). Las tres constantes dependen (a lo sumo) de tres de éstas magnitudes fundamentales, a saber, la masa, la longitud, y el tiempo (M,L y T respectivamente). Por otro lado, necesitaremos para los cálculos posteriores los valores de dichas constantes, valores que también adoptaremos en unidades SI. Tampoco nos interesaremos por valores exactos (basados en mediciones recientes precisas, etc ...), nos bastará con valores estándar de dichas magnitudes, pues los datos que buscamos serán sólo valores de referencia. A continuación analizamos las tres constantes por separado. Las ecuaciones de dimensiones se recogen en el cuadro 3.1.

G_N Las magnitudes que intervienen en la ecuación dimensional de G_N se deducen directamente de la ley de Newton de la Gravitación Universal:

$$F_g = G\frac{m_1 m_2}{r^2}\ \hat{\mathbf{u}}_\mathbf{r}$$

Teniendo en cuenta que la fuerza tiene dimensiones de masa×aceleración, y a su vez,

las dimensiones de la aceleración, y despejando G_N, podemos deducir fácilmente su ecuación dimensional.

$\hbar$ Las dimensiones de la constante de Plank son de energía×tiempo. A su vez, las dimensiones de la energía se pueden obtener de cualquier "fórmula clásica" para la energía: cinética, potencial, trabajo mecánico...:

$$\begin{aligned} T &= \frac{1}{2}mv^2 \Rightarrow [E] = ML^2T^{-2} \\ U &= mg\triangle h \Rightarrow [E] = ML^2T^{-2} \\ W &= \oint \vec{F}\vec{dl} \Rightarrow [E] = ML^2T^{-2} \\ &\dots \end{aligned}$$

c Sin duda este es el caso más sencillo, pues las dimensiones de c son las de una velocidad, es decir espacio/tiempo.

3.7.2. Ecuaciones dimensionales de $G_N, \hbar, c$

Cuadro 3.1: Valores de las constantes

Magnitud	Valor	Dimensiones
G_N	$6.672 \cdot 10^{-11} \quad m^3kg^{-1}s^{-2}$	$M^{-1}L^3T^{-2}$
$\hbar$	$1.54571596 \cdot 10^{-4} \quad J$	ML^2T^{-1}
c	$2.99792458 \cdot 10^8 \quad m/s$	LT^{-1}

Ahora, cualquier magnitud que pretendamos obtener a partir de estas, le impondremos forma de monomio con las dimensiones que deseemos:

$$\xi_p = G_N^{\alpha}\, c^{\beta}\, \hbar^{\gamma} \tag{3.1}$$
$$[\xi_p] = M^{\delta_M}\, L^{\delta_L}\, T^{\delta_T} \tag{3.2}$$

Así, por ejemplo, para la *longitud de Planck* tendremos $\delta_M = \delta_T = 0,\ \delta_L = 1$. A partir de estas dos ecuaciones estableceremos la combinación de exponentes deseada, con la información resumida del cuadro 3.2.

De las ecuaciones (3.1) y (3.2) y el cuadro (3.2) establecemos las siguientes ecuaciones:

$$-\alpha + \gamma = \delta_M \tag{3.3a}$$
$$3\alpha + \beta + 2\gamma = \delta_L \tag{3.3b}$$
$$-2\alpha - \beta - \gamma = \delta_T \tag{3.3c}$$

Cuadro 3.2: Exponentes de las constantes

	M	L	T
G_N	-1	3	-2
c	0	1	-1
$\hbar$	1	2	-1
ξ_p	δ_M	δ_L	δ_T

Sumando las dos últimas ecuaciones obtenemos una simplificación notable. Obtenemos, junto con la primera, dos ecuaciones referidas a α y γ.

$$-\alpha+\gamma = \delta_M \tag{3.4}$$
$$\alpha+\gamma = \delta_L+\delta_T \tag{3.5}$$

De aquí obtenemos, sumando, el valor de γ:

$$\gamma = \frac{\delta_M+\delta_L+\delta_T}{2} \tag{3.6}$$

y para α:

$$\alpha = \frac{-\delta_M+\delta_L+\delta_T}{2} \tag{3.7}$$

Por último, para el valor de β, obtenemos:

$$\beta = \frac{\delta_M-3\delta_L-5\delta_T}{2} \tag{3.8}$$

Esta es una solución que hemos obtenido analíticamente. Todas estas expresiones se simplificarían mucho más cuando particularizaramos los valores de los δ_i, ya que muchas veces, uno o dos de ellos se anularán. Sin embargo resulta mucho más cómodo idear un sencillo programa que resuelva el sistema de ecuaciones lineales y que, además, calcule el valor del parámetro deseado. Este es el objetivo de nuestro programa. Pero, como ejemplo de resolución analítica ofreceremos la solución analítica para un caso concreto, el caso de la **longitud de Planck**. En este caso, tendremos $\delta_M=\delta_T=0$ $\delta_L=1$. Por tanto, de las expresiones (3.6)–(3.8), tenemos $\gamma=\alpha=\frac{1}{2}$ y $\beta=-\frac{3}{2}$. Luego la longitud de Planck viene dada por:

$$l_p = \left(\frac{\hbar G_N}{c^3}\right)^{1/2} \tag{3.9}$$

3.7.3. Programa de FORTRAN

El programa es muy sencillo. Comienza con la presentación y algunas asignaciones de valores. En las líneas (11-19) se asigna a la matriz de los coeficientes `a` los valores concretos que va a tener, que son los que vienen dados por las ecuaciones. Después se pide al usuario que inserte los valores de la matriz de coeficientes `b`, que vienen dadas por los valores de los distintos exponentes δ_i dados en el desarrollo teórico. Además estos coeficientes se almacenan en otra matriz, porque serán necesarios después (líneas (20-33)). Ya solo queda pedirle a las subrutinas `ludcmp` y `lubksb` que resuelvan el sistema dados por las matrices `a` y `b`. Después se ofrecen al usuario tres informaciones, que son: las dimensiones de las magnitudes calculadas (introducidas éstas dimensiones por el usuario); la fórmula concreta de la magnitud en función de las constantes fundamentales $\hbar$, G_N y c; y por último, el valor concreto de la magnitud, calculada a partir de la fórmula anterior.

Algunos de estos valores se recogen en la tabla 3.3.

Longitud de Planck	$l_p = \sqrt{\frac{G\hbar}{c^3}}$	$4.9 \cdot 10^{-35}$ m
Tiempo de Planck	$t_p = \sqrt{\frac{G\hbar}{c^5}}$	$1.635 \cdot 10^{-43}$ s
Volumen de Planck	$v_p = \sqrt{\frac{G^3\hbar^3}{c^9}} \equiv l_p$	$4.9 \cdot 10^{-35}$ m^3

Cuadro 3.3: Algunos valores de la escala de Planck

Que pueden ser exactamente estas magnitudes no está claro. Se construyen con la intuición de que, al combinar las tres constantes fundamentales c, $\hbar$ y G_N, es decir, la velocidad de la luz, el tamaño de los cuantos de energía $\hbar$ y la intensidad de la gravedad, G_N, se encuentren las escalas a partir de la cual se puedan hacer manifiestos de manera conjunta los efectos cuánticos y la gravedad, entre otras cosas. De manera más general, las escalas a partir de las cuales las leyes del universo, las interacciones que rigen el estado de las partículas, se unifiquen en una de alguna manera. Las teorías más modernas y exóticas que intentan unificar la gravedad y la cuántica hacen estas consideraciones. Por ejemplo, la teoría de la "gravedad cuántica de bucles" (*Átomos del espacio y del tiempo*, Lee Smolin, *Investigación y Ciencia*, pgs.58–67, Marzo 2004) predice que tanto el espacio y el tiempo están cuantizados, y que los cuantos de estos están relacionados, respectivamente, con las longitudes (y áreas y volúmenes) y el tiempo de Planck. Sólo los experimentos darán o quitarán razón.

El código utilizado en el programa es:

```
1         implicit double precision (a-h,o-z)
2         dimension a(3,3), b(3),bb(3),indx(3)
3
4         write(*,100)
```

```
5         write(*,150)
6         write(*,100)
7         G = 6.672d-11
8         hbar = 1.54571596d-34
9         h = hbar*(2.0d0*3.1416d0)
10        c = 2.998d8
11        a(1,1) = -1.0d0
12        a(1,2) = 0.0d0
13        a(1,3) = 1.0d0
14        a(2,1) = 3.0d0
15        a(2,2) = 1.0d0
16        a(2,3) = 2.0d0
17        a(3,1) = -2.0d0
18        a(3,2) = -1.0d0
19        a(3,3) = -1.0d0
20        write(*,*)' '
21        write(*,*)'         Escribe los valores de los indices b(i)'
22        write(*,*)'      b(1) == MASA *** b(2)==LONGITUD *** b(3)==TIEMPO'
23        write(*,*)' '
24        do i=1,3
25        write(*,*)'b(',i,') = '
26        read(*,*)b(i)
27        b(i) = dble(b(i))
28        end do
29
30        do i=1,3
31        bb(i) = b(i)
32        end do
33
34        call ludcmp(a,3,3,indx,d)
35        call lubksb(a,3,3,indx,b)
36        xp = (G**b(1))*(c**b(2))*(h**b(3))
37        write(*,*)' '
38        write(*,*)'La magnitud deseada tiene dimensiones de:'
39        write(*,*)'M^',bb(1),'L^',bb(2),'T^',bb(3)
40        write(*,*)'Y su valor es:'
41        write(*,*)'X_Planck = (G^',b(1),')(c^',b(2),')(h^',b(3),')'
42        write(*,*)' '
43        write(*,*)'X_Planck = ',xp
44
45   100  format(17x,50('*'))
46   150  format(20x,'Calculo dimensional de las escalas de Planck')
47        end
48  *************** SUBRUTINA LUDCMP ************************************
49        SUBROUTINE ludcmp(a,n,np,indx,d)
50
```

3.8. Capacidad térmica en el modelo de Debye

En el modelo de Debye, la dependencia de la capacidad térmica a volumen constante de los aislantes en función de la temperatura absoluta T viene dada por:

$$C_V = 9Nk\left(\frac{T}{\theta}\right)^2 \int_0^{\frac{\theta}{T}} \mathrm{d}x \frac{x^4 e^x}{(e^x-1)^2} \tag{3.10}$$

El objetivo de nuestro programa es calcular el factor $C_v/9Nk$, es decir, para cada temperatura, hallar el valor de la integral y multiplicarla por su correspondiente factor.

El programa comienza con una serie de sentencias `read` que hacen que el usuario inserten los valores de la temperatura máxima a representar, el número de temperaturas que se usarán en el intervalo entre 0 K y y la temperatura máxima, y la temperatura de Debye θ del material (en el programa, `z`). También se inicializan varias variables, así como el "paso" (`step`). Todo esto en las líneas (1-30). A continuación se pone en marcha el archivo donde se escribirán los datos calculados, y también los cálculos. Los cálculos consisten básicamente en hacer la integral, que se hace de dos maneras, mediante la subrutina "preparada" `GABQ` y una casera `simpson`. La integral no incluye nunca el valor inicial $x = 0$ ya que ahí la función se anula, pero el ordenador no sabe distinguirlo. Es decir, inicialmente, se tendría un 0/0, una indeterminación, pero es directo comprobar que el cero del numerador es un cero "más pequeño" que el del denominador, por lo que podemos considerar que el punto $x = 0$ no aporta nada a la integral. Por ello, definimos el valor inferior que se le pasa a las subrutinas de integración es `xll = xl + epsi`, donde `step` es una cantidad muy pequeña. Por otro lado, se puede observar en el programa ((36)) que los cálculos de C_V no se realizan nunca en la temperatura $T = 0$ K. Esto se debe a que la integral, aunque los límites de integración serían 0 e ∞, está acotada, pues:

$$\int_0^{\infty} \mathrm{d}x \frac{x^4 e^x}{(e^x-1)^2} = \frac{4\pi^4}{15}$$

Luego entonces, para $T = 0\ K$, se tiene $C_V(T = 0\ \mathrm{K}) = 0 \cdot \frac{4\pi^4}{15}$. Estos pequeños detalles quedan así resueltos, evitando grandes problemas en los cálculos. Por último destacar que, en la ejecución, conviene no tomar un número muy grande de puntos `n` (conviene tomar $n < 600$) para evitar problemas con los límites de integración, pero esto no es restrictivo, ya que la curva se puede observar perfectamente. La representación de la capacidad térmica C_V frente a la temperatura, para los valores de $\theta = 3250$ K (elemento Ga) y $\theta = 282$ K (elemento As) se encuentran en la figura 3.1.

El listado del código FORTRAN se encuentra a continuación:

```
1 C
2 C NOMBRE: 3-7-mas.f
3 C AUTOR: Miguel Albaladejo Serrao
4 C DESCRIPCION: Programa que calcula un fator de la capacidad termica
```

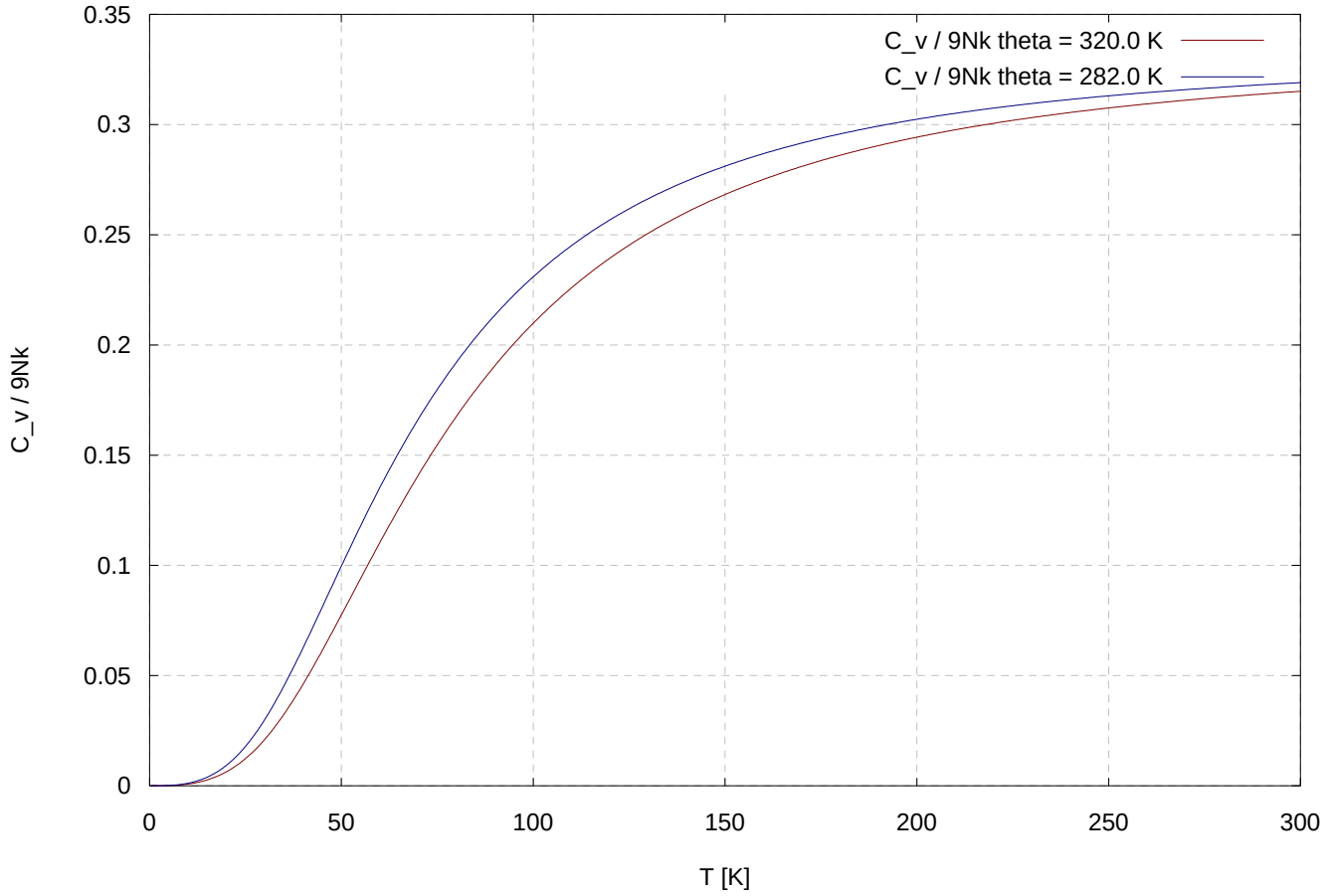

Figura 3.1: Representación de $C_V/9Nk$ frente a T para Ga y As

```
C FECHA: 27-3-2004
C
      implicit real*8 (a-h,o-z)
      external f

      write(*,100)
      write(*,150)
      write(*,100)

      write(*,*)'Temperatura maxima a representar: Tmax(K) = '
      read(*,*)Tmax
      write(*,*)'Numero de temperaturas para calcular Cv/9Nk: n = '
      read(*,*)n
      n = int(n)
      write(*,*)'Temperatura de Debye del aislante z ='
      read(*,*)z
      Tmax = dble(Tmax)
```

```
22        z = dble(z)
23
24        T = 0.0d0
25        tol = 1.0d-6
26        epsi = 1.0d-6
27        xl = 0.0d0
28        xll = xl + epsi
29        m = 1000
30        step = Tmax/dble(n)
31
32        open(1,file='3-7-mas.dat',status='unknown')
33        write(1,300)z,n
34        write(1,250)
35        do i=1,n
36        T = step*dble(i)
37        xd = z/T
38        call GABQ(f,xll,xd,qq1,tol,ier)
39        call simpson(f,xll,xd,m,qq2)
40        q1 = ((T/z)**3.0d0)*qq1
41        q2 = ((1.0d0/xd)**3.0d0)*qq2
42        write(1,200)T,q1,q2
43        end do
44        close(1)
45    100 format(15x,50('*'))
46    150 format(28x,'Capacidad tĆrmica de un gas')
47    200 format(2x,f6.2,3x,f15.10,3x,f15.10)
48    250 format('#',4x,'T(K)',8x,'q1',10x,'q2')
49    300 format('#',4x,'theta = ',f6.2,8x,'numero de temperaturas',i5)
50        end
51
52  *************** funcion a integrar
53        real*8 function f(x)
54        implicit double precision (a-h,o-z)
55        f = (x**4.0d0)*exp(x)/(exp(x)-1.0d0)**2.0d0
56        return
57        end
58
59  ***********************************************************************
60        subroutine simpson (g,xl,xu,npar,simps)
61  ***********************************************************************
62        implicit double precision (a-h,o-z)
63
64        h = (xu - xl)/dble(npar)
65        sp = 0.0d0
66        si = 0.0d0
67        do i=2,npar-2,2
```

```
      sp = sp + g(xl + h*dble(i))
      end do
      do i=1,npar-1,2
      si = si + g(xl + h*dble(i))
      end do
      simps = (h/3.0d0)*( 2.0d0*sp + 4.0d0*si + g(xu) + g(xl) )

      return
      end




C  ************************************************************      GAB00010
C                       SUBROUTINE GABQ                              GAB00020
C  ************************************************************      GAB00030
      SUBROUTINE GABQ(FCT,XL,XU,SUM,TOL,IER)                          GAB00040

```

Tema 4

Ecuaciones diferenciales

4.7. Ley de enfriamiento de Newton

Se trata de resolver numéricamente la ecuación diferencial que rige el enfriamiento de un cuerpo en su entorno, ecuación debida a NEWTON, que es:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -r(T - T_\mathrm{e}) \tag{4.1}$$

En la anterior expresión, T es la temperatura del cuerpo, T_e es la temperatura del entorno, y r es un parámetro propio del cuerpo o sistema. Nosotros usaremos como temperatura inicial $T_0 = 100°\mathrm{C}$ y como valor de r usaremos $r = 0.1\mathrm{min}^{-1} = 0.1/60\mathrm{s}^{-1}$, y la temperatura del entorno $T_\mathrm{e} = 9°\mathrm{C}$. La solución analítica de esta ecuación sería:

$$T = T_\mathrm{e} + (T_0 - T_\mathrm{e})e^{-rt} \tag{4.2}$$

En el programa se definen estos parámetros iniciales y algunos valores más, en las líneas (1-14), incluyendo las declaraciones de las funciones (en forma de variables `dimension`) que se usarán con la subrutina `rk4`. En las líneas (16-18) se abre y se encabeza el archivo en el que se escribirán los valores de la temperatura para cada instante de tiempo, hallados numéricamente. El núcleo del programa lo consituyen las líneas (18-26). En ellas se escriben los valores de `x` (el tiempo), `y(1)` (la temperatura) y `dif` (la diferencia del valor calculado y el valor teórico). Este último valor se define a partir de la función (`function`) `teo(x)`, que define el valor de la temperatura para cada instante de tiempo, mediante la ecuación 4.1. Después se llama a las subrutinas `derivs` (que define las ecuaciones que se han de resolver) y la subrutina `rk4` (que las resuelve). Con los valores obtenidos para la temperatura, se avanza en el valor de `x`, que representa el tiempo, ya que todo esto se hace en un ciclo `do...end do`, que, tras esto, se vuelve iniciar. Las líneas (28-33) son solo el final del programa y la definición de un formato. A partir de ahí se definen la función `teo`, y las subrutinas `derivs` y `rk4`.

Todos los datos se recogen en el archivo *4-7-mas.dat*. Estos datos, así como los valores teóricos, están representados en la figura 4.1. Se puede observar que ambos valores se

solapan. Esto también se puede comprobar ejecutando el programa y haciendo que se escriban los valores del parámetro `dif`, calculado mediante la diferencia en valor absoluto entre el valor calculado y el valor teórico, y que toma valores muy pequeños. Esto se consigue además sin pedirle al programa cálculos muy precisos, pues son solo 5000 pasos para 10 minutos (600 segundos). La temperatura final alcanzada es $T = 42.47°\text{C}$.

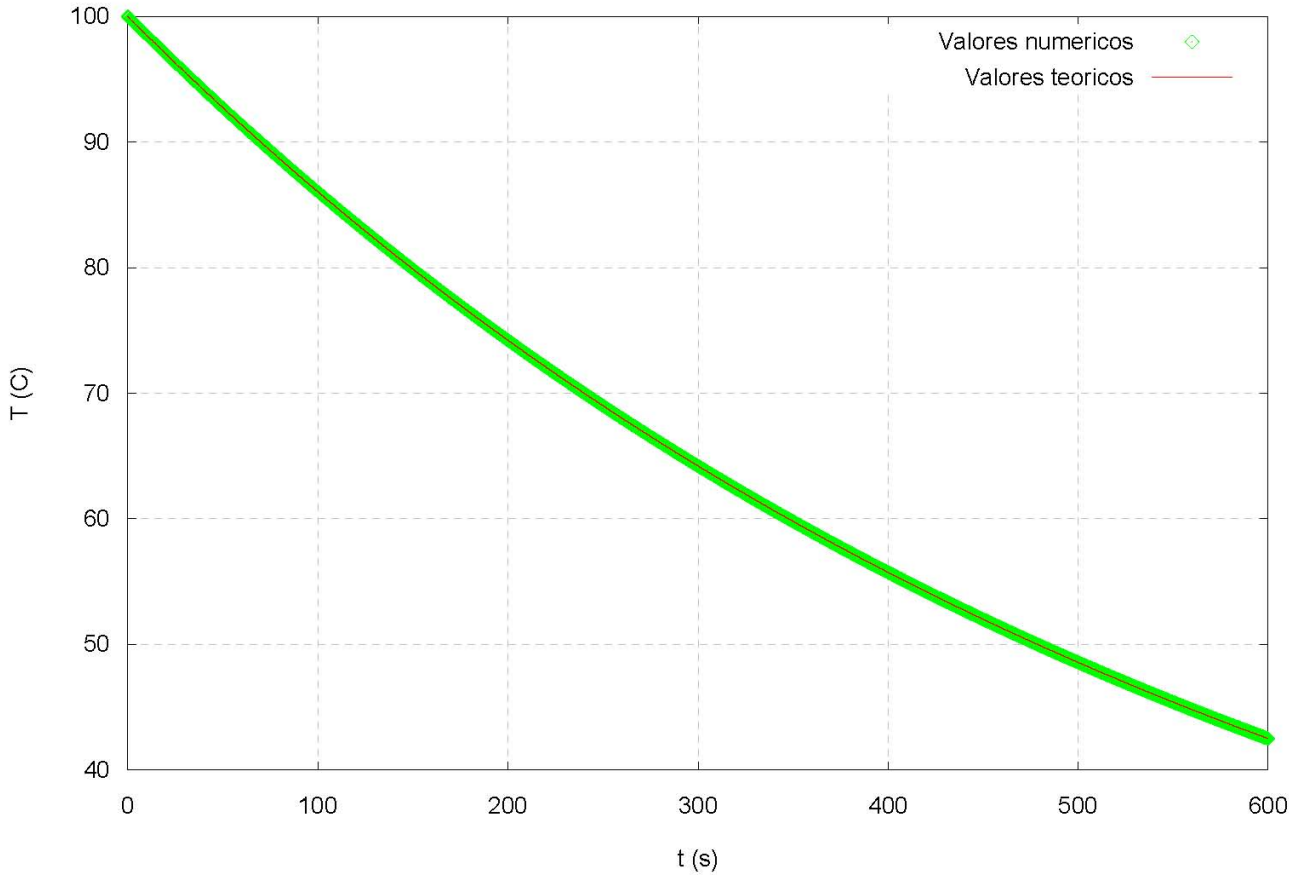

Figura 4.1: Representación de los valores teóricos y numéricos de la temperatura del cuerpo. Ambos valores se solapan.

El listado del código FORTRAN se encuentra a continuación:

```
1         parameter (n=1)
2         dimension y(n),dydx(n),yout(n)
3         external derivs
4         common /ene/nn
5         common /datos/r,Te,T0
6         nn=n
7         r = 0.1/60.0 ! r = 0.1 min^{-1} = 0.1 / 60 s^{-1}
8         Te = 9.0
9         T0 = 100.0
```

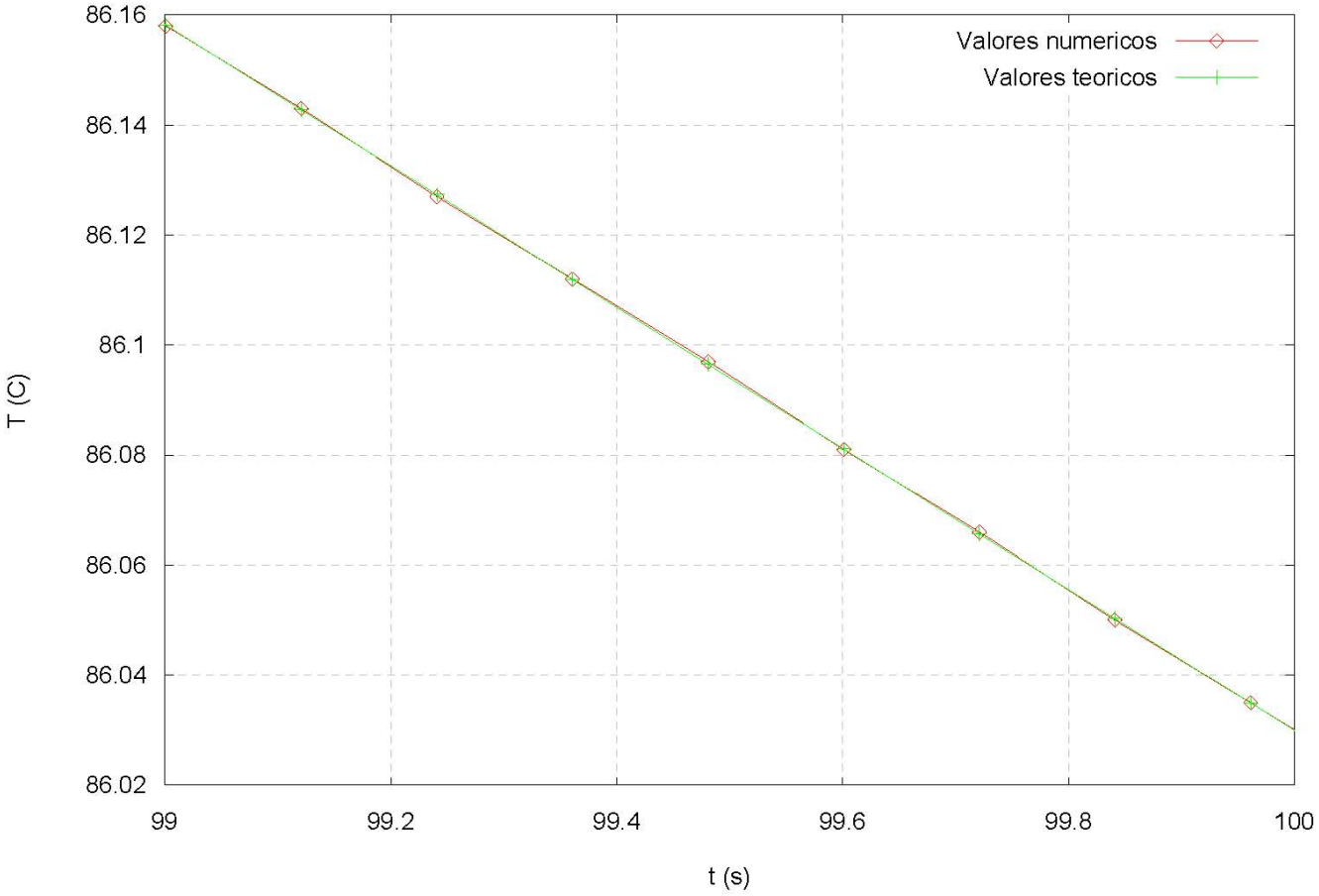

Figura 4.2: Detalles entre los instantes 99 y 100 s. Aquí se observan diferencias.

```
10          y(1) = T0
11          tmax = 10.0 ! Tiempo maximo en minutos
12          npasos=5000
13          dif = 0.0
14          step = tmax*60.0/real(npasos) ! Definicion del paso. Conversion a segundos
15          x=0.0 !Inicio del tiempo a cero segundos.
16          open(1,file='4-7-mas.dat')
17          write(1,*)'# T_0 (C)',y(1),' T_e (C)=',Te,' r(s^{-1}) =',r
18          write(1,*)'#     t(s)      T(C)         T(C) (teorico)'
19          do i=0,npasos + 1
20             write(1,100) x,y(1),teo(x)
21             call derivs(x,y,dydx)
22             call rk4(y,dydx,n,x,step,yout,derivs)
23             x=x+step
24             y(1)=yout(1)
25             dif = (abs(teo(x) - y(1))/teo(x))
26          end do
```

```
27        close(1)
28        do i=1,10
29          write(*,*) ' '
30        end do
31        write(*,*)'                                          THE END'
32   100  format(2(2x,f9.3),4x,f10.5)
33        end
34
35        function teo(x)
36        common /datos/r,Te,T0
37        teo = Te + (T0 -Te)*exp(-r*x)
38        return
39        end
40
41  ************************************************************************
42        subroutine derivs(x,y,dy)
43  ************************************************************************
44        dimension y(nn),dy(nn)
45        common /ene/nn
46        common /datos/r,Te,T0
47        dy(1)= -r*(y(1) - Te)
48        return
49        end
50  ************************************************************************
51        subroutine rk4(y,dydx,n,x,h,yout,derivs)
52  ************************************************************************
53        integer n,nmax
54        real h,x,dydx(n),y(n),yout(n)
55        external derivs
56        parameter (nmax=50)
57        integer i
58        real h6,hh,xh,dym(nmax),dyt(nmax),yt(nmax)
59        hh=h*0.5
60        h6=h/6.
61        xh=x+hh
62        do 11 i=1,n
63          yt(i)=y(i)+hh*dydx(i)
64   11   continue
65        call derivs(xh,yt,dyt)
66        do 12 i=1,n
67          yt(i)=y(i)+hh*dyt(i)
68   12   continue
69        call derivs(xh,yt,dym)
70        do 13 i=1,n
71          yt(i)=y(i)+h*dym(i)
72          dym(i)=dyt(i)+dym(i)
```

```
13    continue
      call derivs(x+h,yt,dyt)
      do 14 i=1,n
        yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.*dym(i))
14    continue
      return
      end
```

4.8. Movimiento planetario

Aquí, como en el ejercicio anterior, partimos de una ley de Newton, pero bastante más importante, la ley de la Gravitación Universal. La fuerza de atracción entre dos cuerpos de masas M y m viene dada por:

$$F = -\frac{GMm}{r^2} \tag{4.3}$$

Nosotros estudiaremos el caso del sistema solar, en que usamos un sistema de unidades tal que $GM = 1$, el Sol está en el cetro del sistema de coordenadas, y los valores iniciales son $x = 0.5$, $y = 0$, $v_x = 0$, y $v_y = 1.63$. Para ello estudiaremos las ecuaciones diferenciales dadas mediante la anterior ecuación y la segunda ley de Newton en los ejes x e y. Además, necesitamos descomponer las ecuaciones doferenciales en distintas ecuaciones de primer grado. Nuestra descomposición (tanto vectorial como en ecuaciones de primer grado) es la siguiente:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v_x \tag{4.4}$$

$$\frac{\mathrm{d}v_x}{\mathrm{d}t} = -a\frac{x}{\sqrt{x^2+y^2}} \tag{4.5}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = v_y \tag{4.6}$$

$$\frac{\mathrm{d}v_y}{\mathrm{d}t} = -a\frac{y}{\sqrt{x^2+y^2}} \tag{4.7}$$

$$a = \frac{1}{x^2+y^2} \tag{4.8}$$

La aparición de los segundos factores en las expresiones de las derivadas de las velocidades se debe a la descomposición vectorial y a la aparición de senos y cosenos. Para evitar trabajar con estas funciones trigonométricas, utilizamos su expresión a partir de las coordenadas x e y (ver figura 4.3).

Estas ecuaciones son las que pasaremos a la subrutina `derivs`, con la siguiente asignación: `y(1)` $= x$, `y(2)` $= y$, `y(3)` $= v_x$ e `y(4)` $= v_y$. El programa consiste fundamentalmente en un ciclo de tantos pasos como se deseen, en el que se llama a la subrutina

`derivs` para pasar a la subrutina `rk4` las funciones adecuadas. Las condiciones iniciales son las que se explicitan en el enunciado del problema.

Para comprobar que efectivamente se cumple que las sucesivas órbitas se superponen, aunque podemos hacerlo "a ojo", comprobando que número de pasos es necesario para hacer una órbita completa, podemos también calcularlo analíticamente mediante la tercera ley de Kepler, que afirma que el período τ cumple, si a es el semieje menor de la órbita, la siguiente relación:

$$\tau^2 = \frac{4\pi^2 a^3}{G(M+m)} \approx \frac{4\pi^2 a^3}{GM} = 4\pi^2 a^3 \tag{4.9}$$

Se puede calcular que el semieje viene dado por $a = \frac{GMm}{2|E|} = \frac{1}{2|E|/m}$. Posteriormente lo calcularemos analíticamente, pero podemos adelantar que $|E|/m = 0.6715$, por lo que $a = 0,7446$, y, por tanto, $\tau = 4.037$. Como usamos un paso `h` $h = 1.0\ 10^{-3}$, necesitaremos aproximadamente 4037 pasos. Efectivamente, vemos que un número menor de esos pasos hace que la órbita se quede abierta. Si hacemos que el programa se ejecute con un valor de `npasos` de 50000, vemos que las sucesivas órbitas se superponen sin ningún problema (veáse figura 4.4). Esto se hace modificando la línea (22).

Para comprobar que se cumple la conservación de la energía, calculamos la energía por unidad de masa E/m a partir de la relación (en la que tenemos en cuenta que $GM = 1$):

$$\frac{E}{m} = \frac{1}{2}v^2 - \frac{1}{r} \tag{4.10}$$

Es evidente que el primer valor de este parámetro, habida cuenta de que inicialmente $v = 1.63$ y $r = 0.5$, es (en nuestras unidades) $E/m = -0.67155$. Como al resolver las ecuaciones obtenemos las velocidades v_x y v_y, así como las coordenadas x e y, podemos calcular $v^2 = v_x^2 + v_y^2$ y $r = \sqrt{x^2+y^2}$ (se hace en las líneas (43-44), lo que permite calcular la energía en cada instante (línea (48)). Esta energía se escribe, para cada instante, en el archivo *4-8-mas-compr.dat* (líneas (24,49)). Si lo representamos, vemos claramente que este valor no varía, sino que oscila (por redondeo) entre los valores -0.6715 y -0.6716, en acuerdo con lo expresado al inicio del párrafo. Es decir, *la energía se conserva.*

Para comprobar finalmente que se cumple la segunda ley de Kepler, recurrimos a la forma diferencial de esta:

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{1}{2}r^2\frac{\mathrm{d}\theta}{\mathrm{d}t} \tag{4.11}$$

En el punto inicial, que es un punto de distancia mínima, no existe velocidad radial, $\dot{r} = \frac{\mathrm{d}r}{\mathrm{d}t} = 0$, precisamente porque en ese punto r es mínimo; es decir, la velocidad es únicamente angular. Por tanto, se cumple en este punto que $\frac{\mathrm{d}\theta}{\mathrm{d}t} = v/r$. Es decir, inicialmente se cumple $\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{1}{2}vr$, donde estos valores son los iniciales. Pero este valor ha de permanecer constante, luego calcularlo en ese punto nos da el valor correcto. Tenemos que $\frac{\mathrm{d}A}{\mathrm{d}t} = 0.4075$. Ahora bien, ahora tenemos que calcular esto en cada punto para nuestro programa. Para ello, debemos tener en cuenta que el diferencial de área $\mathrm{d}A$ representa la

variación de área que el radio-vector barre en cada paso. Si consideramos que con el radio-vector se forma, entre dos instantes de tiempo, un triángulo (que, infinitesimalmente, podemos considerar rectángulo; ver figura 4.5), podemos calcular esta variación de área como el área de dicho triángulo $\mathrm{d}A = \frac{1}{2}bh$. La base b vendrá dada por el radio-vector en el instante anterior, y la altura h la calculamos teniendo en cuenta lo que varían las coordenadas x e y. Podemos decir que la altura h sería el módulo del vector que fuese desde la posición (x, y) en el instante anterior hasta la posición posterior. Todos estos cálculos se realizan en las líneas (42-47). Hemos calculado con esto el valor de $\mathrm{d}A$ en cada instante. Por otro lado, como el bucle se realiza con un paso $\mathrm{d}t$, dado por `h`, que es constante y de valor $\mathrm{d}t = 10^{-3}$, se tiene que también $\mathrm{d}A$ ha de ser constante. Como antes hemos calculado $\frac{\mathrm{d}A}{\mathrm{d}t}$, podemos concluír que $\mathrm{d}A$ ha de valer teóricamente $\mathrm{d}A = 0.0004075$. Como en nuestro progama calculamos $\mathrm{d}A$ (el parámetro `vara`), y lo escribimos en el archivo *4-8-mas-compr.dat* (líneas (47,49)), podemos comprobar que el valor de `vara` se mantiene constante e igual a 0.0004, por lo que podemos deducir que también *se cumple la segunda ley de Kepler*.

El listado del código FORTRAN se encuentra a continuación:

```
1
2         parameter (n=4) !funciones
3         dimension y(n),dydx(n),yout(n)
4         external derivs
5         common /ene/nn
6         common /datos/x0,y0,vx0,vy0
7         nn=n
8         x0 = 0.5
9         y0 = 0.0
10        vx0 = 0.0
11        vy0 = 1.63
12        x=0.0
13
14  C     Asignacion de las variables x,y,vx,vy a los elementos del array y(n)
15  C         y(1) == x     y(2) == y      y(3) == vx       y(4) == vy
16  C     Valores iniciales
17        y(1) = x0
18        y(2) = y0
19        y(3) = vx0
20        y(4) = vy0
21        h = 1.0e-3
22        npasos = 20000
23        k=1
24        A=0.0
25        open(1,file='4-8-mas.dat',status='unknown')
26        open(2,file='4-8-mas-compr.dat',status='unknown')
27        open(3,file='4-8-mas-A.dat',status='unknown')
28
```

```
29        do i=1,npasos
30
31        write(1,1000) x,y(1),y(2),y(3),y(4)
32
33        y1old = y(1)
34        y2old = y(2)
35        thetold = abs(atan(abs(y(1)/y(2))))
36
37  *     Subrutina "derivs" (ecuaciones diferenciales)
38           call derivs(x,y,dydx)
39  *     Subrutina "rk4" (las soluciona)
40           call rk4(y,dydx,n,x,h,yout,derivs)
41
42           x=x+h
43           do j=1,n
44              y(j)=yout(j)
45           end do
46           rold = sqrt(y1old*y1old + y2old*y2old)
47           v = sqrt(y(3)**2 + y(4)**2)
48           rr = sqrt(y(1)*y(1) + y(2)*y(2))
49           base =  rr
50           rx = y2old*(y(1)-y(2))/(x1old + y1old)
51           ry = y1old*(y(1)-y(2))/(x1old + y1old)
52           rrx = y(1) - rx
53           rry = y(2) + ry
54           rrr = sqrt(rrx**2 + rry**2)
55           height=sqrt(rx**2 + ry**2)
56           !height = sqrt( (y(1) - y1old)**2 + (y(2) - y2old)**2 )
57           thet = abs(atan(abs(y(1)/y(2))))
58           vara = 0.5*base**2*abs(thetold - thet)
59           varaa= 0.5*rrr*height
60           varaaa = 0.5*(y(1)*y(4) - y(2)*y(3))
61           ensinm = 0.5*(v**2) - 1/rr
62
63           A = A + varaa
64           if (i.eq.10*k) then
65             write(3,3000)x,A
66             A = 0.0
67             k = k+1
68             continue
69           else
70             continue
71           end if
72
73
74        write(2,2000) x,y(1),y(2),vara,varaa,varaaa
```

```
75
76          end do
77          close(1)
78          close(2)
79          close(3)
80
81          do i=1,10
82            write(*,*) ' '
83          end do
84          write(*,*)'                                        THE END'
85    1000 format(5(3x,f10.4))
86    2000 format(6(3x,g13.5))
87    3000 format(2(5x,g15.7))
88          end
89    C          y(1) == x     y(2) == y      y(3) == vx       y(4) == vy
90
91    ************************************************************************
92          subroutine derivs(x,y,dy)
93    ************************************************************************
94          dimension y(nn),dy(nn)
95          common /ene/nn
96          r = sqrt(y(1)*y(1) + y(2)*y(2))
97          a = 1.0/(r**2)
98          dy(1)=y(3)
99          dy(2)=y(4)
100         dy(3)=-a*(y(1)/r)
101         dy(4)=-a*(y(2)/r)
102         return
103         end
104   ************************************************************************
105         subroutine rk4(y,dydx,n,x,h,yout,derivs)
106   *     Numerical Recipes (2a ed.): p. 706
107   ************************************************************************
108         integer n,nmax
109         real h,x,dydx(n),y(n),yout(n)
110         external derivs
111         parameter (nmax=50)
112         integer i
113         real h6,hh,xh,dym(nmax),dyt(nmax),yt(nmax)
114         hh=h*0.5
115         h6=h/6.
116         xh=x+hh
117         do 11 i=1,n
118           yt(i)=y(i)+hh*dydx(i)
119   11    continue
120         call derivs(xh,yt,dyt)
```

```
121         do 12 i=1,n
122           yt(i)=y(i)+hh*dyt(i)
123   12    continue
124         call derivs(xh,yt,dym)
125         do 13 i=1,n
126           yt(i)=y(i)+h*dym(i)
127           dym(i)=dyt(i)+dym(i)
128   13    continue
129         call derivs(x+h,yt,dyt)
130         do 14 i=1,n
131           yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.*dym(i))
132   14    continue
133         return
134         end
135
```

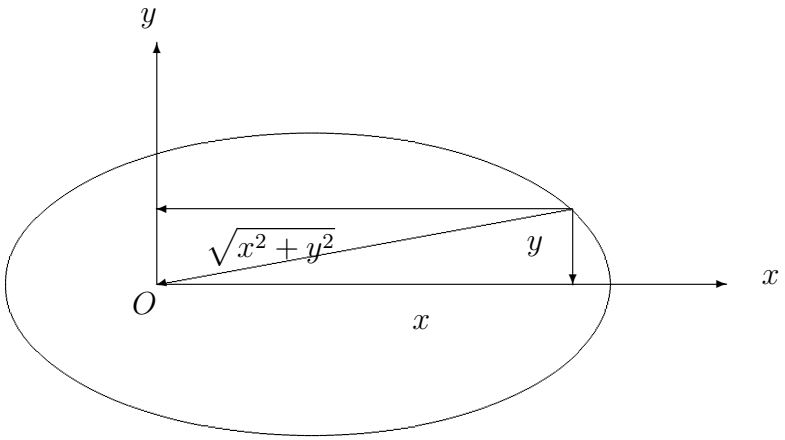

Figura 4.3: Descomposición vectorial

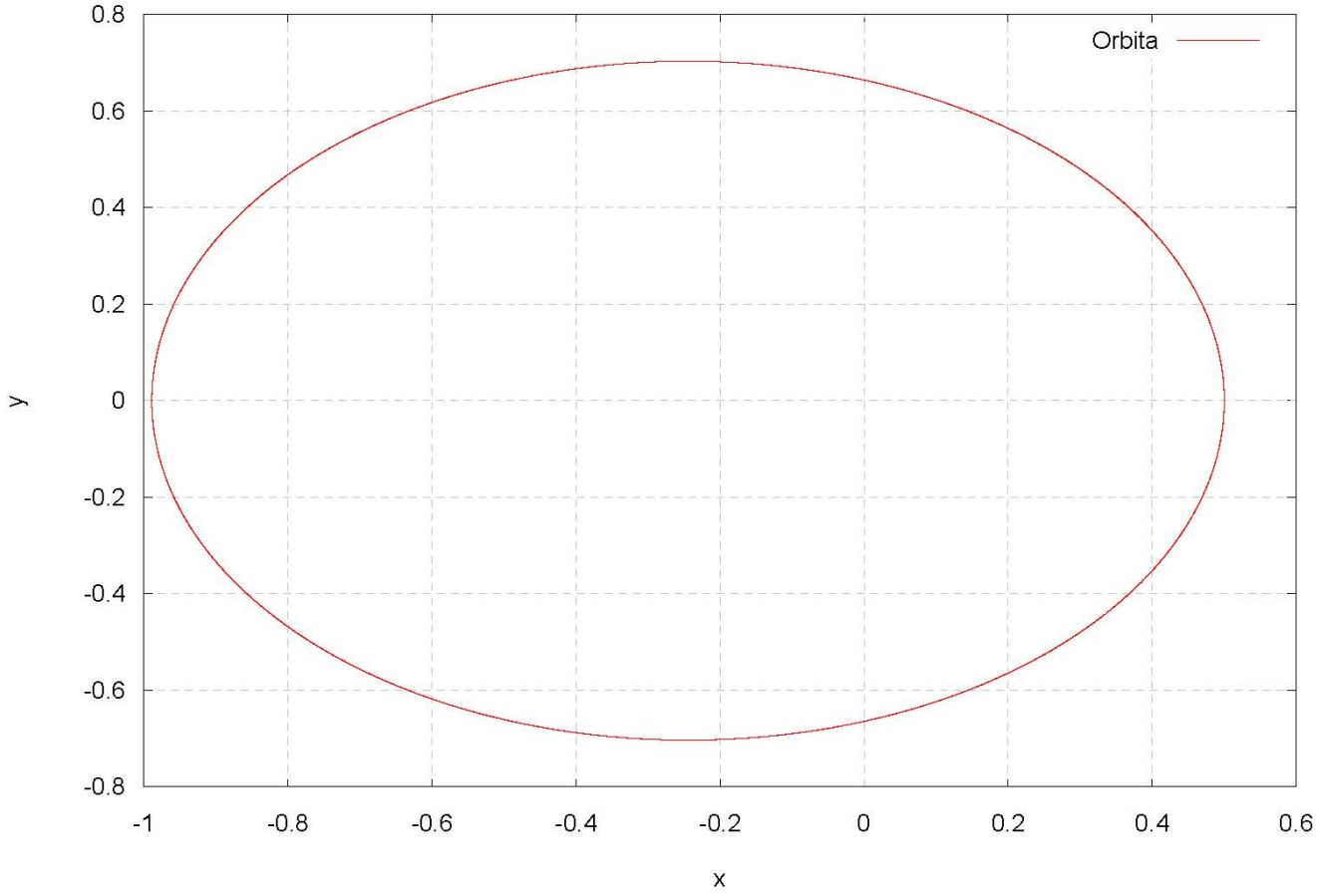

Figura 4.4: Representación de la órbita para 50000 pasos, esto es, aproximadamente, 12 "vueltas".

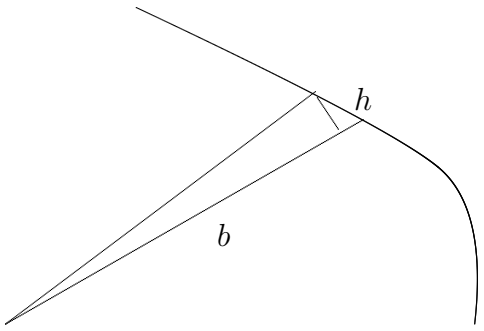

Figura 4.5: Esquema para la segunda ley de Kepler.

Tema 5

Problemas de condiciones de contorno y valores propios

Apéndice al problema anterior sobre el movimiento planetario. Momento angular y segunda ley de Kepler

En la sección 4.8 calculábamos la variación de área por unidad de tiempo en cada variación de tiempo "infinitesimal". Pero al calcularla no nos dábamos cuenta de que dichos desplazamientos no eran realmente infinitesimales. Al calcular el área en un desplazamiento, siempre quedaba un resto de área no computada. Pero este resto de área, lejos de ser algo más o menos aleatorio, resulta que no lo es, y va variando de forma sinusoidal, de manera que la variación de área en cada instante también lo parecía. Esto se debe a que, al tratarse de una elipse, el área que no computamos va creciendo hasta que llega al primer cuarto de la órbita, donde empieza a bajar hasta llegar a la mitad, y así sucesivamente, de forma simétrica. Por muchas maneras se puede intentar paliar este efecto, pero es difícil de solucionar el error si uno sigue intentando calcular el área barrida de esta manera, sin involucrar el ángulo o la velocidad. Incluso haciendo promedios temporales cada cierto número de pasos o cada cierta cantidad de tiempo. Sin duda, esto se debe a la particular geometría del problema.

Para solventar el problema de otro modo, podríamos añadir unas líneas que modificasen los cálculos que en la sección 4.8 hacíamos, sin modificar la esencia y sin entrar en el cálculo del momento angular directamente. Esas líneas podrían ser:

```
1  thet = abs(atan(abs(y(1)/y(2))))
2  vara = 0.5*base**2*abs(thetold - thet)
```

Siendo `theold` una variable que definimos de forma análoga pero con datos anteriores. Esta solución funciona perfectamente en todos los puntos de la órbita, salvo en aquellos en los que, al definir los ángulos, se hace alguna división por cero (en los ejes, claramente). Esto se puede observar en la gráfica 5.1.

Pero, sin duda, la mejor manera de comprobar que efectivamente se cumple la segunda

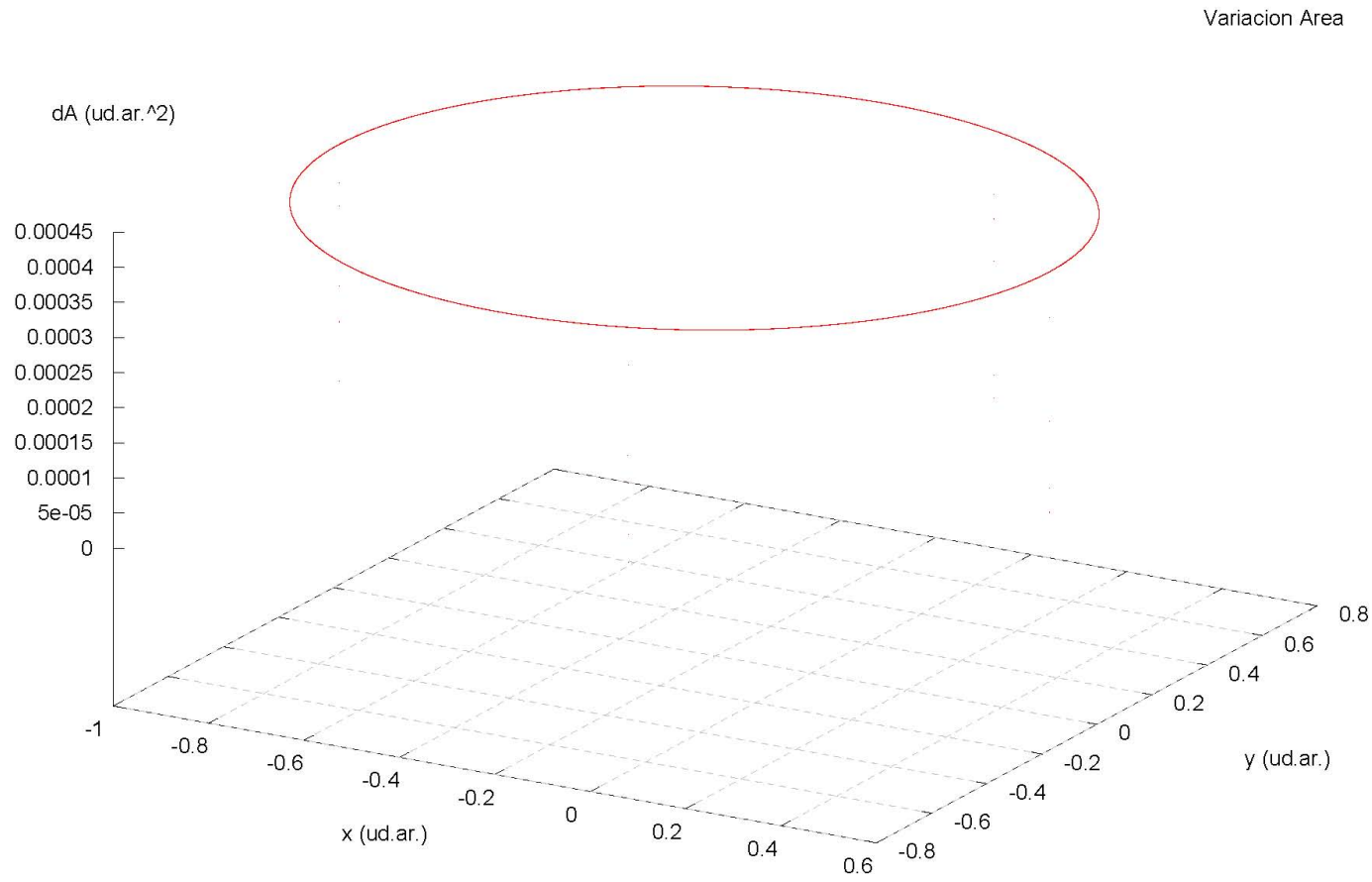

Figura 5.1: Representación del momento angular de otra forma

ley de Kepler es calcular directamente el momento angular, con el que está relacionada la ley, que es la forma en que abordamos el problema en la sección 5.7.

5.5. Problema de autovalores

Queremos calcular los principales autovalores de la ecuación diferencial:

$$y'' + 2y' + k^2 y = 0 \tag{5.1a}$$

$$y(0) = y(1) = 0 \tag{5.1b}$$

Entendemos por principales los de menor magnitud. Los valores analíticos son $k = \sqrt{1+\pi^2}$. Como queremos calcular el autovalor de menor magnitud debemos hallar la inversa de la matriz mediante la que caractericemos la ecuación. Esta ecuación, discretizada mediante fórmulas centrales para las derivadas, que usen a lo sumo el punto central y los puntos inmediatamente anterior y posterior, puede ser expresada matricialmente como:

$$\frac{1}{h^2}\begin{pmatrix} -2 & 1+h/2 & 0 & \cdots & 0 \\ 1-h/2 & -2 & 1+h/2 & \vdots & \vdots \\ 0 & \cdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1+h/2 \\ 0 & \cdots & \cdots & 1-h/2 & -2 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix} = \lambda \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{pmatrix} \tag{5.2}$$

En la anterior ecuación matricial hemos tenido en cuenta que las condiciones de contorno son las (5.1b), y hemos descompuesto el intervalo $(0,1)$ en los puntos $x_0 = 0, x_1, \ldots, x_{N-1}, x_N = 1$. El paso en la descomposición lo llamamos h. Si llamamos A a la matriz que aparece en la ecuación (5.2), se cumplirá que:

$$Ay_i = \lambda_i y_i \Rightarrow A^{-1}y_i = \frac{1}{\lambda_i} y_i \tag{5.3}$$

Como queremos los autovalores de menor magnitud, mediante este proceso conseguimos que sean los autovalores de mayor magnitud de una matriz que representa la misma ecuación diferencial. Para conseguir la inversión matricial usamos la descomposición LU, y a partir de ella, mediante métodos vistos en temas anteriores, conseguimos la matriz inversa. Esto se realiza en la Sección II del programa (líneas (44-59)), ya que en la sección I (líneas (15-43))se ha definido la matriz problema A. Para el cálculo del autovalor aplicamos el eficaz método de la potencia, en la Sección III (líneas (90-65)).

Por último hemos de convertir el autovalor que obtenemos, λ, en su equivalente en función de k. Si observamos las ecuaciones (5.3) y (5.1a), vemos que ha de ser:

$$\lambda = -k^2 \tag{5.4}$$

El programa hace esta transformación directamente, en la sección IV (líneas (96-116)), de forma que tenemos que comparar el valor teórico $\lambda_{teo} = 1 + \pi^2 \approx 10.8696$.

Ahora queremos calcular, a partir de dos resultados obtenidos con el programa mediante la interpolación de Richardson, un valor mejor para el autovalor. Lo hacemos mediante la ecuación:

$$\lambda_{\text{mej}} = \frac{r^n \lambda_{h'} - \lambda_h}{r^n - 1} \tag{5.5}$$

Donde $h' < h$, n es el orden de las aproximaciones hechas, y $r = \frac{h}{h'}$. Una lista de resultados obtenidos se muestra en la tabla 5.1.

h	λ_h	k_h
0.100	-0.0997345	10.02662
0.010	-0.0988365	10.11772
0.005	-0.0988558	10.11574

Cuadro 5.1: Algunos autovalores obtenidos para distintos valores del paso h.

Tomando los dos últimos valores, mediante la mejora de Richardson, obtenemos $\lambda_{\text{mej}} = -0.09883$, por lo que tenemos que $k_{\text{mej}} = 10.11838$, lo que tampoco supone una gran mejora frente a lo que teníamos. El resultado obtenido tiene un error, frente al exacto, del 6.9 %.

El listado del código FORTRAN se encuentra a continuación:

```
1  C
2  C NOMBRE: 5-5-mas.f
3  C AUTOR: Miguel Albaladejo Serrano
4  C DESCRIPCION: Calculo del menor autovalor en y'' + 2y' + k**2 y = 0
5  C FECHA: 09-05-04
6  C
7
8        parameter (n=100)
9        dimension a(1:n-1,1:n-1),aa(1:n-1,1:n-1),y(0:n),ynew(1:n-1)
10       dimension b(1:n-1,1:n-1),bb(1:n-1),indx(1:n-1)
11       real lambda
12       niter=500
13       exact=10.8696
14       h=1.0/real(n)
15 ************************************************************************
16 ****** SECCION I: Definicion de la matriz a ***************************
17 ************************************************************************
18       do i=1,n-1
19       do j=1,n-1
20        a(i,j) = 0.0
21       end do
```

```
22        end do
23
24        do i=1,n-1
25         a(i,i) = -2.0
26         a(i,i+1) = (1.0 + h/2.0)
27         a(i+1,i) = (1.0 - h/2.0)
28        end do
29
30        do i=1,n-1
31        do j=1,n-1
32        a(i,j) = a(i,j)/(h**2)
33        end do
34        end do
35
36        do i=1,n-1
37        do j=1,n-1
38         aa(i,j) = a(i,j)
39        end do
40        end do
41
42
43
44  ************************************************************************
45  ****** SECCION II: Inversion de la matriz a : inversa es b *************
46  ************************************************************************
47        call ludcmp(aa,n-1,n-1,indx,d)
48
49        do j=1,n-1
50         do i=1,n-1
51            bb(i) = 0.0
52         end do
53            bb(j) = 1.0
54
55        call lubksb(aa,n-1,n-1,indx,bb)
56          do i=1,n-1
57             b(i,j) = bb(i)
58          end do
59        end do
60  ************************************************************************
61  ******** SECCION III: Calculo del autovalor ****************************
62  ************************************************************************
63          y(0)= 0.0
64          y(n)= 0.0
65          do i=1,n-1
66            y(i) = -1.0**n
67          end do
```

```
68
69          do k=1,niter
70
71    C      inicializo los ynew
72           do i=1,n-1
73             ynew(i) = 0.0
74           end do
75
76    C      los y(0) e y(n) han de ser cero, condiciones de contorno
77
78    C      multiplicacion de matrices
79           do i=1,n-1
80           do j=1,n-1
81             ynew(i) = ynew(i) + b(i,j)*y(j)
82           end do
83           end do
84           sup = 0.0
85    C      busco el valor mas grande
86           do i=1,n-1
87             if(abs(ynew(i)).gt.abs(sup)) sup=ynew(i)
88           end do
89
90           do i=1,n-1
91             y(i) = ynew(i)/sup
92           end do
93           y(0)= 0.0
94           y(n)= 0.0
95          end do
96    ***************************************************************************
97    ****** SECCION IV: Escritura de solucion/autovalor *********************
98    ***************************************************************************
99
100         open (2,file='5-5-mas-b.dat',status='unknown')
101         write(2,*)'# autovalor: ', sup
102         do j=0,n
103         write(2,2000) j*h,y(j)
104         end do
105         lambda = -1.0/sup
106
107
108         write(*,*)'Con h=',h,' el autovalor sin transformar es =',sup
109         write(*,*)'Con h=',h,' el autovalor es a.v.=',lambda
110         write(*,*)'El valor exacto es: exacto=',exact
111         write(*,*)'El error relativo es e=',abs(exact-lambda)/exact
112
113    2000 format(3x,2(2x,e10.4))
```

```
114         close(1)
115         close(2)
116         end
117
118   ***********************************************************************
119   ******  APENDICE: Subrutinas para LU  *********************************
120   ***********************************************************************
121
122         SUBROUTINE ludcmp(a,n,np,indx,d)
123         INTEGER n,np,indx(n),NMAX
124         REAL d,a(np,np),TINY
125         PARAMETER (NMAX=500,TINY=1.0e-20)
126         INTEGER i,imax,j,k
127         REAL aamax,dum,sum,vv(NMAX)
128         d=1.
129         do 12 i=1,n
130           aamax=0.
131           do 11 j=1,n
132             if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
133   11      continue
134           if (aamax.eq.0.) pause 'singular matrix in ludcmp'
135           vv(i)=1./aamax
136   12    continue
137         do 19 j=1,n
138           do 14 i=1,j-1
139             sum=a(i,j)
140             do 13 k=1,i-1
141               sum=sum-a(i,k)*a(k,j)
142   13        continue
143             a(i,j)=sum
144   14      continue
145           aamax=0.
146           do 16 i=j,n
147             sum=a(i,j)
148             do 15 k=1,j-1
149               sum=sum-a(i,k)*a(k,j)
150   15        continue
151             a(i,j)=sum
152             dum=vv(i)*abs(sum)
153             if (dum.ge.aamax) then
154               imax=i
155               aamax=dum
156             endif
157   16      continue
158           if (j.ne.imax)then
159             do 17 k=1,n
```

```
160               dum=a(imax,k)
161               a(imax,k)=a(j,k)
162               a(j,k)=dum
163   17        continue
164             d=-d
165             vv(imax)=vv(j)
166           endif
167           indx(j)=imax
168           if(a(j,j).eq.0.)a(j,j)=TINY
169           if(j.ne.n)then
170             dum=1./a(j,j)
171             do 18 i=j+1,n
172               a(i,j)=a(i,j)*dum
173   18        continue
174           endif
175   19    continue
176         return
177         END
178         SUBROUTINE lubksb(a,n,np,indx,b)
179         INTEGER n,np,indx(n)
180         REAL a(np,np),b(n)
181         INTEGER i,ii,j,ll
182         REAL sum
183         ii=0
184         do 12 i=1,n
185           ll=indx(i)
186           sum=b(ll)
187           b(ll)=b(i)
188           if (ii.ne.0)then
189             do 11 j=ii,i-1
190               sum=sum-a(i,j)*b(j)
191   11        continue
192           else if (sum.ne.0.) then
193             ii=i
194           endif
195           b(i)=sum
196   12    continue
197         do 14 i=n,1,-1
198           sum=b(i)
199           do 13 j=i+1,n
200             sum=sum-a(i,j)*b(j)
201   13      continue
202           b(i)=sum/a(i,i)
203   14    continue
204         return
205         END
```

206

5.7. (Aún más) Movimiento planetario

En este problema queremos realizar un cálculo muy parecido al del tema anterior, con estos añadidos:

- Usamos dimensiones reales en el problema
- No conocemos la velocidad del planeta en ningún punto, solo algunos datos como su excentricidad, el perihelio, etc...

Por ello, debemos aplicar el método del disparo para obtener alguna condición de contorno más, aparte de las que nos dan para el perihelio. Para ello actuaremos de la siguiente manera. Haremos los cálculos de cuartos de órbita del planeta, sin mucha precisión en el paso, pues tampoco es necesario. En ese punto tomaremos el tiempo invertido, y calculamos de forma aproximada el período de la órbita en esas condiciones. En función de que se parezca más o menos al período real o no, probaremos con otra velocidad, usando un método inspirado en el método de bisección, cambiando los límites laterales para la velocidad[1]. Todo esto se realiza en las líneas (43-87). Anteriormente no se ha hecho sino definir e inicializar parámetros y variables. Una vez que tenemos un valor adecuado de la velocidad inicial, hacemos los cálculos de nuevo, esta vez con la precisión deseada, e incluyendo esta vez los cálculos de la energía y el momento angular, así como la salida de datos a un fichero, *5-7-mas.dat*.

Para calcular la energía lo hacemos sumando cinética y potencial:

$$E = -\frac{GMm}{r} + \frac{1}{2}mv^2 \tag{5.6}$$

El cumplimiento de la segunda ley de Kepler está íntimamente conectado con la conservación del momento angular. Como las fuerzas presentes son conservativas, y dirigidas en la dirección de el vector de posición, el momento angular se conserva. Si llamamos l a su módulo, se puede demostrar que la velocidad areolar cumple, si μ es la masa reducida del sistema, la siguiente relación:

$$\frac{\mathrm{d}A}{\mathrm{d}t} = \frac{l}{2\mu} \tag{5.7}$$

Por otro lado, como el movimiento transcurre en un plano, el momento angular solo tiene componente en z, siendo por tanto $l = r_x p_y - r_y p_x = m(r_x v_y - r_y v_x)$. Por tanto,

[1] Podemos hacer esto ya que, para una posición dada, la velocidad inicial determina univocamente la energía, con esta el semieje mayor de la órbita y, con este, a su vez, se determina el periodo de la órbita. De hecho, podríamos hacer los cálculos analíticamente, pero entonces no tendría mucho sentido el cálculo numérico

la conservación del momento angular y el cumplimiento de la segunda ley de Kepler equivalen a que se conserve la cantidad contenida en el último paréntesis. Esta cantidad es la que se calcula en la línea (109).

Podemos ver en las gráficas 5.2 (página 73), 5.3 (página 74), 5.4 (página 75), que tanto la energía como el momento angular se conservan a lo largo de la trayectoria, incluyendo bastantes órbitas, y que, efectivamente, las sucesivas órbitas se superponen.

El listado del código FORTRAN se encuentra a continuación:

```
1  *     Neptuno
2  *     masa = 102.43e24 (kg)
3  *     periodo = 60190 dias
4  *     semieje mayor = 4495.06e6 (km)
5  *     perihelio = 4444.45e6 (km)
6  *     excentricidad = 0.0113
7  *     Trabajo en unidades SI
8        implicit real*8 (a-h,o-z)
9        parameter (n=4) !funciones
10       dimension y(n),dydx(n),yout(n)
11       external derivs
12       real*8 Msol,mnep,l
13       common /ene/nn
14       common /datos/x0,y0,vx0,vy0,Msol,mnep,G
15       nn=n
16       G=6.6726d-11
17       Msol = 1.9889d30
18       mnep = 102.43d24
19       period = 60190d0*24.0d0*3600.0d0
20       sem = 4495.060d9
21       perihe = 4444.45d9
22       hprueb=1000.0d0
23       vuel = 10.0d0
24       ttotal = vuel*period
25       npasos = int(ttotal/hprueb)
26
27       x0 = perihe
28       y0 = 0.0d0
29       vx0 = 0.0d0
30       vescape = dsqrt(2.0d0*G*Msol/x0)
31       vr = 0.8d0*vescape
32       vl= 0.2d0*vescape
33       eps = 0.0113d0
34       alpha = dsqrt((1.0d0+eps)/2.0d0)
35       iter = 20
36       difmej=period*period
37
```

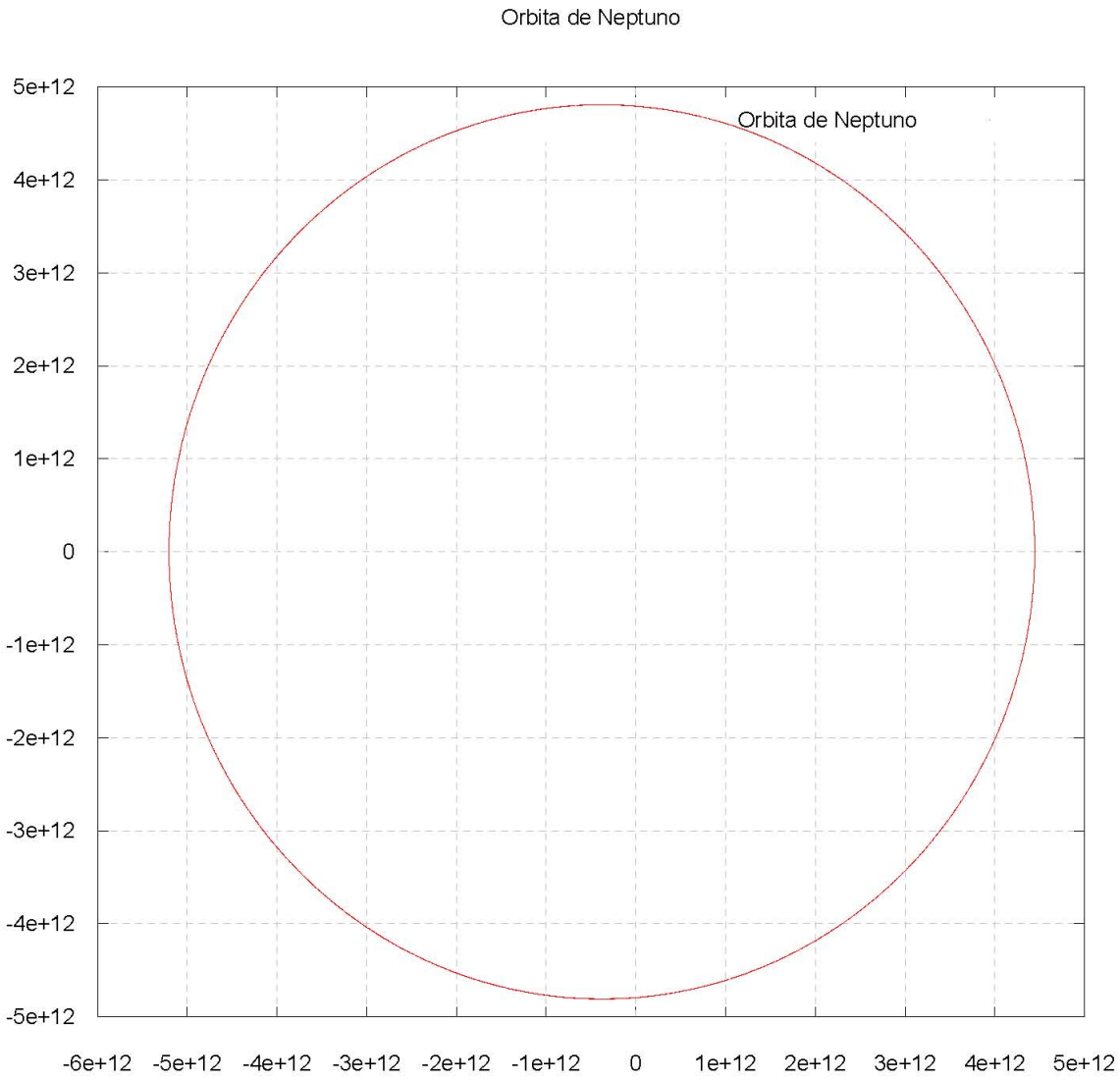

Figura 5.2: Representación de la órbita de Neptuno. Es una órbita muy excéntrica.

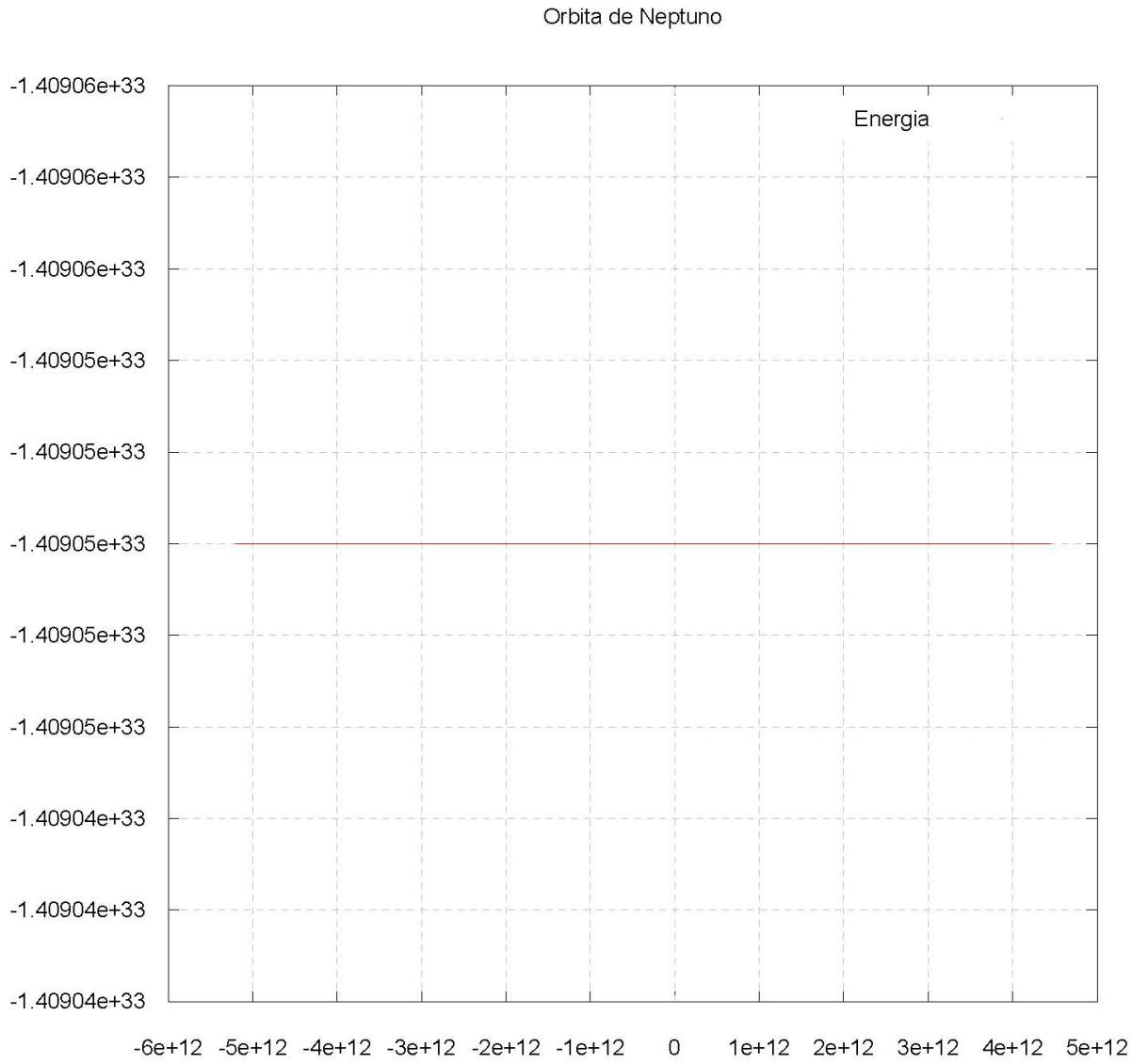

Figura 5.3: Representación de la energía de la órbita de Neptuno.

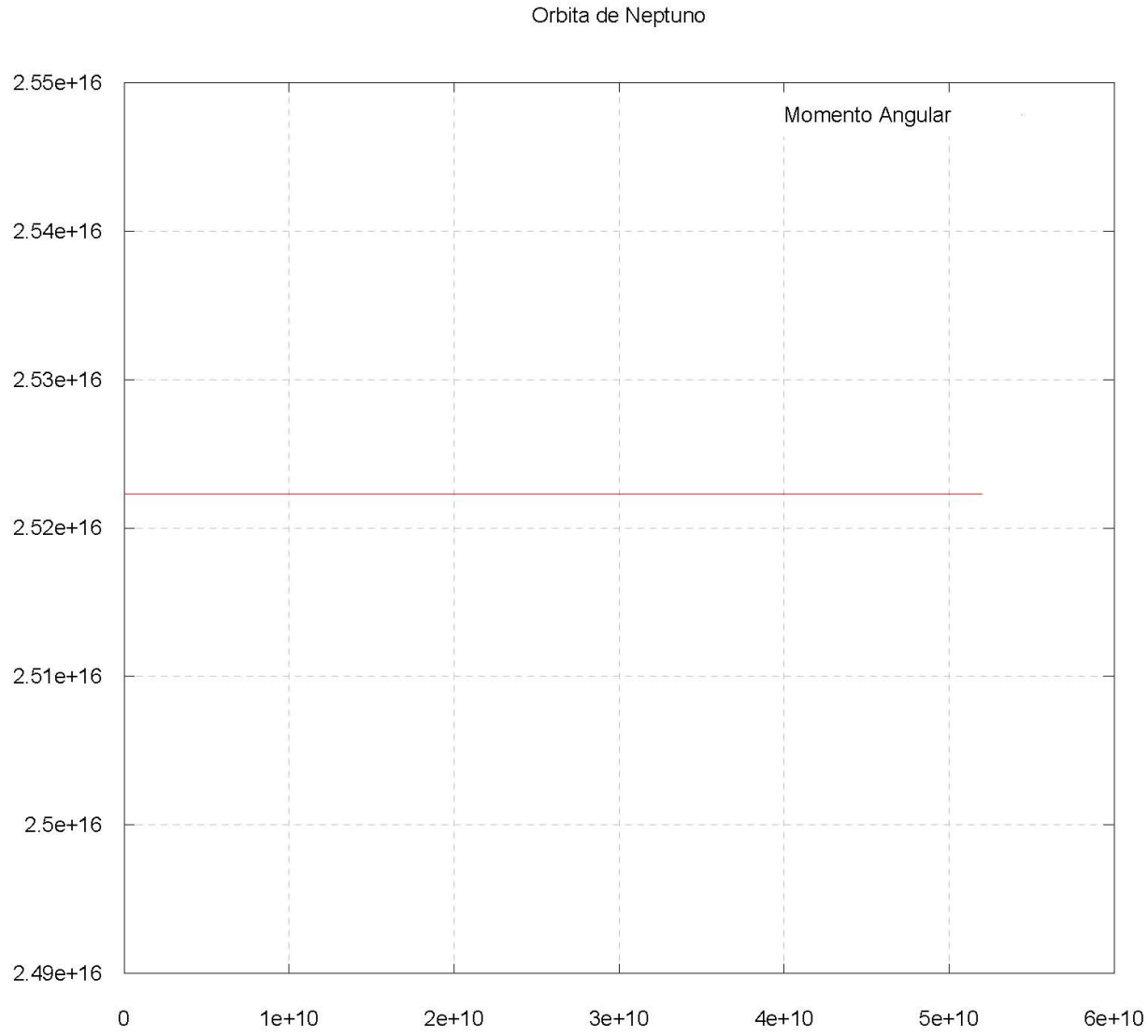

Figura 5.4: Representación del momento angular de la órbita de Neptuno.

```
38  C     Asignacion de las variables x,y,vx,vy a los elementos del array y(n)
39  C         y(1) == x     y(2) == y      y(3) == vx       y(4) == vy
40  C     Valores iniciales
41
42
43        do k=1,iter
44        vy0 = (vl + vr)/2.0d0
45
46        x=0.0d0
47        y(1) = x0
48        y(2) = y0
49        y(3) = vx0
50        y(4) = vy0
51
52        do while (y(1).gt.0.0d0)
53
54  *     Subrutina "derivs" (ecuaciones diferenciales)
55           call derivs(x,y,dydx)
56  *     Subrutina "rk4" (las soluciona)
57           call rk4(y,dydx,n,x,hprueb,yout,derivs)
58
59           x=x+hprueb
60           do j=1,n
61              y(j)=yout(j)
62           end do
63
64        end do ! do...end do de los calculos para una vy0 dada
65
66        tau = 4.0d0*x
67        if (tau.lt.period) then
68            vl = vy0
69            continue
70          else
71            vr = vy0
72            continue
73        end if
74
75        dif = abs(tau-period)
76        if (dif.lt.difmej) then
77           difmej = dif
78           vy0mej = vy0
79           else
80           continue
81        end if
82        write(*,*) difmej,tau,period
83        end do
```

```
84        tao = tau/(24.0d0*3600.0d0)
85        write(*,*)'La mejor velocidad inicial encontrada es vy0(m/s)=',vy0
86        write(*,*)'El periodo con dicha velocidad inicial es T(s)   =',tau
87        write(*,*)'El periodo con dicha velocidad inicial es T(dias)=',tao
88
89
90
91        x=0.0d0
92        y(1) = x0
93        y(2) = y0
94        y(3) = vx0
95        y(4) = vy0mej
96        h=10000000.0d0
97        ttotal = vuel*period
98        npasos = int(ttotal/h)
99
100
101
102       open(1,file='5-7-mas.dat',status='unknown')
103
104
105       do i=1,npasos
106       r = dsqrt(y(1)**2 + y(2)**2)
107       v = dsqrt(y(3)**2 + y(4)**2)
108       E = -(G*Msol*mnep)/r + 0.5d0*mnep*v**2
109       l = y(1)*y(4) - y(2)*y(3)
110       write(1,1000) x,y(1),y(2),y(3),y(4),E,l
111
112 *     Subrutina "derivs" (ecuaciones diferenciales)
113          call derivs(x,y,dydx)
114 *     Subrutina "rk4" (las soluciona)
115          call rk4(y,dydx,n,x,h,yout,derivs)
116
117          x=x+h
118          do j=1,n
119             y(j)=yout(j)
120          end do
121
122       end do ! do...end do de los calculos para una vy0 dada
123
124
125       close(1)
126       write(*,*)'                                        THE END'
127    1000 format(6(3x,g12.6),3x,g15.9)
128
129       end
```

```
130
131 ************************************************************************
132       subroutine derivs(x,y,dy)
133 ************************************************************************
134       implicit real*8 (a-h,o-z)
135       dimension y(nn),dy(nn)
136       real*8 Msol,mnep
137       common /ene/nn
138       common /datos/x0,y0,vx0,vy0,Msol,mnep,G
139       r = dsqrt(y(1)*y(1) + y(2)*y(2))
140       a = G*Msol/(r**2)
141       dy(1)=y(3)
142       dy(2)=y(4)
143       dy(3)=-a*(y(1)/r)
144       dy(4)=-a*(y(2)/r)
145       return
146       end
147 ************************************************************************
148       subroutine rk4(y,dydx,n,x,h,yout,derivs)
149 *     Numerical Recipes (2a ed.): p. 706
150 ************************************************************************
151       implicit real*8 (a-h,o-z)
152       integer n,nmax
153       real*8 h,x,dydx(n),y(n),yout(n)
154       external derivs
155       parameter (nmax=50)
156       integer i
157       real*8 h6,hh,xh,dym(nmax),dyt(nmax),yt(nmax)
158       hh=h*0.5d0
159       h6=h/6.0d0
160       xh=x+hh
161       do 11 i=1,n
162         yt(i)=y(i)+hh*dydx(i)
163 11    continue
164       call derivs(xh,yt,dyt)
165       do 12 i=1,n
166         yt(i)=y(i)+hh*dyt(i)
167 12    continue
168       call derivs(xh,yt,dym)
169       do 13 i=1,n
170         yt(i)=y(i)+h*dym(i)
171         dym(i)=dyt(i)+dym(i)
172 13    continue
173       call derivs(x+h,yt,dyt)
174       do 14 i=1,n
175         yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.0d0*dym(i))
```

```
176  14    continue
177        return
178        end
179
```

5.8. Ecuación de Poisson

Queremos resolver la ecuación de Poisson para un potencial radial, $\Phi(r)$, para una densidad de carga electrónica dada:

$$\frac{1}{r^2}\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}\Phi}{\mathrm{d}r}\right) = -4\pi\rho \tag{5.8}$$

$$\rho(r) = \frac{e^{-r}}{8\pi} \tag{5.9}$$

Para ello elegimos un cambio de variable de la forma: $r\phi(r) = \Phi(r)$. Para este cambio de variables, es inmediato ver que se ha de cumplir:

$$\frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} = -\frac{re^{-r}}{2} \tag{5.10}$$

Lo difícil de este problema es elegir correctamente las condiciones de contorno. Para ello, vamos a buscar que condiciones han de cumplir Φ y ϕ en $r = 0$ y/o $r = \infty$.

$r = 0$ Es lógico pedir al potencial $\Phi(r)$ que no sea divergente, es decir, que permanezca acotado incluso al acercarnos al origen. Por tanto, podemos pensar que, al estar Φ acotado en el origen, y ser $\phi(r) = r\Phi(r)$, se tiene que $u(0) = 0$.

$r = \infty$ Por otro lado, si pensamos que la distribución tiene simetría esférica, y que decrece exponencialmente, podemos plantearnos la posibilidad de que, a grandes distancias, la distribución pueda ser vista como una esfera donde se conserve toda la carga. Entonces el potencial debería ir como:

$$\Phi(r) \approx \frac{Q}{r} \tag{5.11}$$

donde Q sería la carga total. Pero si integramos la densidad ρ a todo el volumen, obtenemos:

$$\int_{\mathbb{R}^3}\rho(r)\mathrm{d}V = 4\pi\int_0^\infty r^2\frac{e^{-r}}{8\pi}\mathrm{d}r = 1 \tag{5.12}$$

Pero entonces, cuando $r \to \infty$, se tiene que $\phi(r \to \infty) = r\frac{1}{r} = 1$

Por tanto, las condiciones de contorno que debemos imponer son $u(0) = 0$ y $u(\infty) = 1$. Obteniendo las soluciones analíticamente de $\Phi(r)$ y $\phi(r)$, podemos comprobar que estas son las condiciones adecuadas para que el potencial tenga un buen sentido físico.

La resolución del problema la hacemos entonces para la función $\phi(r)$, y escribiremos la solución como $\Phi(r) = \phi(r)/r$, con las subrutinas `rk4` y `derivs`. La definición de estas son evidentes a partir de las ecuaciones diferenciales anteriores. Usamos el método del disparo como habitualmente lo hacemos, buscando un valor inicial de la derivada, hasta que se cumplan las condiciones de contorno, mediante el método de la bisección. Estas se imponen en las líneas (6-13). Después, con una derivada inicial óptima, reiniciamos los cálculos para obtener la solución correcta, y los datos se imprimen en un fichero, *5-8-mas.dat*. La solución se encuentra en la gráfica 5.5.

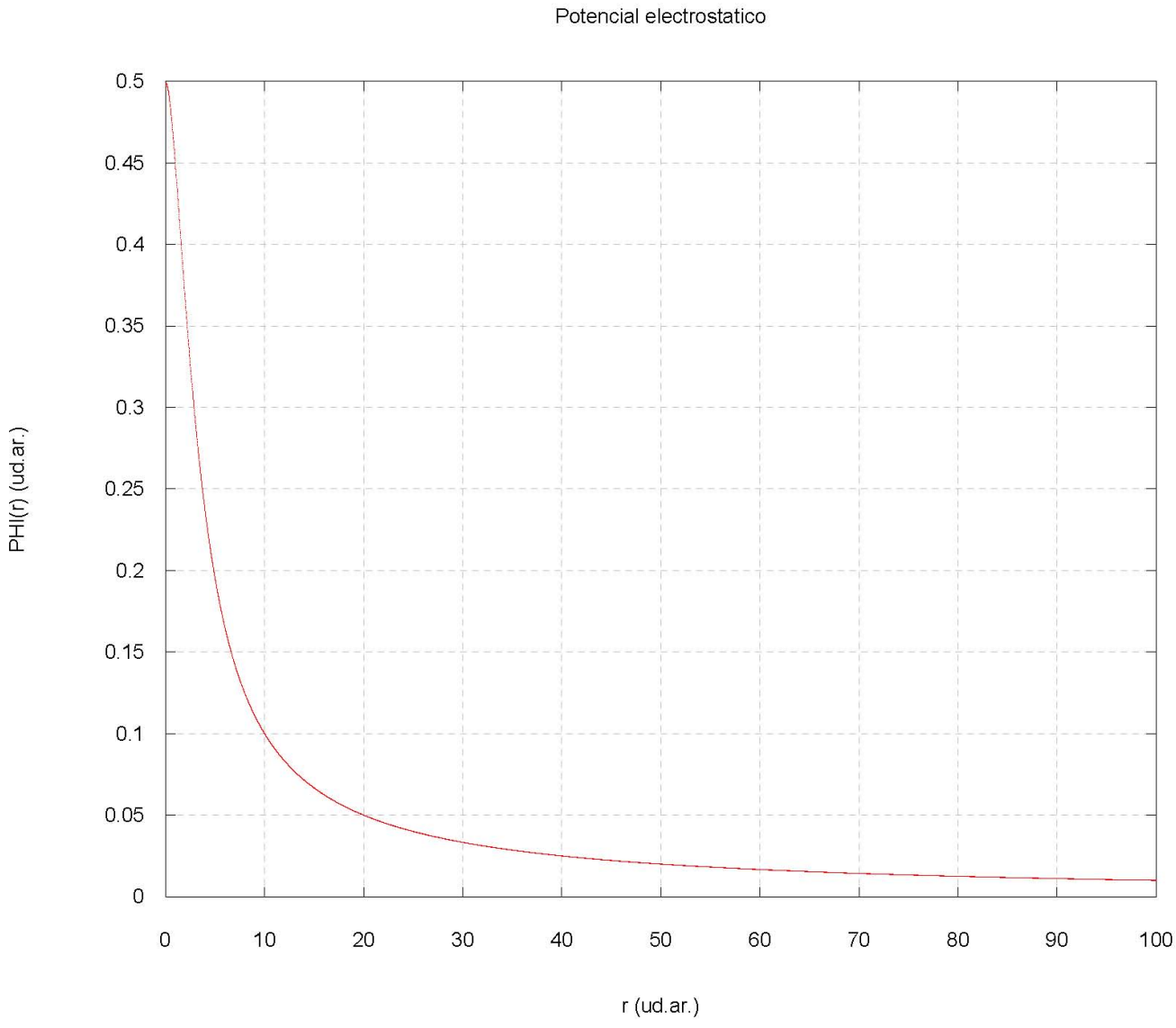

Figura 5.5: Representación del potencial para la densidad de carga dada.

El listado del código FORTRAN se encuentra a continuación:

```
1       parameter (n=2)
```

```
2         parameter (mpasos=100)
3         dimension y(n),dydx(n),yout(n),error(mpasos),yd(mpasos)
4         external derivs
5         common /ene/nn
6         nn=n
7         npasos = 10000
8         xi=0.0
9         xf=100.0
10        x=xi + tol  !Inicio de la variable r
11        step = (xf-xi)/real(npasos+1)
12        y1ini = 0.0
13        y1fin = 1.0
14
15        y(1) = y1ini
16        yd(1) = 10.0
17        yd(2) = -10.0
18        tol = 1.0e-4
19
20        open(1,file='5-8-mas.dat',status='unknown')
21        write(1,*)'#     r(m)      $\phi(r)$         $\Phi(r)$'
22
23        do m=1,mpasos
24
25        y(1) = y1ini
26        x=xi
27
28        if (m.ge.3) then
29         divis = error(m-1) - error(m-2)
30         yd(m) = yd(m-1) - error(m-1) * ( yd(m-1) - yd(m-2) ) / divis
31         y(2) = yd(m)
32         else
33         y(2) = yd(m)
34        end if
35
36        y2old = y(2)
37
38        do i=1,npasos
39           call derivs(x,y,dydx)
40           call rk4(y,dydx,n,x,step,yout,derivs)
41           x=x+step
42           y(1)=yout(1)
43           y(2)=yout(2)
44        end do
45
46        write(*,*)x,y(1),y2old
47        error(m) = y(1) - y1fin
```

```
48          if (abs(error(m)).lt.tol) then
49           goto 50
50           else
51           continue
52          end if
53         end do
54
55    50   do i=1,mpasos
56         write(*,*)yd(i),error(i)
57         end do
58
59         write(*,*)'Valor de la pendiente inicial adecuada : y(2) = ',y2old
60         y(1) = y1ini
61         y(2) = y2old
62         x = xi + tol
63
64         do while (x.ge.xi.and.x.le.xf)
65            write(1,100) x,y(1),y(1)/x
66            call derivs(x,y,dydx)
67            call rk4(y,dydx,n,x,step,yout,derivs)
68            x=x+step
69            y(1)=yout(1)
70            y(2)=yout(2)
71         end do
72
73
74
75         close(1)
76         do i=1,10
77           write(*,*) ' '
78         end do
79         write(*,*)'                                        THE END'
80    100  format(3x,3(2x,g15.9))
81         end
82
83
84    ************************************************************************
85         subroutine derivs(x,y,dy)
86    ************************************************************************
87         dimension y(nn),dy(nn)
88         common /ene/nn
89
90         dy(1)= y(2)
91         dy(2) = -x*Exp(-x)/2.0
92         return
93         end
```

```
94  ************************************************************************
95        subroutine rk4(y,dydx,n,x,h,yout,derivs)
96  ************************************************************************
97        integer n,nmax
98        real h,x,dydx(n),y(n),yout(n)
99        external derivs
100       parameter (nmax=50)
101       integer i
102       real h6,hh,xh,dym(nmax),dyt(nmax),yt(nmax)
103       hh=h*0.5
104       h6=h/6.
105       xh=x+hh
106       do 11 i=1,n
107         yt(i)=y(i)+hh*dydx(i)
108   11  continue
109       call derivs(xh,yt,dyt)
110       do 12 i=1,n
111         yt(i)=y(i)+hh*dyt(i)
112   12  continue
113       call derivs(xh,yt,dym)
114       do 13 i=1,n
115         yt(i)=y(i)+h*dym(i)
116         dym(i)=dyt(i)+dym(i)
117   13  continue
118       call derivs(x+h,yt,dyt)
119       do 14 i=1,n
120         yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.*dym(i))
121   14  continue
122       return
123       end
```

Tema 6

Ecuaciones diferenciales con derivadas parciales

6.6. Ecuación de Laplace

En este ejercicio se trata de resolver la ecuación de Laplace para un cable coaxial, de sección interior circular (radio $r = 0.1$ cm) y de sección exterior cuadrada (lado $l = 1.0$ cm) . La sección interior se encuentra a 100 V y la interior conectada a tierra (0 V). La ecuación es:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{6.1}$$

Nosotros usamos para resolverla en un mallado discreto la siguiente ecuación:

$$u_{i,j} = \frac{\dfrac{u_{i+1,j}}{f_1(f_1+f_3)} + \dfrac{u_{i,j+1}}{f_2(f_2+f_4)} + \dfrac{u_{i-1,j}}{f_3(f_1+f_3)} + \dfrac{u_{i,j-1}}{f_4(f_1+f_3)}}{\dfrac{1}{f_1 f_3} + \dfrac{1}{f_2 f_4}} \tag{6.2}$$

Los valores de f_i hacen referencia a si el punto se encuentra cerca de otro en el que el valor de la función se conoce exactamente o bien se encuentra cerca de una condición de contorno dada en un contorno irregular. Esto está detallado en los apuntes, no merece la pena explicarlo otra vez.

Como en realidad nosotros no conoceremos todos los puntos que se necesitan en la anterior ecuación, trabajaremos haciendo una serie de iteraciones en las que vayamos obteniendo nuevos valores con los que trabajar. Es decir, se parte de unos valores del potencial en el mallado más o menos caprichosos (salvo los que se corresponden a las condiciones de contorno del problema, que son fijas), y con esos valores obtenemos todos los valores nuevos del potencial en el mallado. Con estos valores volvemos a repetir el proceso, obteniendo unos nuevos valores, y así sucesivamente (modelo *Gauss-Seidel*). Se requerirán bastantes iteraciones, pero esto tampoco consume excesivo tiempo.

En las líneas (1-44) se definen una serie de valores que se necesitarán. Las iteraciones se inician en la línea (48). Por si fueran modificadas durante el proceso, las condiciones de contorno de la sección cuadrada se reinician otra vez en las líneas (50-54). El recorrido del mallado para cada iteración comienza en las líneas (61-62). En (63-66) se le asigna a las f_i un valor $f_i = 1.0$, y se usará ese valor si no se indica lo contrario. Se excluyen los cálculos en el interior de la sección circular mediante las líneas (67-68). El cálculo de las f_i, y de las condiciones que llevan a su definición y su uso se hace en las líneas (71-98). Ello se hace en base a sencillas consideraciones geométricas.

A partir de estos cálculos de los f_i, definimos los valores de los sumandos que aparecen en nuestra discretización de la ecuación, y calculamos el valor de $u_{i,j}$, en las líneas (100-104). Después se calcula el potencial en los puntos simétricos del cuadrado, y se cierra el ciclo, para una nueva iteración con los valores recién calculados. Finalmente, los datos se escriben en el archivo *6-6-mas.dat*. Estos datos pueden ser representados mediante GNUPLOT, con los archivos *6-6-mas-lateral.plt* –que ofrece una vista tridimensional del potencial en cada punto– y *6-6-mas-superior.plt* –para ver las curvas equipotenciales–; estos gráficos se muestran también en las figuras 6.2. La primera de las curvas equipotenciales tiene forma de quebrada por como se definen la sección interior circular.

El listado del código FORTRAN se encuentra a continuación:

```
1  C
2  C NOMBRE: 6-6-mas.f
3  C AUTOR: Miguel Albaladejo Serrano
4  C DESCRIPCION: Programa para calcular el potencial en el interior de un
5  !              cable coaxial de forma geometrica "especial"
6
7  *     Cable coaxial:
8  *        - Interior: Seccion circular, 0.1 cm de radio, 100 V
9  *        - Exterior: Seccion cuadrada, 1.0 cm de  lado    0 V
10
11       parameter (np=100)
12       dimension x(-np:np),y(-np:np), v(-np:np,-np:np)
13
14 ! El resultado es sensible a las modificaciones de las condiciones, ya
15 ! que se requieren muchas iteraciones para un calculo final "decente"
16       niter = 4000
17       b=1.0 !  lado de la seccion cuadrada (en cm)
18       r=0.1 ! radio de la seccion circular (en cm)
19       h = b/real(np) ! separacion entre los puntos
20       f1 = 1.0
21       f2 = 1.0
22       f3 = 1.0
23       f4 = 1.0
24       dr = sqrt(2.0)*h
```

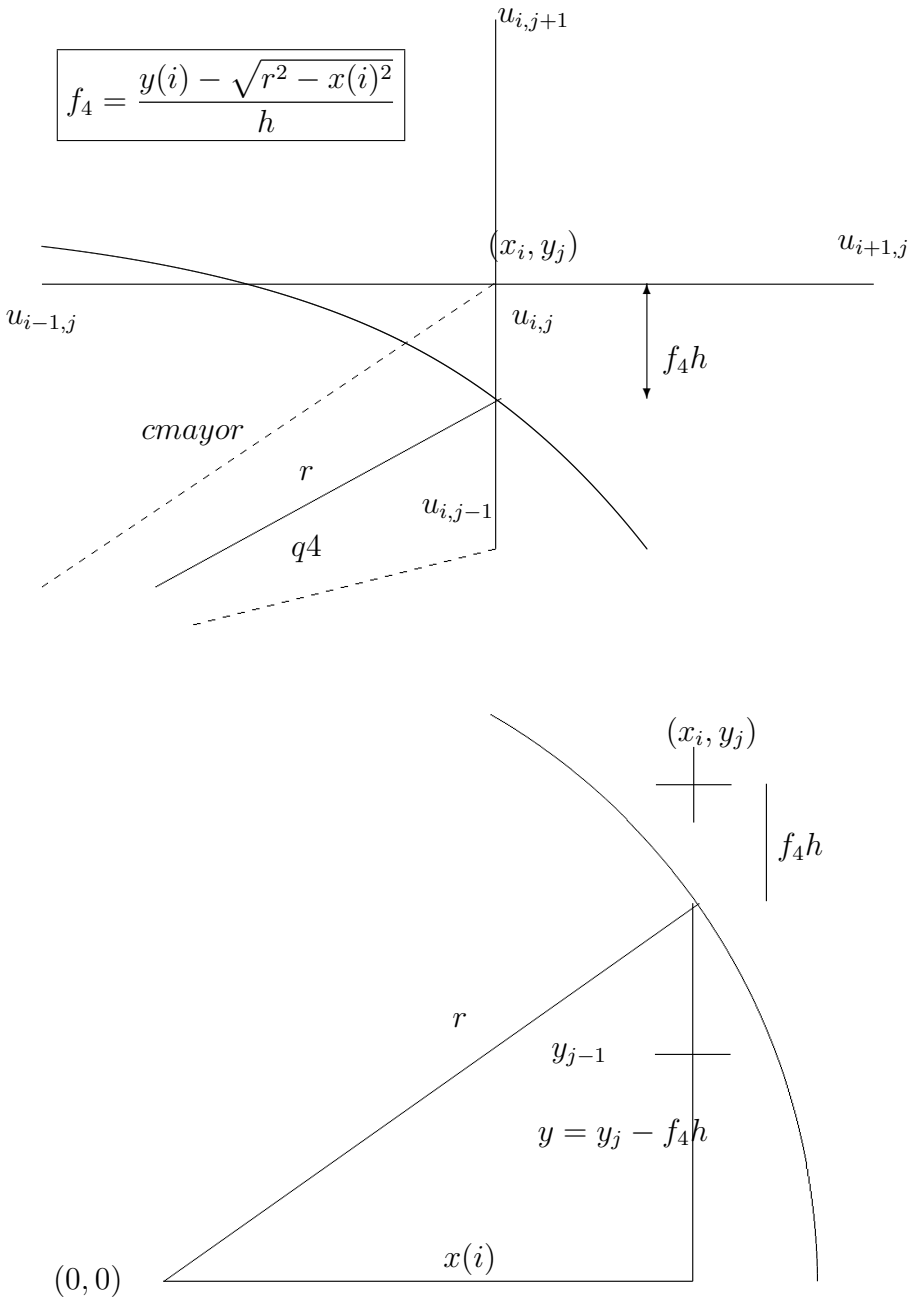

Figura 6.1: Definición de f_4. De forma análoga se define f_3 .

```
25        ! Inicio los valores de v(i,j) a cero
26        do i=-np,np
27        x(i) = real(i)*h
28        y(i) = real(i)*h
29        end do
```

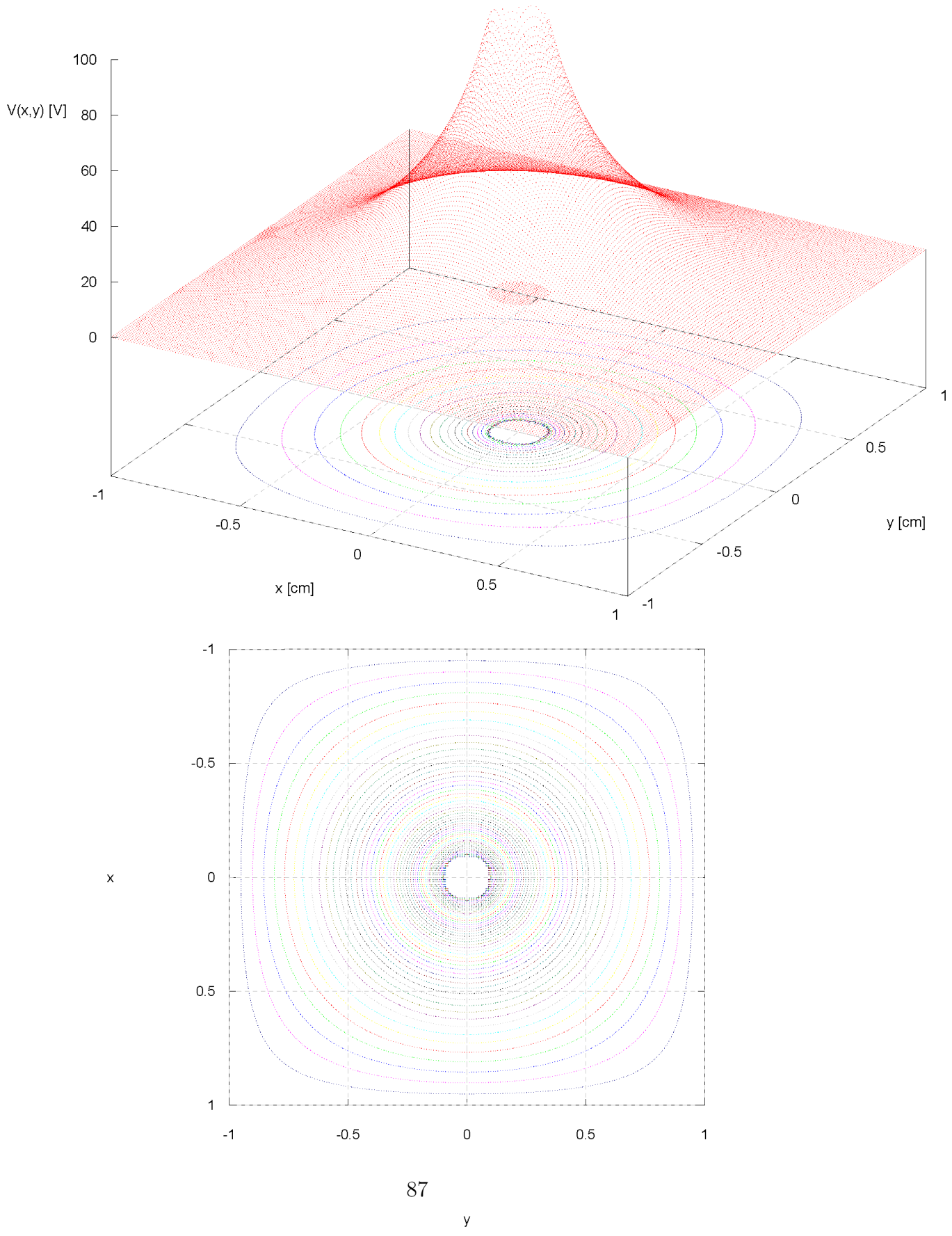

Figura 6.2: Representación del potencial y las curvas equipotenciales.

```
30
31  C Inicializo los valores de v a cero. Esta es una solucion doble: por un
32  C lado, tengo los valores inicializados. Por otro lado, como no voy a
33  C calcular valores en el interior de la seccion circular, me aseguro de
34  C que en el interior haya algun valor en el interior para poder
35  C representar, ya que para hacer curvas de nivel no queda mas remedio
36
37        do i=-np,np
38        do j=-np,np
39         !v(i,j) = 100.0*(1.0 - sqrt(x(i)**2 + y(j)**2))
40         v(i,j) = 0.0
41        enddo
42        enddo
43
44        open(1,file='6-6-mas.dat',status='unknown')
45
46        ! Estas son las iteraciones por las que se van resolviendo los
47        ! v(i,j), a partir de valores que se van corrigiendo
48        do l=1,niter
49
50        do i=-np,np
51        v(-np,i) = 0.0
52        v(np,i)  = 0.0
53        v(i,-np) = 0.0
54        v(i,np)  = 0.0
55        !do j=-np,np
56        !  cmenor = sqrt(x(i)**2 + y(j)**2)
57        !  if ((cmenor+0.5*dr).gt.r.and.(cmenor-0.5*dr).lt.r)v(i,j) = 100.0
58        !enddo
59        enddo
60
61        do i=0,np-1
62        do j=0,np-1
63        f1 = 1.0
64        f2 = 1.0
65        f3 = 1.0
66        f4 = 1.0
67        cmayor = sqrt(x(i)**2 + y(j)**2)
68        if (cmayor.lt.r) goto 100 !Si esta en el interior del circulo,
69                                  !debe saltar a una nueva coordenada
70
71        q3 = sqrt(x(i-1)**2 + y(j)**2)
72        q4 = sqrt(x(i)**2 + y(j-1)**2)
73
74  C Aqui el ordenador comprueba si esta cerca de alguna condicion de
75  C contorno dada por la seccion circular, y calcula las f que necesita.
```

```
C Como es imposible que tenga condiciones en el circulo a la derecha y
C arriba de los puntos, no considero f1 y f2. La seccion circular se
C quedara siempre abajo y/o a la izquierda del punto en el que trabaje.

       if (q3.lt.r) then
         f3 = abs(x(i) - sqrt(r**2 - y(j)**2))/h
         t3 = 100.0/(f3*(f1+f3))
         continue
       else
         f3 = 1.0
         t3 = v(i-1,j)/(f3*(f1+f3))
         continue
       end if

      if (q4.lt.r) then
         f4 = abs( y(j) - sqrt(r**2 - x(i)**2) )/h
         t4 = 100.0/(f4*(f2+f4))
         continue
       else
         f4 = 1.0
         t4 = v(i,j-1)/(f4*(f2+f4))
         continue
       end if

      t1 = v(i+1,j)/(f1*(f1+f3))
      t2 = v(i,j+1)/(f2*(f2+f4))
      d1 = 1.0/(f1*f3)
      d2 = 1.0/(f2*f4)
      v(i,j) = (t1 + t2 + t3 + t4)/(d1 + d2)
C     Por simetria, se ha de tener:
      v(-i,-j) = v(i,j)
      v(i,-j) = v(i,j)
      v(-i,j) = v(i,j)
 100  end do !do de las i (variable x)
      end do !do de las j (variable y)
      end do !do de las l (iteraciones Gauss-Seidel)

C     Aqui escribimos los resultados... por fin!!!!
      do m=-np,np
      do k=-np,np
      write(1,1000) x(m),y(k),v(m,k)
 200  enddo
      write(1,*)
      enddo
      close(1)

```

```
122   1000 format(2x,3(f15.7,3x))
123
124        end
```

Tema 7

Método de Montecarlo

7.5. Camino aleatorio

Queremos calcular un *camino aleatorio*, es decir, imaginamos uan persona caminando en direcciones aleatorias dando pasos de una longitud dada por distribuciones aleatorias, pero de un tipo dado: gaussianas y uniformes. Para ello debemos obtener dos datos al azar, que son la *longitud del paso* y su *dirección*. La longitud del paso la obtenemos mediante una de las dos fórmulas siguientes, suponiendo que z_1, z_2, z_3 y z_4 son dos números aleatorios entre 0 y 1:

- Distribución gaussiana:

$$r = x_{\text{media}} + \sigma\sqrt{-2\log z_1}\cos(2\pi z_2) \tag{7.1}$$

- Distribución uniforme:

$$r = x_{\text{media}} + (x_{\text{máx}} - x_{\text{mín}})z_3 \tag{7.2}$$

Para hallar la dirección del paso usaremos un ángulo, que podemos obtener como:

$$\theta = 2\pi z_4 \tag{7.3}$$

Por otro lado, una vez obtenida la longitud del paso y el ángulo que marca la dirección, determinaremos la coordenada en la que el caminante se encuentra a través de las fórmulas:

$$x_i = x_{i-1} + r\cos\theta \tag{7.4a}$$
$$y_i = y_{i-1} + r\sin\theta \tag{7.4b}$$

Nosotros hacemos un número dado de simulaciones, dado por el parámetro `nsim` (por defecto, `nsim` = 100) , de un "paseo" aleatorio de 100000 (cien mil) pasos, definidos mediante el parámetro `nmax`. Después hacemos simultáneamente el paseo para la distribución gaussiana y para la distribución uniforme. En cada simulación, calculamos para

cada valor del número de pasos la distancia al cuadrado recorrida, y vamos sumándola a un contador correspondiente a ese número de pasos. Finalmente, cuando se han hecho todas las simulaciones, calculamos la media, y hallamos su raíz... Podemos hacerlo también al revés, hallar la raíz y después calcular la media, pero esto solo influirá en la pendiente de la teórica recta que debemos obtener. El criterio que seguimos es el siguiente: hallamos la media de la distancia cuadrática, y de ella, hallamos la raíz, obteniendo así una estimación de la distancia a la que estará en cada paso. Todos estos datos (para los parámetros seleccionados, se obtiene un total de 23 MB de datos) se almacenan en los siguientes ficheros:

- Distribución gaussiana
  - Dibujo del paseo: *7-5-mas-G-camino.dat*
  - Ajuste y datos de la distancia cuadrática media: *7-5-mas-gauss.dat*, *datos-ajuste-gauss.dat*
-
  - Dibujo del paseo: *7-5-mas-U-camino.dat*
  - Ajuste y datos de la distancia cuadrática media: *7-5-mas-unif.dat*, *datos-ajuste-unif.dat*

Una vez que se ejecuta el programa *7-5-mas.f*, se generan los ficheros de datos 7-5-mas-gauss.dat y 7-5-mas-unif.dat, pero después hay que ejecutar los archivos *0-12-mas-gauss.f* y *0-12-mas-unif.f* para ajustar los datos de las distribuciones gaussiana y uniforme, obteniendo, respectivamente, *datos-ajuste-gauss.dat* y *datos-ajuste-gauss.dat*. El valor del número de datos con el que hay que realizar el ajuste es, evidentemente, el número de pasos que se da en la simulación, que viene dado por el parámetro `nmax`, que por defecto vale `nmax` = 100000[1]. En resumen, hay que ejecutar los siguientes programas:

- *7-5-mas.f*
- *0-12-mas-gauss.f*
- *0-12-mas-unif.f*

Una vez ejecutados, puede abrir con GNUPLOT los archivos que contienen los datos y la recta del ajuste, *7-5-mas-gauss.plt* y *7-5-mas-unif.plt*. En las gráficas 7.1 y 7.2, basadas en estos archivos, se puede observar como estos datos se ajustan perfectamente a una recta, para ambos casos.

Las rectas que obtenemos son:

$$d_{\mathrm{CM}}^{\mathrm{gausiana}} = (50.53 \pm 0.01)\sqrt{N} + (297.97 \pm 2.47) \tag{7.5}$$

$$d_{\mathrm{CM}}^{\mathrm{uniforme}} = (59.85 \pm 0.01)\sqrt{N} + (-101.69 \pm 2.54) \tag{7.6}$$

[1]Es curioso ver como estos programas (basados en el ejercicio 0.12 que ya hicimos) son capaces de ajustar los cien mil datos en unos instantes... No me imagino haciéndolo a mano.

La pendiente parece estar claramente relacionada con la media de la distribución. Por otro lado, no debe asombrarnos la ordenada en el origen, que es del orden de 100 mientras que se llegan a recorrer "distancias" de 20000, es decir, la ordenada en el origen es del orden del 1 %, algo despreciable.

El listado del código FORTRAN se encuentra a continuación:

```
!     xg=xmean+sigma*sqrt(-2.0*log(z1))*cos(2.0*pi*z2)
      parameter (nmax=100000)
      dimension dg2(1:nmax),ddg2(1:nmax),du2(1:nmax),ddu2(1:nmax)
      pi=4.0*atan(1.0)
      iseed1 = -564213568
      iseed2 = -231876339
      iseed3 = -548236787

      x1=0.0
      y1=0.0

      jmax = 100
      jpinta = 15
      rgmean = 50.0
      sigma = 10.0

      rumean = 50.0
      rmin = 40.0
      rmax = 60.0

      nsim=100

      x=x1
      y=y1

      open(1,file='7-5-mas-gauss.dat',status='unknown')
      open(2,file='7-5-mas-G-camino.dat',status='unknown')
      open(3,file='7-5-mas-unif.dat',status='unknown')
      open(4,file='7-5-mas-U-camino.dat',status='unknown')



      do k=1,nmax !do del ciclo k=1,nn=j*1000 --> ciclo del paseo
      dg2(k)=0.0
      du2(k)=0.0
      enddo


      do j=1,nsim
      xg=x1
```

```
41        yg=y1
42        xu=x1
43        yu=y1
44
45        do k=1,nmax !do del ciclo k=1,nn=j*1000 --> ciclo del paseo
46        if (j.eq.nsim) write (2,200) xg,yg
47        if (j.eq.nsim) write (4,200) xu,yu
48
49        !write(2,200) x,y
50        z1 = ran1(iseed1)
51        z2 = ran1(iseed2)
52        z3 = 2.0*pi*ran1(iseed3)
53
54        rg = rgmean + sigma*sqrt(-2.0*log(z1))*cos(2.0*pi*z2)
55        ru = rmin + (rmax-rmin)*z2
56        xg = xg + rg*cos(z3)
57        yg = yg + rg*sin(z3)
58
59        xu = xu + ru*cos(2.0*pi*z1)
60        yu = yu + ru*sin(2.0*pi*z1)
61
62        dg2(k) = dg2(k) + (xg-x1)**2 + (yg-y1)**2
63        du2(k) = du2(k) + (xu-x1)**2 + (yu-y1)**2
64        enddo
65        enddo
66
67        do k=1,nmax !do del ciclo k=1,nn=j*1000 --> ciclo del paseo
68        ddg2(k)=sqrt(dg2(k)/real(nsim))
69        ddu2(k)=sqrt(du2(k)/real(nsim))
70
71        write(1,200) sqrt(real(k)),ddg2(k)
72        write(3,200) sqrt(real(k)),ddu2(k)
73        enddo
74
75
76
77        close(1)
78        close(2)
79        close(3)
80        close(4)
81
82  * 100  format(3x,i9,4x,f14.7)
83   200  format(3x,f20.9,4x,f20.9)
84        end
85  ************************************************************
86        FUNCTION ran1(idum)
```

```
87  * generador de numeros aleatorios del Numerical Recipes, 2 ed.
88  ****************************************************************
89        INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
90        REAL ran1,AM,EPS,RNMX
91        PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
92       *NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
93        INTEGER j,k,iv(NTAB),iy
94        SAVE iv,iy
95        DATA iv /NTAB*0/, iy /0/
96        if (idum.le.0.or.iy.eq.0) then
97          idum=max(-idum,1)
98          do 11 j=NTAB+8,1,-1
99            k=idum/IQ
100           idum=IA*(idum-k*IQ)-IR*k
101           if (idum.lt.0) idum=idum+IM
102           if (j.le.NTAB) iv(j)=idum
103 11      continue
104         iy=iv(1)
105       endif
106       k=idum/IQ
107       idum=IA*(idum-k*IQ)-IR*k
108       if (idum.lt.0) idum=idum+IM
109       j=1+iy/NDIV
110       iy=iv(j)
111       iv(j)=idum
112       ran1=min(AM*iy,RNMX)
113       return
114       END
115
```

7.6. Aguja de Buffon

El problema de la aguja de Buffon consiste en la aplicación de un método aleatorio para hacer una estimación del número π. Según está estimación, si dejamos caer agujas de longitud l en una baldosa cuadrada de lado d, la probabilidad de que aquellas corten con la losa viene dada por:

$$P = \frac{4l}{\pi d} \tag{7.7}$$

y despejando π:

$$\pi = \frac{4l}{Pd} \tag{7.8}$$

La probabilidad de que una aguja y alguno de los lados se corten podemos simularla tirando muchas agujas, y viendo cuantas de estas cortan a la losa respecto del número total de tiradas.

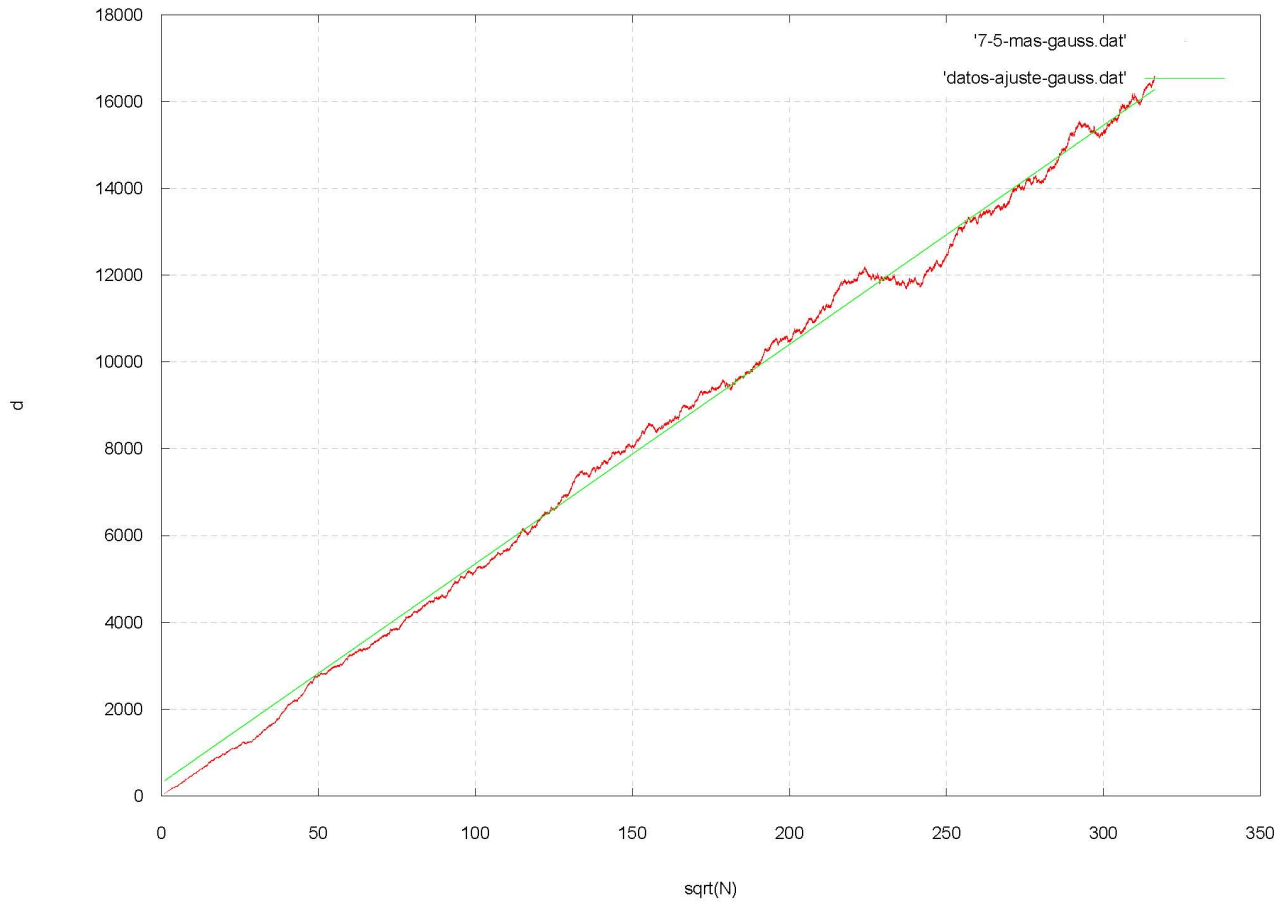

Figura 7.1: Datos y su ajuste para el paseo con distribución gaussiana

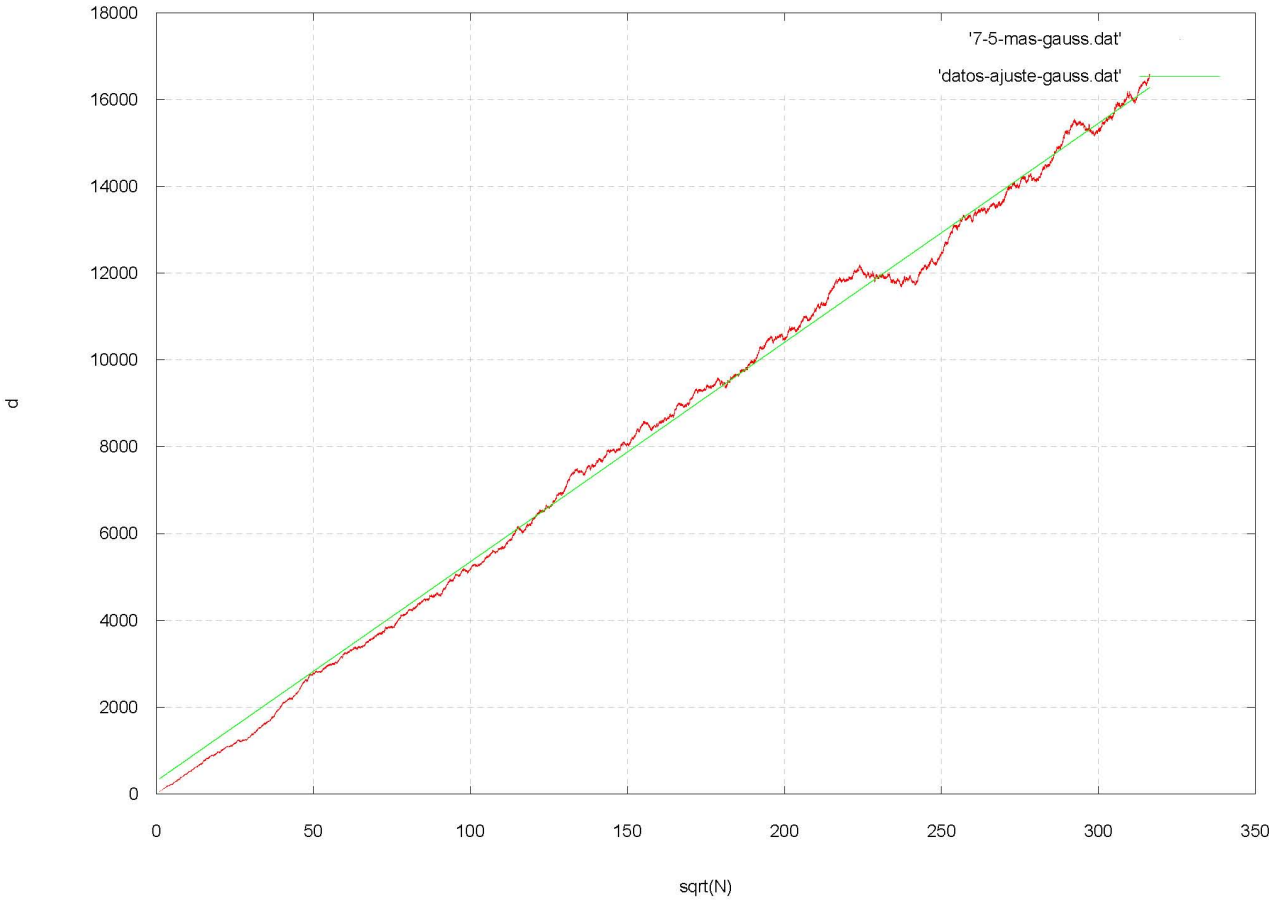

Figura 7.2: Datos y su ajuste para el paseo con distribución uniforme

En esto consiste nuestro programa. En él vamos haciendo varias simulaciones con un número mayor de agujas lanzadas, y calculando en cada uno de estos pasos el correspondiente valor de π. Estos datos se almacenan en el fichero *7-6-mas.dat*, y la representación de los sucesivos valores de π en función del número de tiradas viene dado en el archivo *7-6-mas.plt*, para GNUPLOT, y también se representan en la figura 7.3.

Los resultados numéricos que se obtienen, a partir de los cuales se obtiene la gráfica 7.3, son los siguientes:

```
#Num. de cortes  lanzamientos          probabilidad          pi
           17          512            0.0332031250        2.2588236332
           17          724            0.0234806631        3.1941175461
           24         1024            0.0234375000        3.2000000477
           31         1448            0.0214088392        3.5032258034
           54         2048            0.0263671875        2.8444445133
           57         2896            0.0196823198        3.8105263710
          118         4096            0.0288085938        2.6033897400
          139         5793            0.0239944756        3.1257195473
          212         8192            0.0258789063        2.8981132507
          272        11585            0.0234786365        3.1943933964
          375        16384            0.0228881836        3.2767999172
          553        23170            0.0238670688        3.1424050331
          749        32768            0.0228576660        3.2811748981
         1062        46341            0.0229170714        3.2726695538
         1537        65536            0.0234527588        3.1979179382
         2139        92682            0.0230789147        3.2497196198
         3160       131072            0.0241088867        3.1108860970
         4487       185364            0.0242064260        3.0983507633
         6361       262144            0.0242652893        3.0908348560
         8842       370727            0.0238504335        3.1445968151
        12618       524287            0.0240669716        3.1163039207
        17442       741454            0.0235240478        3.1882266998
        24969      1048575            0.0238123164        3.1496305466

 # Resultados teniendo en cuenta todas las iteraciones

 Num. de cortes  lanzamientos          probabilidad
        85235      3578822            0.0238164961

 Valor de PI =  3.14963055
 Error relativo = abs(pi-piexacto)/piexacto =   0.00255851285
Porcentaje de error =   0.26 %
```

El listado del código FORTRAN se encuentra a continuación:

```
      integer cortan,tirads
      real l
```

```
      d=40.0
      l=0.75
      ncort=0
      ntotal=0
      nmin=18
      nmax=40

C Aquą hay otras semillas con valores de nmax adecuados para los
C cuales se obtiene un buen valor de pi
      !iseed = -845126256  !--> nmax = 30
      iseed = -34593417  !--> nmax = 40
      !iseed = -546213546 !--> nmax = 46

      piex = 4.0*atan(1.0)


      open(1,file='7-6-mas.dat',status='unknown')
      write(1,*)'#Num. de cortes  lanzamientos          ',
     &'probabilidad          pi'
      do i=nmin,nmax
      n = nint(sqrt(2.0)**i)
      cortan = 0

      do j=1,n
      x1 = d*ran1(iseed)
      y1 = d*ran1(iseed)
      z = 2.0*piex*ran1(iseed)

      x2 = x1 + l*cos(z)
      y2 = y1 + l*sin(z)

      if (x2.gt.d.or.x2.lt.0.0.or.y2.gt.d.or.y2.lt.0.0) then
          cortan = cortan + 1
          continue
          else
          continue
      endif

      enddo

          prob = real(cortan)/real(n)
          pi = 4.0*l/(d*prob)
          write(1,100) cortan,n,prob,pi
          ncort = ncort + cortan
          ntotal = ntotal + n
      enddo
```

```
49
50        probb = real(ncort)/real(ntotal)
51        pi = 4.0*l/(d*prob)
52        dif = abs(pi-piex)/piex
53        !write(1,100) ncort,ntotal,probb,pi
54
55        write(1,*)' '
56        write(1,*)'# Resultados teniendo en cuenta todas las iteraciones'
57        write(1,*)' '
58        write(1,*)'Num. de cortes  lanzamientos          ',
59       &'probabilidad'
60        write(1,200) ncort,ntotal,probb
61        write(1,*)' '
62        write(1,*) 'Valor de PI =',pi
63        write(1,*) 'Error relativo = abs(pi-piexacto)/piexacto = ',dif
64        write(1,300)dif*100
65        close(1)
66   100  format(4x,2(i9,4x),4x,f16.10,4x,f16.10)!,4x,f16.10)
67   200  format(4x,2(i9,4x),4x,f16.10)!,4x,f16.10,4x,f16.10)
68   300  format( 'Porcentaje de error = ',f6.2,' %')
69        end
70  ************************************************************
71        FUNCTION ran1(idum)
72  * generador de numeros aleatorios del Numerical Recipes, 2 ed.
73  ************************************************************
74        INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
75        REAL ran1,AM,EPS,RNMX
76        PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
77       *NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
78        INTEGER j,k,iv(NTAB),iy
79        SAVE iv,iy
80        DATA iv /NTAB*0/, iy /0/
81        if (idum.le.0.or.iy.eq.0) then
82          idum=max(-idum,1)
83          do 11 j=NTAB+8,1,-1
84            k=idum/IQ
85            idum=IA*(idum-k*IQ)-IR*k
86            if (idum.lt.0) idum=idum+IM
87            if (j.le.NTAB) iv(j)=idum
88  11      continue
89          iy=iv(1)
90        endif
91        k=idum/IQ
92        idum=IA*(idum-k*IQ)-IR*k
93        if (idum.lt.0) idum=idum+IM
94        j=1+iy/NDIV
```

```
95          iy=iv(j)
96          iv(j)=idum
97          ran1=min(AM*iy,RNMX)
98          return
99          END
```

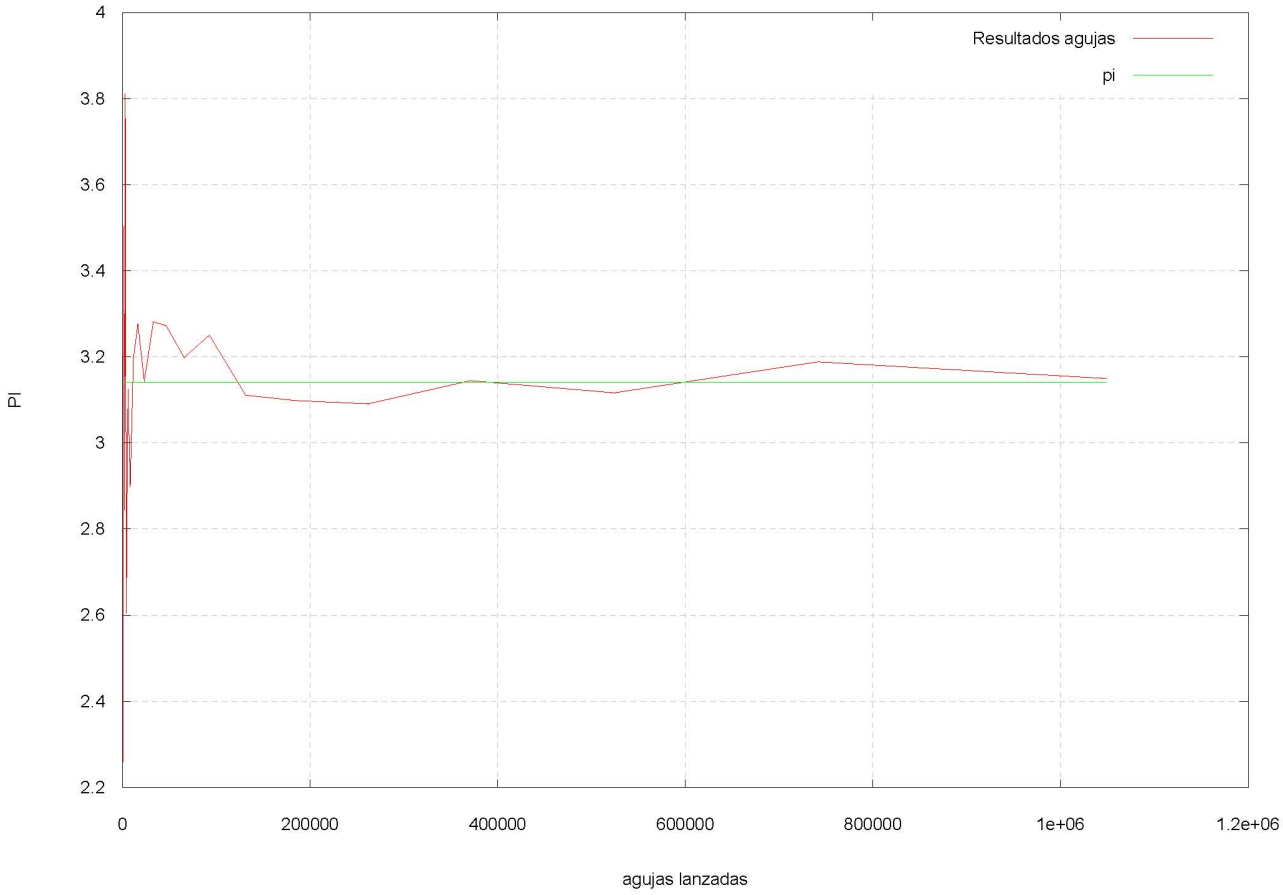

Figura 7.3: Sucesivos valores de π en función del número de agujas tiradas.

Tema 8

Transformada de Fourier

8.4. Transformada discreta de Fourier de un conjunto de datos

El programa consiste en hacer la transformada de Fourier de un conjunto de datos tipo (t_i, f_i). Esto se realiza mediante dos programas, de dos formas diversas. Una de ellas, con el programa *8-4-mas.f*, usando la subrutina `four1`, y la otra, con el programa *8-4ii-mas.f*, haciendo las operaciones directamente.

Como los datos son datos dados, no tenemos que preocuparnos de establecer tiempos totales o intervalos de tiempo, todo esto viene obligado ya por los datos. La única dificultad del programa es entonces leer los datos, lo que se realiza mediante un ciclo `do\ldots enddo` con un comando `read` en el interior de éste. Esto se ha de realizar en ambos programas. Las operaciones son ya aplicaciones directas de las fórmulas de la DFT (*Discrete Fourier Transform*) o de la FFT (*Fast Fourier Transform*) en el caso de la subrutina `four1`. La fórmula de la DFT es:

$$F(\nu_n) = \sum_{k=0}^{N-1} f_k \exp\left(2\pi i n k/N\right) \tag{8.1}$$

Los datos a "transformar" se representan en la gráfica 8.1, y su transformada, en la gráfica 8.2.

¿Qué deducimos de esta transformada de Fourier?. Suponemos que los datos iniciales vienen dados por (tiempo, una función del tiempo). Por ejemplo, en segundos. La frecuencia de Nyquist, al ser el intervalo temporal $\Delta t = 0.05$ s, viene dada por $\nu_c = 10.0$ Hz. Por lo tanto, no podemos esperar obtener informaciones fiables para frecuencias más allá de esta. Es más, esperamos obtener una imagen especular respecto a esta frecuencia. Por otro lado, los "picos" de $|F(\nu)|$ se alcanzarán en las frecuencias más importantes en la composición de la función original. Por tanto, podemos decir, como conclusiones obtenidas del espectro de frecuencias, que las más importantes que aparecen en la función original $f(t)$ son las frecuencias: $v = 1, 3, 4, 6, 8$, en las unidades que corresponda. Si los tiempos originalmente estaban en segundos, pues entonces las unidades serán hercios.

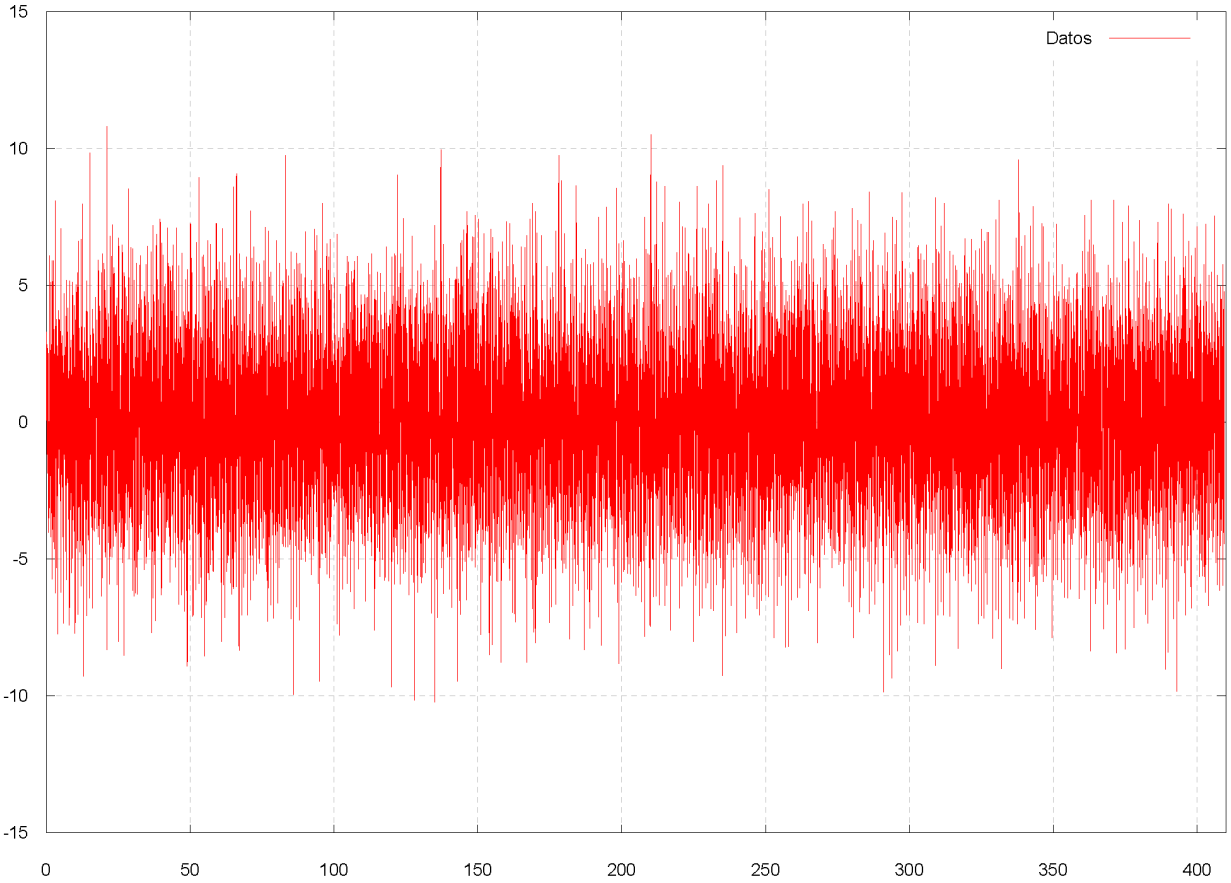

Figura 8.1: Representación de los datos iniciales.

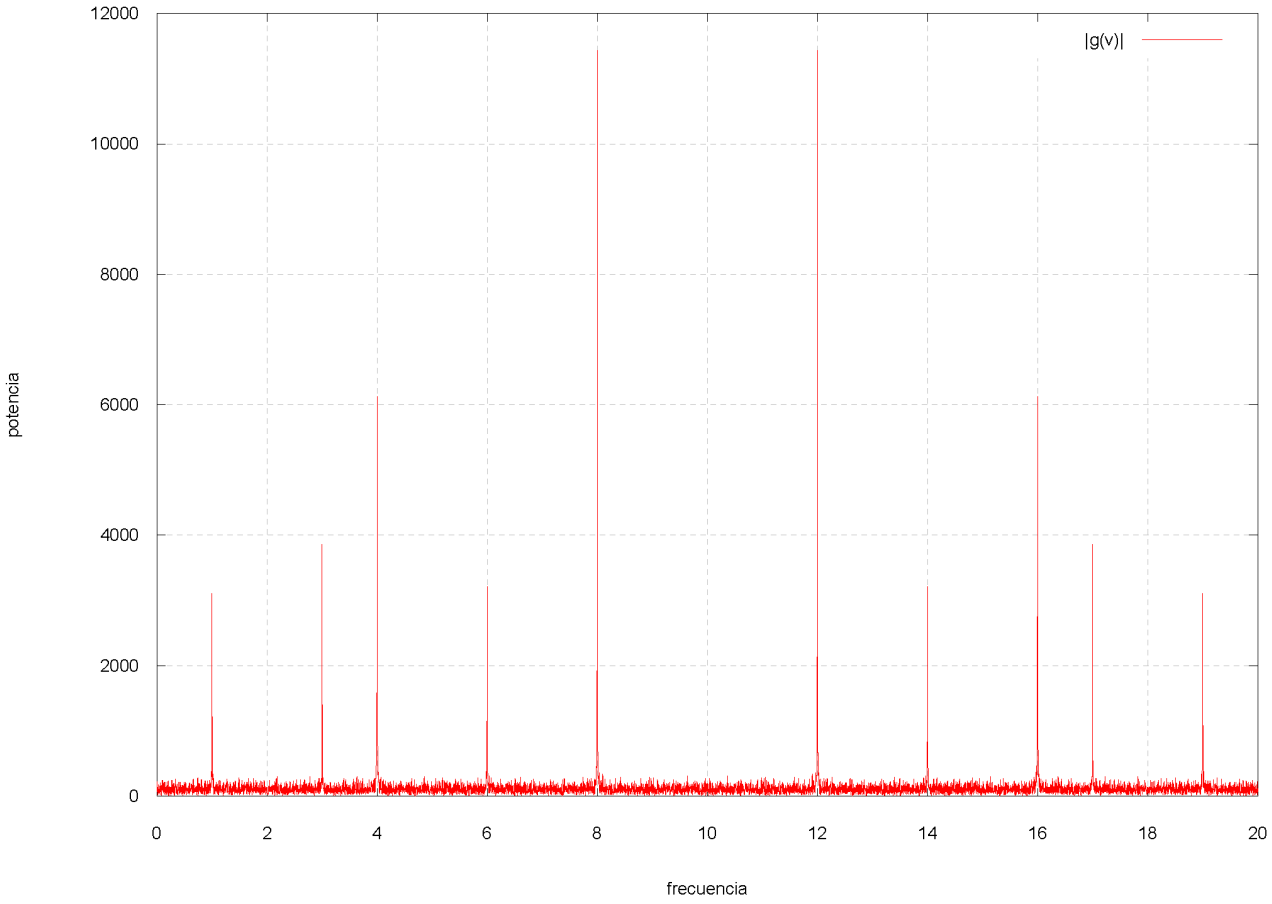

Figura 8.2: Representación de $|g(\nu)|$ en función de la frecuencia ν.

8.4.1. De porqué esas frecuencias, y no otras

Esas frecuencias que aparecen son las que aparecen en mi DNI, a saber: 48483618, porque "el DNI da para mucho". Por eso, además, se explica que la proporción en que aparece la frecuencia $\nu = 8$ sea la mayor, pues aparece 3 veces en mi DNI, después está el 4, que aparece dos veces, y después están el 1,3 y 6, que solo aparecen 1 vez. Qué curioso...

El listado del código FORTRAN usando la FFT, que constituye el programa *8-4-mas.f*, se encuentra a continuación:

```
******* Programa para calcular la transformada de Fourier rapida *******
      parameter (nn=8192)
      dimension tt(nn),y(nn),data(2*nn),gr(nn),gi(nn)
      dt=0.05
      dv=1.0/(real(nn)*dt)

      open(1,file='mas.dat',status='unknown')
      open(2,file='mas-poten.dat',status='unknown')

      do i=1,nn
      read(1,*) tt(i),y(i)
      enddo

      do i=1,2*nn,2
      ind = (i+1)/2
        data(i)=y(ind)
        data(i+1)=0.0
      end do

      isign=1
      call four1(data,nn,isign)

      write(*,*) 'Frecuencia de Nyquist : nu_s = ', 1.0/(2.0*dt)
      do i=1,2*nn,2
        ind = (i+1)/2
        gr(ind)=data(i)
        gi(ind)=data(i+1)
      enddo
      write(*,*) 'Frecuencia de Nyquist : nu_s = ', 1.0/(2.0*dt)

      do j=0,nn
        v=dv*real(j)
        gg=sqrt(gr(j)**2+gi(j)**2)
        write(2,100) v,gg
      enddo

      close(1)
```

```
38        close(2)
39
40        write(*,*) 'Frecuencia de Nyquist : nu_s = ', 1.0/(2.0*dt)
41      100 format (2x,2(g12.5,3x))
42        end
43
44
45  ************************************************************
46        SUBROUTINE four1(data,nn,isign)
47  * subrutina four1.for del Numerical Recipes
48  ************************************************************
49        INTEGER isign,nn
50        REAL data(2*nn)
51        INTEGER i,istep,j,m,mmax,n
52        REAL tempi,tempr
53        DOUBLE PRECISION theta,wi,wpi,wpr,wr,wtemp
54        n=2*nn
55        j=1
56        do 11 i=1,n,2
57          if(j.gt.i)then
58            tempr=data(j)
59            tempi=data(j+1)
60            data(j)=data(i)
61            data(j+1)=data(i+1)
62            data(i)=tempr
63            data(i+1)=tempi
64          endif
65          m=n/2
66  1       if ((m.ge.2).and.(j.gt.m)) then
67            j=j-m
68            m=m/2
69          goto 1
70          endif
71          j=j+m
72  11    continue
73        mmax=2
74  2     if (n.gt.mmax) then
75          istep=2*mmax
76          theta=6.28318530717959d0/(isign*mmax)
77          wpr=-2.d0*sin(0.5d0*theta)**2
78          wpi=sin(theta)
79          wr=1.d0
80          wi=0.d0
81          do 13 m=1,mmax,2
82            do 12 i=m,n,istep
83              j=i+mmax
```

```
84              tempr=sngl(wr)*data(j)-sngl(wi)*data(j+1)
85              tempi=sngl(wr)*data(j+1)+sngl(wi)*data(j)
86              data(j)=data(i)-tempr
87              data(j+1)=data(i+1)-tempi
88              data(i)=data(i)+tempr
89              data(i+1)=data(i+1)+tempi
90  12        continue
91            wtemp=wr
92            wr=wr*wpr-wi*wpi+wr
93            wi=wi*wpr+wtemp*wpi+wi
94  13      continue
95          mmax=istep
96        goto 2
97        endif
98        return
99        END
100
101
102
103
```

Para el caso del programa *8-4ii-mas.f*, en el que los calculos se realizan directamente, el listado de FORTRAN es:

```
1         parameter (nn=8192,nt=8192)
2         dimension t(0:nn-1),f(0:nn-1)
3
4         pi = 4.0*atan(1.0)
5
6         step = 0.05
7         stepfr = 1.0/(real(nn)*step)
8
9         open (2,file='mas.dat',status='unknown')
10        do i=0,nn-1
11        read (2,*) t(i),f(i)
12        enddo
13        open (1,file='8-4-mas-poten.dat',status='unknown')
14        v = 0.0
15        do i=0,nn-1
16        tt = t(i)
17        gr = 0.0
18        gi = 0.0
19           do j=0,nt-1
20             gr = gr + cos( 2.0*pi*real(i)*real(j)/real(nn) ) * f(j)
21             gi = gi + sin( 2.0*pi*real(i)*real(j)/real(nn) ) * f(j)
22             tt = tt + step
```

```
23            enddo
24          gg = sqrt(gr**2 + gi**2)
25          write (1,100) v,gg,gr,gi
26          v = v + stepfr
27          enddo
28
29    100   format(4x,4(f13.7,4x))
30          end
31
```

8.5. Mebrana vibratoria

El problema consiste en resolver la evolución temporal de una membrana cuadrada que puede vibrar, obligada a permanecer fija en su perímetro, con una posición inicial dada. Se trata pues de discretizar la ecuación:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \tag{8.2}$$

La discretización, usando fórmulas adecuadas, y suponiendo el mismo paso para el eje x que para el eje y, $h_x = h_y = h_s$, nos conduce a:

$$u_{i,j}^{k+1} = 2uu_{i,j}^k - u_{i,j}^{k-1} + \frac{h_t^2 c^2}{h_s^2}\left(u_{i-1,j}^k + u_{i+1,j}^k + u_{i,j-1}^k + u_{i,j+1}^k - 4u_{i,j}^k\right) \tag{8.3}$$

Si definimos (en el programa, variable `ratio`) $r = \frac{h_t^2 c^2}{h_s^2}$, podemos intuir que el valor más adecuado puede ser $r = 1/2$, en analogía con problemas anteriores.

El desarrollo del programa es pesado, pero eficaz. Definimos un "rectángulo" (en realidad, cuadrado), de lado $2n_x \times 2n_y$. También definimos tres matrices, $u(i,j)$, $uu(i,j)$, $uuu(i,j)$, que representarán la perturbación en tres instantes sucesivos. En las líneas (12-22) se definen unos parámetros que necesitaremos a lo largo del programa. En las líneas (23-37) definimos los valores iniciales de la perturbación, a partir de una función que tiene unos "picos" positivos en $(0.5, 0.5)$ y $(-0.5, -0.5)$, y un mínimo en medio de esos puntos. Después se va anvanzando en el tiempo, a partir de la ecuación 8.3. Una vez hechos todos los cálculos, se reasignan los valores $uuu = uu$ y $uu = u$. Las líneas (39-70) son un poco de ingeniería computacional para conseguir que el programa escriba, "fotografíe", diez de los instantes de la evolución temporal, aparte del inicial. Después, las líneas (91-93) son formatos y el cierre del programa. Estos fotogramas se recogen de izquierda a derecha y de arriba hacia abajo en las figuras 8.3 y 8.4.

El listado del código FORTRAN se encuentra a continuación:

```
1  ************************************************************************
2  *******   ULTIMO PROGRAMA QUE HAGO EN FISICA COMPUTACIONAL ....*********
3  *******   ... PERO NO EL ULTIMO QUE HARE CON FORTRAN. VOLVERE  *********
4  ************************************************************************
```

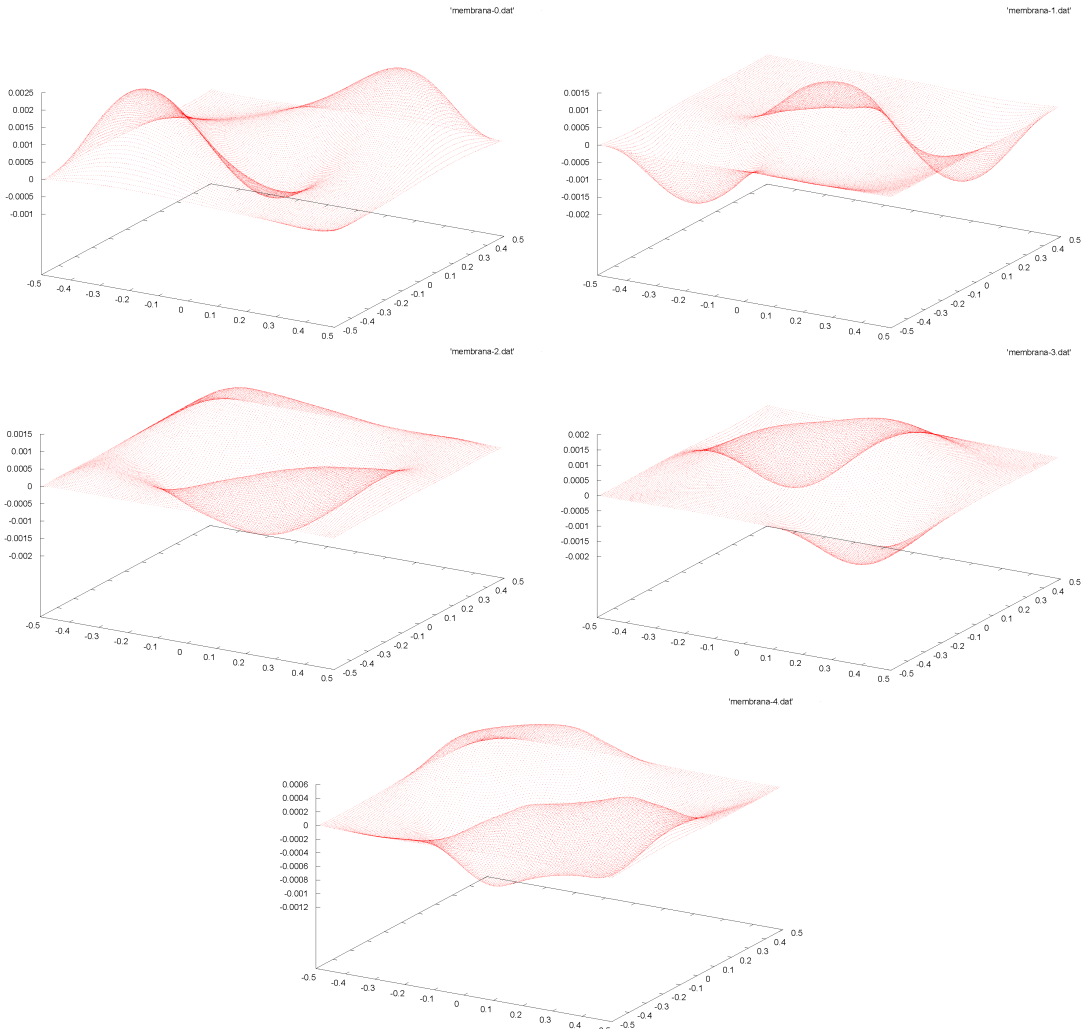

Figura 8.3: Instantes iniciales (0-4)

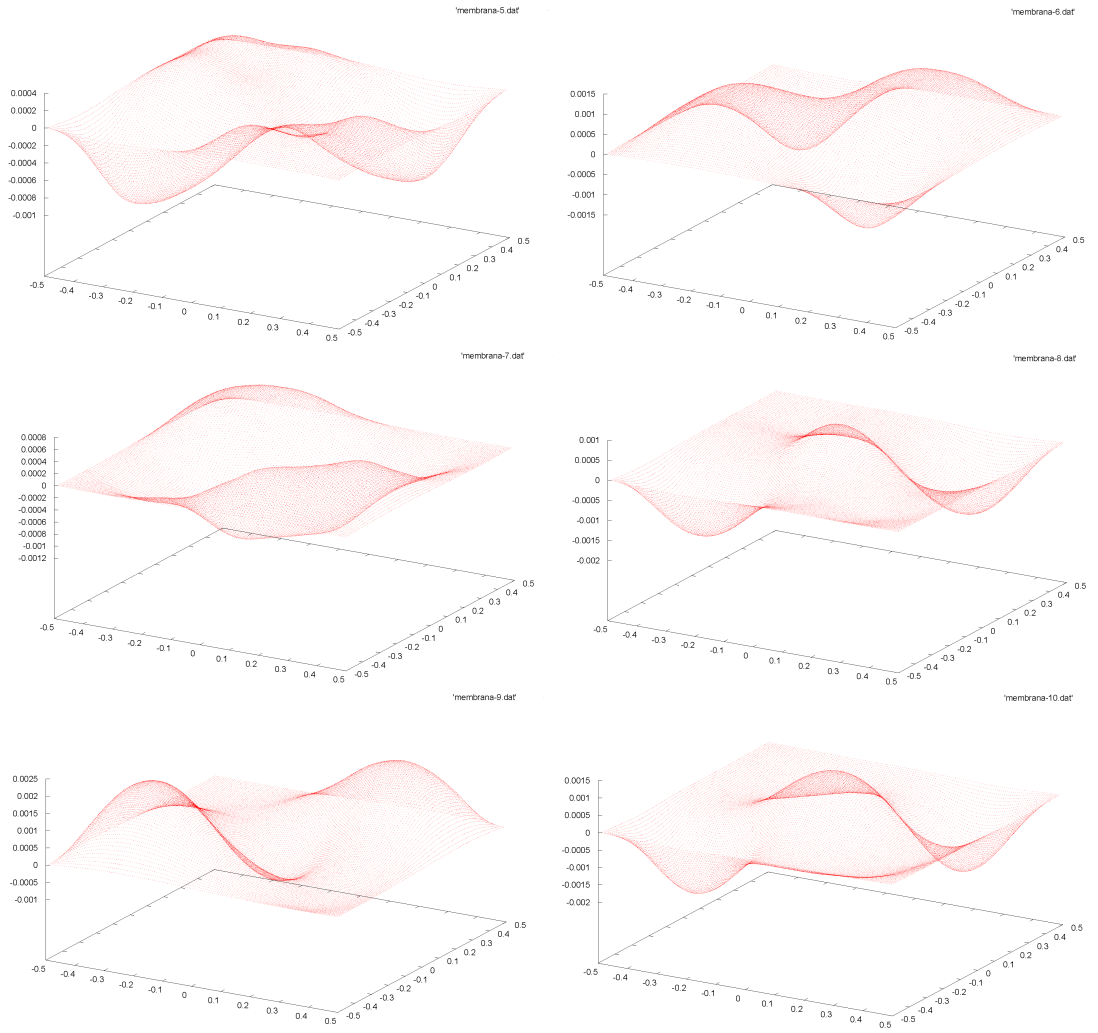

Figura 8.4: Instantes posteriores (5-10)

```
5  *     Se trata de calcular el avance en el tiempo de la vibracion de
6  *     una membrana, partiendo de una configuracion inicial arbitraria
7        parameter (nx=75,ny=75,nt=10000)
8        ! nt ha de ser multiplo de 10 !!!!!!!!!!!!!!!
9        dimension u(-nx:nx,-ny:ny),uu(-nx:nx,-ny:ny),uuu(-nx:nx,-ny:ny)
10 ****** Convenio : x-->i ; y-->j ; t-->k
11 ************************************************************************
12 *     Definicion de parametros
13       c = 1.0
14       rlado = 1.0
15       xymax = rlado/2.0
16       height = rlado
17       hs = rlado/real(2*nx) !hx = hy = hs
18       ratio= 0.5
19       ht = sqrt(ratio)*hs/c
20       stept = nt/10
21       tmax = ht*real(nt)
22       write(*,*)c,hs,ht,ratio,tmax
23 ************************************************************************
24 *     Inicio los valores iniciales de la membrana, asegurandome de que
25 *     en los lados de ella la perturbacion vale u=0. Lo hago para los
26 *     instantes iniciales
27       do i=-nx,nx
28       do j=-ny,ny
29       x=real(i)*hs
30       y=real(j)*hs
31       fey=rlado/4.0
32       u(i,j)=-height*(fey-(x-fey)**2-(y-fey)**2)*(fey-(x+fey)**2-
33      &(y+fey)**2)*(xymax**2-x**2)*(xymax**2-y**2)
34       uu(i,j) = u(i,j)
35       uuu(i,j)= u(i,j)
36       enddo
37       enddo
38 ************************************************************************
39       open(20, file='membrana-0.dat', status='unknown')
40       do i=-nx+1,nx-1
41       x=real(i)*hs
42       do j=-ny+1,ny-1
43       y=real(j)*hs
44          write(20,2000) x,y,u(i,j)
45       enddo
46       enddo
47
48       open(21, file='membrana-1.dat', status='unknown')
49       open(22, file='membrana-2.dat', status='unknown')
50       open(23, file='membrana-3.dat', status='unknown')
```

```
51        open(24, file='membrana-4.dat', status='unknown')
52        open(25, file='membrana-5.dat', status='unknown')
53        open(26, file='membrana-6.dat', status='unknown')
54        open(27, file='membrana-7.dat', status='unknown')
55        open(28, file='membrana-8.dat', status='unknown')
56        open(29, file='membrana-9.dat', status='unknown')
57        open(30, file='membrana-10.dat',status='unknown')
58
59        l=1
60        ll=21
61        do k=1,nt
62           t = real(k)*ht
63           if (l*stept.eq.k) write (ll,1000)t
64           do i=-nx+1,nx-1
65              x=real(i)*hs
66           do j=-ny+1,ny-1
67              y=real(j)*hs
68              chorizo=uu(i-1,j)+uu(i+1,j)+uu(i,j-1)+uu(i,j+1)-4.0*uu(i,j)
69              u(i,j)=2.0*uu(i,j) - uuu(i,j) + ratio*chorizo
70              if (l*stept.eq.k) write (ll,2000) x,y,u(i,j)
71           enddo
72           enddo
73  * Una vez hechos todos los calculos, asignamos de nuevo los valores :
74  * uu=u y uuu=uu
75           do i=-nx+1,nx-1
76           do j=-ny+1,ny-1
77           uuu(i,j)=uu(i,j)
78           uu(i,j)=u(i,j)
79           enddo
80           enddo
81
82           if (l*stept.eq.k) then
83             l  =  l + 1
84             close(ll)
85             ll = ll + 1
86            else
87             continue
88           endif
89        enddo
90  ***************************************************************************
91   1000 format('# Tiempo transcurrido: t = ',f10.5,' s')
92   2000 format(3x,3(g20.12,2x))
93        end
94
```