ÍNDICE

1.- PLANTEAMIENTO DEL PROBLEMA

Realizar un programa en MATLAB que permita señalizar el kilometraje en una carretera definida por un polinomio de interpolación mediante splines paramétricas. El trabajo debe considerar los siguientes aspectos y requisitos:

1.- Obtener ejemplos reales de mapas de carreteras. Una vez transportada la curva a un sistema de referencia adecuado, elegir el tamaño de paso de forma adecuada. Considerar siempre que se trata de una curva plana.

2.- Utilizar el teorema fundamental del cálculo integral y el método de Newton-Rapshon para la resolución del problema, prestando especial atención a que estamos trabajando con interpolación a trozos. Suponer que el primer punto del soporte representa el kilómetro cero. Además de obtener los puntos que dividen la longitud de la curva cada kilómetro, calcular la longitud total entre dos tramos extremos del soporte.

3.- Representar gráficamente el polinomio interpolador y el kilometraje obtenido.

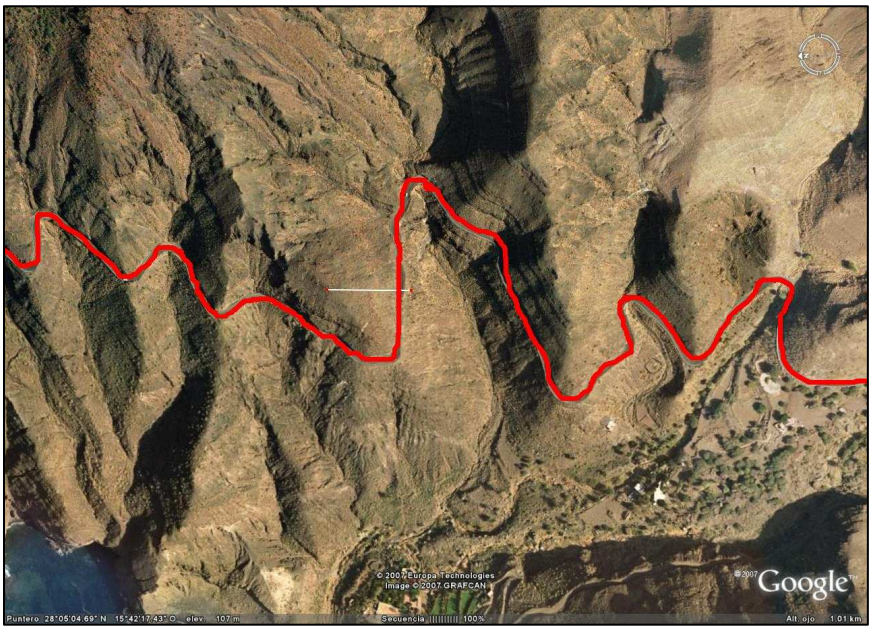
2.- INTRODUCCIÓN:

El documento que se expone a continuación, se ha realizado con la intención de poder calcular el kilometraje de cualquier carretera, colocando un hito cada 1000m. El programa utilizado para poder llevar acabo el proceso descrito anteriormente es el MATLAB (es la abreviatura de *Matrix Laboratory "laboratorio de matrices"*).

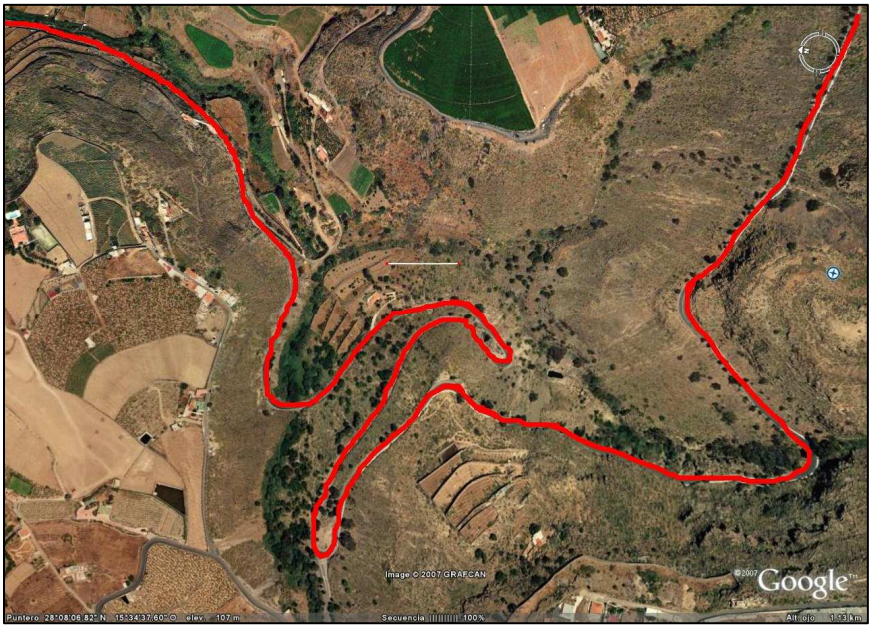
A continuación se expondrán las carreteras que han sido analizadas. Se ha elegido carreteras de la isla de Gran Canaria debido principalmente a que es nuestro entorno más cercano, pero el programa sirve para cualquier vía.

Las carreteras seleccionadas son:

- ***CARRETERA 1:*** *"G.C.-810. Carretera de la Aldea"*. Esta vía se caracteriza por tener curvas sinuosas con dificultades en la visibilidad. Además, se encuentra situada entre barrancos, aumentando de esta manera su peligrosidad. Es una vía de doble sentido pero con ciertos estrechamientos. Se encuentra a unos 100 metros sobre el nivel del mar.



• ***CARRETERA 2:*** *"G.C.-160. Carretera de subida a Moya".* Esta vía se encuentra localizada en la zona norte de la isla, siendo una carretera de doble sentido pero con ciertos estrechamientos tal y como se puede ver en la figura. Se sitúa a unos 350 metros sobre el nivel del mar.



• ***CARRETERA 3:*** *"G.C.-100. Carretera del Barranco de Aguaje".* Esta vía se encuentra situada a lo largo del Barranco de Azuaje. Se caracteriza por tener curvas sinuosas sin mucha visibilidad. Conecta las poblaciones de *"Lomo Blanco y Buen Lugar".* Se sitúa a unos 400 metros sobre el nivel del mar.

3.- FUNDAMENTOS TEÓRICOS

Para poder llevar a cabo el problema que se ha planteado anteriormente, es necesario conocer algunos fundamentos teóricos, los cuales se muestran a continuación.

3.1.- MÉTODO DE INTERPOLACIÓN MEDIANTE SPLINE CÚBICA.

La teoría de la interpolación nos permite encontrar una solución que coincida con una función dada en una serie de datos conocidos. En este caso en particular, estos datos conocidos, son los distintos valores de coordenadas de la carretera.

Lo que queremos, es obtener, a partir de una tabla de parejas (x, y) definida en un cierto intervalo $[a, b]$, el valor de la función para cualquier x perteneciente a dicho intervalo.

Supongamos que disponemos de las siguientes parejas de datos, que definen la trayectoria de nuestra carretera:

$$\boxed{\begin{array}{l} X \Rightarrow x_0, x_1, x_2 ..... x_n \\ Y \Rightarrow y_0, y_1, y_2 ..... y_n \end{array}}$$

El objetivo es encontrar una función continua lo más sencilla posible tal que $f(x_i) = y_i$.

Se dice entonces que la función $f(x)$, es una función de interpolación de los datos representados en la tabla.

Dentro de los distintos métodos para conseguir la interpolación, tenemos el de la interpolación polinomial a trozos, que va a ser la utilizada en este programa.

Una función spline está formada por varios polinomios, cada uno definido sobre un subintervalor, de que sea un método de interpolación a trozos, que se unen entre sí obedeciendo a ciertas condiciones de continuidad.

Supongamos que disponemos de $(n+1)$ (t_i), denominados nodos, tales que $t_0 < t_1 < ... < t_{n+1}$. Supongamos además que se ha fijado un entero $k \geq 0$. Decimos entonces que una función spline de grado k con nodos en $t_0 < t_1 < ... < t_{n+1}$ es una función "S" que satisface ciertas condiciones, las cuales se describirán a continuación:

- En cada intervalo $[t_{i-1}, t_i]$, *"S"* es un polinomio de grado menor o igual a *k.*
- *"S"* tiene una derivada de orden $(k-1)$, continua en $[t_0, t_n]$.

Sabemos que las splines de grado impar, dan funciones más suaves que das de grado par y que la elección de un grado elevado complicaría el problema. Por ello la opción más factible es seleccionar la spline cúbica, es decir, de grado 3, con derivadas de hasta grado 2.

El spline cúbico $(k=3)$ proporciona un excelente ajuste a los puntos tabulados y su cálculo no es excesivamente complejo.

Sobre cada intervalo $[t_0,t_n],[t_1,t_2].....[t_{n-1},t_n]$, *"S"* está definido por un polinomio cúbico diferente. Sea S_i, el polinomio cúbico que representa *"S"* en el intervalo $[t_i,t_{i+1}]$, por tanto:

$$S(x)=\begin{cases} S_0(x).....x\in[t_0,t_1) \\ S_1(x).....x\in[t_1,t_2) \\ \\ S_{n-1}(x).....x\in[t_{n-1},t_n) \end{cases}$$

Los polinomios S_{i-1} y S_i interpolan el mismo valor en el punto t_i, es decir, se cumple:

$$S_{i-1}(t_i)=y_i=S_i(t_i)\Rightarrow para\Rightarrow 1\le i\le n-1$$

Esto garantiza que S es continuo en todo el intervalo. Además, se supone que S' y S'' son continuas, condición que se emplea en la deducción de una expresión para la función spline cúbico.

Como se había comentado, este método lleva consigo dos derivadas, decimos entonces que si $S_i(x)$ es cúbica, su segunda derivada será lineal, siendo $S''(t_i)=z_i$, a su vez, si $S''(t_{i+1})=z_{i+1}$.

Aplicando las condiciones de continuidad del spline S y la de las derivadas S' y S'', es posible encontrar la expresión analítica del spline.

La expresión resultante es:

$$S_i(x)=\frac{z_i}{6h_i}(t_{i+1}-x)^3+\frac{z_{i+1}}{6h_i}(x-t_i)^3+(\frac{y_{i+1}}{h_i}+\frac{z_{i+1}h_i}{6})(x-t_i)+(\frac{y_i}{h_i}-\frac{z_ih_i}{6})(t_{i+1}-x)$$

En esta expresión $h_i=t_{i+1}-t_i$ y $z_0,z_1.....z_n$ son incógnitas. Para determinar sus valores, utilizamos las condiciones de continuidad que deben cumplir estas funciones. El resultado sería el siguiente:

$$h_{i-1}z_{i-1}+2(h_i+h_{i-1})\cdot z_i+h_iz_i+1=\frac{6}{h_{i-1}}\cdot(y_{i+1}-y_i)-\frac{6}{h_{i-1}}\cdot(y_i-y_{i-1})$$

La ecuación anterior, donde $i=1,2....,n-1$, genera un sistema de $(n-1)$ ecuaciones lineales con $(n+1)$ incógnitas $z_0,z_1.....z_n$.

Si hacemos $z_0 = z_1 = 0$, para la resolución del sistema de ecuaciones generado, la función spline resultante se denomina spline cúbico natural y el sistema de ecuaciones lineal expresado en forma matricial es:

$$\begin{pmatrix} u_1....h_1 & & & & \\ h_2....u_2.....h_3 & & & & \\ .....h_2.....u_3.....h_3 & & & & \\ & & & & \\ .....h_{n-3}.....u_{n-2}.....h_{n-2} & & & & \\ .....h_{n-2}....u_{n-1} & & & & \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \\ z_{n-2} \\ z_{n-3} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ \\ v_{n-2} \\ v_{n-3} \end{pmatrix}$$

En donde:

$$b_i = \frac{6}{h_i}(y_{i+1} - y_i)$$

$$u_i = 2(h_{i-1} + h_1) - \frac{h_i^2 - 1}{u_i - 1}$$

$$v_i = b_i - b_{i-1} - \frac{h_{i-1}}{u_{i-1}} v_{i-1}$$

$$h_i = t_{n+1} - t_i$$

3.2.- MÉTODO DE INTERPOLACIÓN MEDIANTE SPLINE PARAMÉTRICA.

Como el problema, nos pide estudiar una carretera multivaluada, tendremos que utilizar el método de interpolación mediante spline paramétrica, la cual se define a continuación.

Entre los modelos de trazador cúbico (splines) que hemos estudiado, los splines no paramétricos (es decir, aquellos que dados (x_i, y_i) tal que $i = 0...n$ son capaces de interpolar valores en x ó en y, dependiendo de cómo se construya el trazador) han demostrado ser capaces de aproximar curvas en el plano, siempre que estas sean abiertas.

El problema propuesto sugiere como entrada un conjunto de (x_i, y_i), que describen la frontera de algún objeto bidimensional, tal que entre algunos de estos puntos podrían repetirse valores de x e y. Esto trae como consecuencia el inconveniente de que algunos puntos no pueden ser expresados como función de x ni como función de y.

Al introducir la ecuación $P(t) = (x(t), y(t))$ lo que hacemos es obtener los puntos interpolados mediante dos splines:

$$x_i = spline(x, t_i)$$
$$y_i = spline(y, t_i)$$

Siendo t_i un conjunto de puntos, apropiadamente seleccionados. De esta forma, tenemos una spline que interpola a las x con respecto a t y otro que interpola a las y con respecto a t. Sea t un vector escogido apropiadamente, mediante alguna de las forma indicada.

Luego escribimos estos dos vectores x_i e y_i, asumiendo que corresponden a las abscisas y ordenadas, respectivamente. Al hacer esto, igualamos el parámetro t y obtenemos valores en (x, y) que nos permiten modelar la frontera de cualquier figura bidimensional.

Es importante mencionar, que de las distintas estrategias que se proponen para calcular el parámetro t, aquella que utiliza la suma acumulada de las distancias euclidianas es la que mejor funciona, y la explicación que se le podría dar a ese hecho estriba en que cuando la distancia entre dos puntos (x, y) consecutivos es menor, entonces la distancia entre los t que se encuentran cercanos a ellos también será menor, aportando un mayor numero de puntos de interpolación donde haga falta mas detalle; de esta misma manera, cuando la distancia entre dos puntos (x, y) consecutivos es mayor, la distancia entre los t correspondientes será mayor.

En este caso, podríamos decir que el parámetro es adaptativo, y en la práctica aporta mejores resultados (curvas más suaves, por ejemplo) que los otros métodos, los cuales toman simplemente un vector de n números consecutivos, sin que estos reflejen ninguna relación con los datos de entrada que se están interpolando.

3.3.- MÉTODO DE DERIVACIÓN NUMÉRICA

En los problemas de derivación numérica, lo que se pretende es aproximar la derivada de una función dada en un punto concreto.

Sabemos que la definición de la derivada de una función en un punto "a" corresponde a la siguiente expresión:

$$\boxed{f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}}$$

Para la realización del programa propuesto, utilizaremos derivadas de primer orden y de entre las distintas fórmulas a utilizar, usaremos una de dos puntos.

Estos puntos serán equidistantes una distancia h, respecto de otro punto c, de esta manera obtendremos la derivada de la siguiente forma

$$\boxed{(c-h) \to (c+h) \Rightarrow f'(c) = \frac{f(c+h) - f(c-h)}{2h}}$$

Siendo el error cometido, el siguiente:

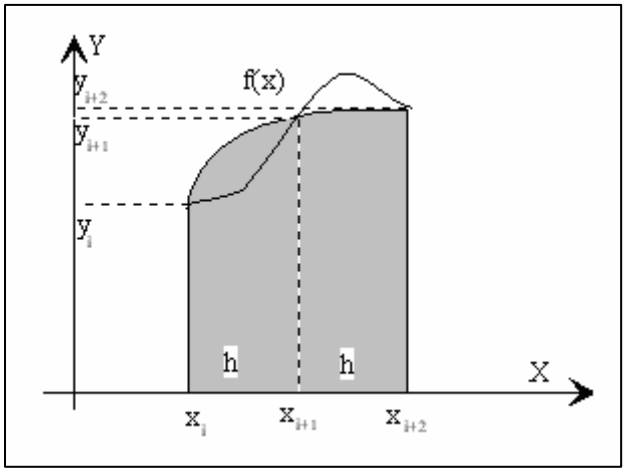
Como es lógico, cuanto mayor sea el número de puntos seleccionados, menor será el error.

$$\boxed{R(f) = -\frac{h}{6} \cdot f'''(\zeta)}$$

3.4.- INTEGRACIÓN NUMÉRICA

En este procedimiento, se toma el intervalo de anchura *2h*, comprendido entre x_i y x_{i+2}, y se sustituye la función *f(x)* por la parábola que pasa por tres puntos (x_i, y_i), (x_{i+1}, y_{i+1}), y (x_{i+2}, y_{i+2}). El valor del área aproximada, sombreada en la figura, se calcula con un poco más de trabajo y el resultado es:

$$\frac{h}{3}(y_i + 4y_{i+1} + y_{i+2})$$



La simple inspección visual de esta figura y la que describe el procedimiento de los trapecios nos confirma que el método de Simpson deberá ser mucho más exacto que el procedimiento del trapecio. El área aproximada en el intervalo *[a, b]* es;

$$\int_a^b f(x)dx \approx \frac{h}{3}(y_o + 4y_1 + y_2) + \frac{h}{3}(y_2 + 4y_3 + y_4) + ... + \frac{h}{3}(y_{n-2} + 4y_{n-1} + y_n)$$

Agrupando términos tenemos:

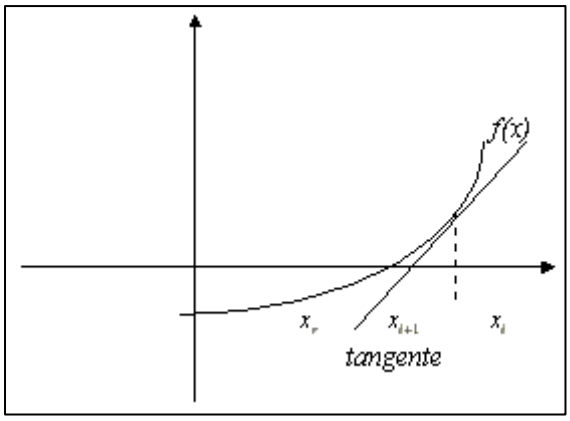
$$\int_a^b f(x)dx \approx \frac{h}{3}[(y_0 + y_n) + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2})]$$

El primer paréntesis, contiene la suma de los extremos, el segundo, la suma de los términos de índice impar, y el tercero la suma de los términos de índice par. En el método de Simpson, el número de divisiones *n* debe de ser par. En el caso de que el usuario introduzca un número impar el programa lo convierte en el número par siguiente.

3.5.- FÓRMULA DE NEWTON-RAPSHON

Este método, el cual es un método iterativo, es uno de los más usados y efectivos. A diferencia de otros métodos, el de Newton-Rapshon no trabaja sobre un intervalo sino que basa su fórmula en un proceso iterativo.

Supongamos que tenemos la aproximación x_i a la raíz x_r de $f(x)$.



Trazamos la recta tangente a la curva en el punto $(x_i, f(x_i))$ ésta cruza al eje x en un punto x_{i+1} que será nuestra siguiente aproximación a la raíz x_r.

Para calcular el punto x_{i+1}, calculamos primero la ecuación de la recta tangente. Sabemos que tiene pendiente

$$\boxed{m = f'(x_i)}$$

Y por lo tanto la ecuación de la recta tangente es:

$$\boxed{y - f(x_i) = f'(x_i)(x - x_i)}$$

Hacemos $y = 0$:

$$\boxed{-f(x_i) = f'(x_i)(x - x_i)}$$

Y despejamos x:

$$\boxed{x = x_i - \frac{f(x_i)}{f'(x_i)}}$$

Esta ecuación es la fórmula iterativa de Newton-Raphson para calcular la siguiente aproximación:

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \Rightarrow si \Rightarrow f'(x_i) \neq 0}$$

Se tiene que destacar que el método de Newton-Raphson no trabaja con intervalos donde nos asegure que encontraremos la raíz, y de hecho no tenemos ninguna garantía de que nos aproximaremos a dicha raíz. Desde luego, existen ejemplos donde este método no converge a la raíz, en cuyo caso se dice que el método diverge. Sin embargo, en los casos donde si converge a la raíz lo hace con una rapidez impresionante, por lo cual es uno de los métodos preferidos por excelencia.

También se observa que en el caso de que $f'(x_i) = 0$, el método no se puede aplicar. De hecho, vemos geométricamente que esto significa que la recta tangente es horizontal y por lo tanto no intersecta al eje x en ningún punto, a menos que coincida con éste, en cuyo caso x_i mismo es una raíz de $f(x)$.

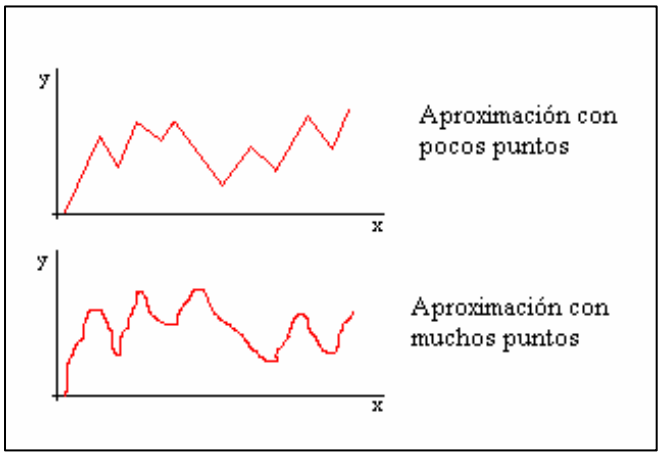
4.- SOLUCIÓN ADOPTADA

Una vez definidas las funciones a utilizar, se expondrá el planteamiento del programa en MATLAB.

4.1.- DEFINICIÓN DE LA FUNCIÓN QUE REPRESENTA LA CARRETERA.

Como es lógico, para poder definir una carretera, se necesita un polinomio. Debido a esto se usará para ello un polinomio de interpolación.

Un aspecto importante a tener en cuenta, es que como se quería ajustar lo máximo posible la función a la realidad, se tomaron muchos puntos para que la aproximación fuera lo más exacta posible. Si no hubiese sido así, nos hubiese quedado un resultado lejano a dichas carreteras.



Resumiendo, tenemos que para poder empezar el programa necesitamos introducir en un fichero los puntos que representan a nuestra carretera, debido a esto se crearán dos columnas, una para los valores de *"x"* y otra para los valores de *"y"*. Una vez realizado esto, con la función spline, conseguiremos construir el polinomio que represente dicho trazado.

4.2.- CÁLCULO DEL KILOMETRAJE.

Para calcular la longitud de una curva paramétrica se ha utilizado la siguiente expresión:

$$\int_a^b \sqrt{f'(t)^2 + g'(t)^2}$$

Siendo "a" y "b" los límites de integración. $a = t_1 \Rightarrow b = t_{i+1}$

Esto quiere decir, que cuando:

$$\int_{t_i}^{t_{i+1}} \sqrt{f'(t)^2 + g'(t)^2} = 1$$

La longitud del tramo de la curva entre t_1 y t_{i+1} vale un kilómetro, debiéndose tener en cuenta la escala en cada caso.

Para poder obtener el valor de "t" para que la ecuación de 1, se aplica el método de Newton-Rapshon cuya fórmula es la siguiente:

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

Siendo f(t):

$$\int_{t_i}^{t_{i+1}} \sqrt{f'(t)^2 + g'(t)^2}\, dt - 1 = 0$$

Obteniendo la siguiente expresión:

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)} = t_i - \frac{\int_{t_i}^{t_{i+1}} \sqrt{f'(t)^2 + g'(t)^2}\, dt - 1}{\frac{d}{dx}\int_{t_i}^{t_{i+1}} \sqrt{f'(t)^2 + g'(t)^2}\, dt - 1}$$

Para obtener la solución se creará un bucle donde se pondrá a funcionar esta fórmula de modo que nos de la "t" en cada nuevo kilómetro.

Vemos que en la expresión aparecen derivadas e integrales, para resolver las derivadas se usará una subrutina creada por nosotros y la integral se resuelve con una función de la librería de MATLAB (quad) que se resuelve por cuadratura de Simpson.

4.3.- REPRESENTACIÓN GÁFICA.

Finalmente para representar la carretera gráficamente, se evaluarán los resultados obtenidos por la funciones spline, con estos datos se dibujará la curva y con los datos obtenidos de aplicar Newton-Rapshon dibujaremos una referencia en cada kilómetro.

5.- PROGRAMA

El programa que se va a desarrollar a continuación realiza el kilometraje de una carretera multivaluada cuyas medidas han sido grabadas en un fichero, siendo la primera columna las coordenadas en el eje *"x"* y la segunda las de las coordenadas en el eje *"y".*

5.1.- FUNCIONES DE LA LIBRERÍA DE MATLAB UTILIZADAS.

- ***load('nombrefichero', '-ASCII'):***

Devuelve las variables contenidas en el fichero especificado. En este caso devolverá una matriz Mx2 que serán las medidas de la carretera.

- ***size(A):***

Nos devuelve las dimensiones de la matriz A. De esta forma podremos averiguar el número de medidas tomadas.

- ***spline(X,Y):***

Nos devuelve la forma polinomial a trozos del spline cúbico interpolador para los puntos (x, y) introducidos, usando la función que se va a explicar a continuación.

- ***ppval(PP,XX):***

Podremos evaluar dichos splines en los puntos indicados en XX.

- ***length(X):***

Calcula el número de elementos de un vector X.

- ***handle:***

Es una referencia de una función.

- ***quad(FUN, A, B, TOL, TRACE, P1, P2,...):***

Intenta aproximar la integral de la función FUN de A a B con un error de 1e-6 usando reiteradamente el método adaptativo de la cuadratura de Simpson.

TOL: representa la tolerancia, la cual, entre mayor valor tenga, peor será la evaluación de la función aunque el proceso de resolución sea más rápido.

TRACE: con valores no nulos de trace mostramos los valores soluciones de cada iteración.

P1, P2: son argumentos adicionales que se pasan directamente a la función FUN.

- ***istruct(arg1):***

Permite saber si arg1 es o no una estructura.

- ***fix(x):***

Redondea hacia el entero más próximo a cero.

- ***min(x,y):***

Devuelve el valor más pequeño entre x e y.

- ***abs(f):***

Nos devuelve el valor absoluto de f.

- ***Inputname(x):***

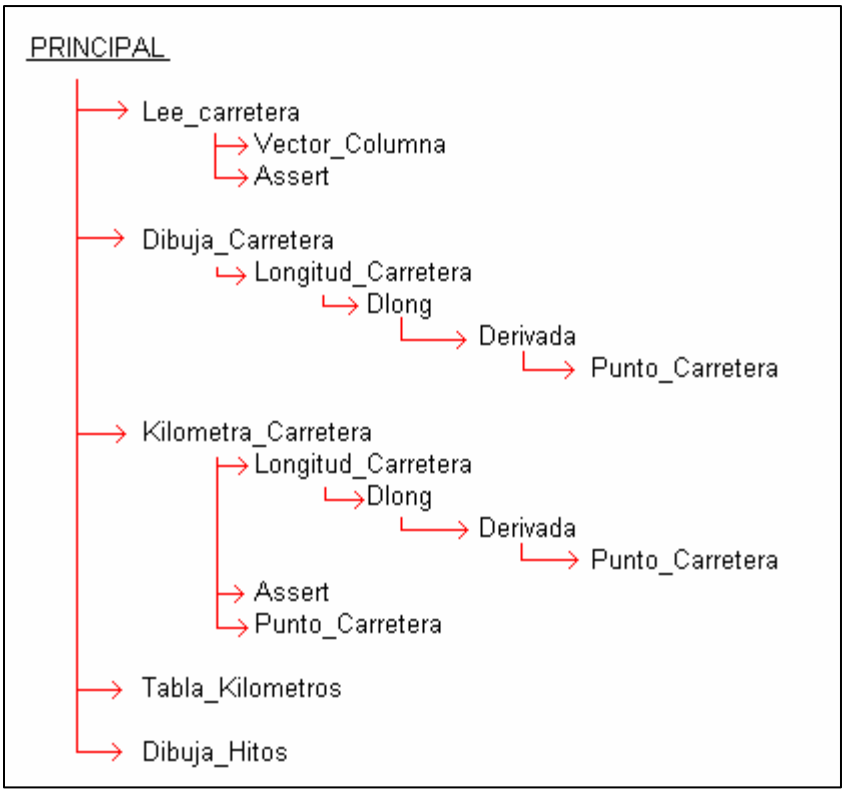
Argumento de la función anterior. Devuelve el que se sitúa en la posición x.

- ***Isnumeric:***

Devuelve verdadero si el valor es numérico.

5.2.- SUBRUTINAS EMPLEADAS.

A continuación se muestra un esquema, donde se puede visualizar de manera sencilla la estructura del programa creado.



- ***Principal:*** Programa principal o coordinador de las restantes subrutinas que se especifican a continuación.

- ***Lee_carretera:*** crea la estructura carretera con todos lo necesario para trabajar con ésta.

    - c = LEE_CARRETERA(nombrefichero) lee a partir del fichero especificado los puntos de la carretera.

    - c = LEE_CARRETERA(X,Y) genera la estructura carretera a partir de los vectores X e Y.

La función devuelve la estructura CARRETERA, que contiene los siguientes campos:

- x, y: Puntos X e Y iniciales de la carretera, para cada t.
- N: Número de puntos. => t comprendido entre 1 y N.
- escala: Número de Kilómetros que representa cada unidad. Tiene que ser positivo.
- nintervalos: Número de intervalos en los que está dividida la carretera = N - 1.
- nombrefichero: Nombre del fichero donde fueron leídos los puntos de la carretera.
- splineX: A su vez, es una estructura que almacena el spline cúbico de X = f(t). Sus campos más importantes son:
  - order: Número de coeficientes por polinomio interpolador.
  - breaks: valores de t donde se define cada x o y según el caso.
  - pieces: Número de intervalos o polinomios = nintervalos.
  - coefs: contiene los coeficientes a, b, c, d para los polinomios interpoladores de cada intervalo.
- splineY: contiene el spline cúbico para Y = g(t).

- ***Vector_columna:*** comprueba que los vectores X e Y introducidos por el usuario son vectores columna y si no es así los modifica.

- ***Assert(a, mensaje):*** Confirma la condición 'a' y si no se cumple devuelve el mensaje correspondiente.

- ***Dibuja_carretera:*** dibuja la carretera con la resolución determinada.
  - DIBUJA_CARRETERA(carretera): por defecto se asume resolución de 0.25.
  - DIBUJA_CARRETERA(carretara,resolucion): especifica una resolución. Se recomienda que sea menor que 1.

- ***Longitud_carretera:*** Devuelve la longitud en Kilómetros de la carretera en un tramo dado.
  - L = LONGITUD_CARRETERA(carretera) devuelve la longitud de toda la carretera.
  - L = LONGITUD_CARRETERA(carretera,a,b) devuelve la longitud de la carretera desde t = a hasta t = b, 't' tiene que estar comprendido entre 1 y carretera.N, que es el numero de puntos de la carretera.
  - L = LONGITUD_CARRETERA(carretera,t0) devuelve la longitud de la carretera desde t = t0.

- ***Dlong:*** función diferencial de longitud utilizada en la integración.

    - dl = sqrt(dx^2 + dy^2) = sqrt(f'(t)^2 + g'(t)^2)*dt donde:
      dx = f'(t)dt
      dy = g'(t)dt

Acepta las formas DLONG(carretera,t) y DLONG(t,carretera) para poder usarse con QUAD.

- ***Derivada:*** derivada en x e y de la carretera en el punto definido por t calculada mediante la fórmula de dos puntos: f'(x) = (f(x+h) - f(x-h)) / 2*h

    - d = DERIVADA(carretera,t) devuelve la derivada de x e y respecto de t respecto al trazo de la carretera.
      d(1) = dx / dt
      d(2) = dy / dt

- ***Punto_carretera:*** devuelve el punto de la carretera para el valor t.

    - p = PUNTO_CARRETERA(carretera,t) devuelve el punto para esa t
      p(1) = x
      p(2) = y

    - p = PUNTO_CARRETERA(carretera,T) donde T es un vector, devuelve el punto para cada t
      p(:,1) contiene los respectivos valores de x
      p(:,2) contiene los respectivos valores de y

    - p = PUNTO_CARRETERA(t,carretera)

    - p = PUNTO_CARRETERA(T,carretera)

- ***Kilometra_carretera:*** Calcula la posición de los hitos kilométricos en la carretera teniendo en cuenta que deben estar separados la misma distancia.

    - T = kilometra_carretera(carretera) devuelve un vector con los distintos valores de t donde coincide cada hito kilométrico separados una distancia por defecto de 1 Kilómetro.

    - T = kilometra_carretera(carretera,distancia) especifica un valor de distancia diferente a la distancia por defecto.

    - [T,puntos,Iter] = kilometra_carretera(carretera) devuelve además un vector puntos con los puntos evaluados en cada t y un vector con el número de iteraciones que fueron necesarias para calcular cada valor.

- ***Tabla_Kilometros:*** Muestra en una tabla los resultados tras haber calculado los hitos kilométricos de la carretera.
    - TABLA_KILOMETROS(T,puntos,Iter). recibe como parámetros la salida de la función KILOMETRA_CARRETERA.

- ***Dibuja_Hitos:*** Dibuja los hitos kilométricos descritos por la matriz puntos. Esta matriz la devuelve la función KILOMETRA_CARRETERA. DIBUJA_HITOS (puntos)

5.3.-VARIABLES.

A continuación, se muestran las variables utilizadas a la hora de crear el programa, las cuales son:

- ***D:*** vector que almacena los ficheros tipo 'carretera*.mat'
- ***f_num:*** entrada del ejemplo solicitado por el usuario.
- ***escala:*** escala a la que se quiere la carretera.
- ***c:*** estructura que contiene los datos de la carretera.
- ***Ppuntos_leidos:*** datos que se extraen del fichero.
- ***carretera.escala:*** da la escala de la carretera estudiada.
- ***carretera.x:*** da los valores de *"x"* para los puntos leídos.
- ***carretera.y:*** da los valores de *"y"* para los puntos leídos.
- ***carretera.N:*** da el número de puntos.
- ***carretera.nintervalos:*** da el número de intervalos.
- ***carretera.splineX:*** devuelve la estructura obtenida por la función spline (x, t).
- ***carretera.splineY:*** devuelve la estructura obtenida por la función spline (y, t).
- ***s:*** dimensión del vector introducido por el usuario.
- ***resolución:*** paso para evaluar cada punto de la carretera para su posterior representación gráfica.
- ***kmTotal:*** longitud o kilometraje total de la carretera.
- ***a,b:*** punto inicial y final en la evaluación de la carretera.
- ***p:*** matriz que contiene a los puntos evaluados.
- ***Distancia:*** distancia entre hitos, en este caso un kilómetro.
- ***T:*** vector que contiene los valores (x, y) de la situación de cada hito kilométrico, es decir, sus coordenadas. *(Aparece otro vector T que contiene los t que el usuario a creado para poder obtener las spline de x e y).*
- ***Iter:*** número de iteraciones del método para cada punto.
- ***MAX_NEWTON_ITER:*** límite de iteraciones impuesto para el método de Newton Rapshon.
- ***PASO_t:*** paso entre los valores de t.
- ***nhitos:*** número de hitos.
- ***niter_hitos:*** número de iteraciones en la obtención de los hitos.
- ***f:*** longitud de cada tramo de la carretera en el proceso de kilometraje.
- ***puntos:*** evaluación de (x, y) para cada t.

5.4.- SECUENCIA DEL PROGRAMA.

A continuación, en este apartado, se comentará, cual es la función de cada una de las subrutinas que componen dicho programa.

PROGRAMA PRINCIPAL

```
function [c,T,puntos,Iter,PASO_t] = principal

        fprintf(1,'\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++');
        fprintf(1,'\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++');
        fprintf(1,'\n              KILOMETRAJE DE UNA CARRETERA REAL');
        fprintf(1,'\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++');
        fprintf(1,'\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++\n');
        fprintf(1,'\n  Indique el fichero desde que se leera el trazado de la carretera: \n');
```

Mostramos todos los ficheros de carreteras del directorio de trabajo.

```
        D = dir('carretera*.mat');
        fprintf(1,'\n  Ficheros de ejemplo disponibles: \n');
        for i = 1:length(D)
                fprintf(1,'    %2d: %s\n',i,D(i).name);
        end
```

Seguidamente el usuario introduce el nombre del ejemplo que quiere estudiar.

```
        f_num = input('   >');
        while ~isnumeric(f_num) || f_num > length(D) || f_num < 1
                fprintf(1,'\n  Introduzca un número de la lista: \n');
                f_num = input('   >');
        end
```

El usuario introduce la escala

```
        fprintf(1,'\n  Indique la escala: número de kilómetros por unidad (p.e. 0.1 para
        0.1 Kilómetros cada unidad)\n');
        escala = input(' >');

        while ~isnumeric(escala) || escala <= 0
                fprintf(1,'\n   Introduzca un número mayor que 0. (p.e. 0.1 para 0.1
                Kilómetros cada unidad:\n');
                escala = input('   >');
        end

        fprintf(1,'\n  Procesando el fichero %s\n',D(f_num).name);
```

A continuación, el programa llama a la función lee_carretera y de esta manera se puede introducir el nombre del fichero que tiene que leer y la escala correspondiente.

```
c = lee_carretera(D(f_num).name,escala);
figure;
```

Una vez realizado el paso anterior, el programa principal, solicita la función dibuja_carretera en la que dicho programa introducirá la información obtenida en la anterior subrutina.

```
dibuja_carretera(c,0.25); %la mostramos

fprintf(1,'\n  Pulse enter para calcular el kilometraje\n');
input('');
```

El siguiente paso que realiza dicho programa principal, es llamar a la función kilometra_carretera, a la cual se le introducen los datos de la carretera y esta nos devuelve los datos de los hitos kilométricos.

```
[T,puntos,Iter,PASO_t] = kilometra_carretera(c); %ejecutamos el algoritmo de kilometraje
```

La función tabla_kilometros saca por pantalla los datos de los hitos.

```
tabla_kilometros(T,puntos,Iter); %resultados de la ejecucion

hold on;
```

Por último, la función dibuja_hitos, coloca una referencia en las coordenadas “x” e “y”, los cuales son introducidos con el vector puntos.

```
dibuja_hitos(puntos); %mostramos los hitos kilométricos calculados
return;
```

SUBRUTINA “LEE_CARRETERA”

En esta subrutina, dependiendo del número y del tipo de argumentos, se pueden introducir los datos de cuatro maneras distintas, como por ejemplo:

- Nombre del fichero.
- Nombre del fichero y escala.
- Dos vectores
- Dos vectores y la escala.

```
function carretera = lee_carretera(arg1,arg2,arg3)
```

A continuación se cargan los datos del trazador desde el fichero indicado.

```
if nargin < 3 && isa(arg1,'char')

  nombrefichero = arg1;
```

Se extrae la información del fichero y se almacena en “puntos_leidos”, luego la primera columna se copia en un vector X y la segunda en otro Y.

```
puntos_leidos = load(nombrefichero, '-ascii');
  X = puntos_leidos(:,1);
  Y = puntos_leidos(:,2);
  if nargin < 2
```

En cuanto a la escala, esta será 1 por defecto.

```
        escala = 1;
  else
```

También la escala puede ser introducida por el usuario, tal como se ve a continuación.

```
 escala = arg2;
 end
```

Se introducen los datos con vectores, la función vector_columna confirma que se han introducido los datos en un vector columna y si no es así, los traspone.

```
elseif nargin > 1
        X = vector_columna(arg1);
        Y = vector_columna(arg2);
        nombrefichero = '[introducido a mano]';
   if length(X) ~= length(Y)
                error('el tamaño de los vectores tiene que ser el mismo');
   end
        if nargin < 3
                escala = 1;
        else
                escala = arg3;
   end
else
        error('numero de argumentos erroneo. consulte la ayuda');
end
```

Dentro de esta subrutina, existe otra, denominada “assert” la cual confirma que la escala sea mayor que 0 y si no es así saca por pantalla el mensaje ‘la escala tiene que ser mayor que 0’

```
assert(escala > 0,'la escala tiene que ser mayor que 0');
```

Por último el resultado que obtenemos de la subrutina “lee_carretera” es el siguiente:

```
carretera.nombrefichero = nombrefichero;
carretera.escala = escala;
carretera.x = X;
carretera.y = Y;
carretera.N = size(carretera.x,1);
carretera.nintervalos = carretera.N -1;
```

Seguidamente se crea un vector T, para poder de esta manera obtener los polinomios interpoladores.

```
T = 1:carretera.N;
```

```
        carretera.splineX = spline(T,carretera.x); %X(t)
        carretera.splineY = spline(T,carretera.y); %Y(t)

return;
```

SUBRUTINA "DIBUJA_CARRETERA"

```
function dibuja_carretera(carretera,resolucion)
        if nargin < 2
```

Esta subrutina, introduce un valor de resolución por defecto, en caso de que no se haya puesto anteriormente.

```
        resolucion = 0.25;
        end
```

Se evalúa el polinomio interpolador de "x" e "y" para cada valor de "t" de forma que se pueda dibujar la carretera.

```
        x = ppval(carretera.splineX,[1:resolucion:carretera.N]);
        y = ppval(carretera.splineY,[1:resolucion:carretera.N]);
```

La subrutina "longitud_carretera" nos devuelve la longitud total de la carretera introduciéndole la estructura carde dicha vía.

```
        kmTotal = longitud_carretera(carretera);
```

Por último, esta subrutina dibuja la carretera en azul y con la función "handle" se hace referencia a la función "plot", seguidamente, lo que se hace es poner el título a la figura y a continuación se define el nombre del eje de abscisas y el de ordenadas.

```
        handle = plot(x,y,'b-');
        title(sprintf('Trazado de la carretera "%s". KM TOTALES:
%3.4f',carretera.nombrefichero,kmTotal));
        xlabel(sprintf('x (%.2d Kilómetros)',carretera.escala));
        ylabel(sprintf('y (%.2d Kilómetros)',carretera.escala));
```

Se puede aumentar el grosor de la línea con los siguientes comandos.

```
        set(handle,'LineWidth',1.5);

return
```

SUBRUTINA "KILOMETRA_CARRETERA"

Esta subrutina calcula los puntos de la carretera donde se encuentra cada kilómetro, nos devuelve los valores de t, x e y de cada kilómetro, el número de iteraciones y la diferencia entre t_i y t_{i+1}.

```
function [T,puntos,Iter,PASO_t] = kilometra_carretera(c,distancia)
   if nargin < 2
```

Por defecto, se separa cada hito una unidad.

```
      distancia = 1;
   end
```

Se empieza a hacer el recorrido desde el principio.

```
t0 = 1;
```

Inicializamos nhitos.

```
nhitos = 1;
```

Se introduce en T, el lugar donde se encuentra el primer hito, en este caso va a ser en el kilómetro 0.

```
T(1) = t0;
```

Se inicializa el número de iteraciones.

```
Iter(1) = 0;
```

Se establece el último valor de “t” para tener un límite de referencia.

```
tfinal = c.N;
TOL = 0.000001*c.escala;
MAX_NEWTON_ITER = 40;
```

Se mete la variable “h” para la derivación.

```
h = 0.01;

PASO_t = 1;

fprintf('\nKilometrando la carretera "%s" a escala x,y = %d Km. [Usando paso t = %d]\n',c.nombrefichero,c.escala,PASO_t);
fprintf('Calculando: .');
```

Seguidamente se calcular el número de hitos de debe salir en pantalla.

```
sti_hitos = fix(longitud_carretera(c) / (distancia - TOL));
```

Se buscan hitos hasta que lleguemos al final de la carretera o la carretera no sea suficientemente larga y el siguiente paso será, el de inicializar el número de iteraciones del método.

```
niter_hitos = 0;
```

Mientras “t0” sea menor que el “tfinal” y además el número de iteraciones del programa general sea menor que el número estimado o la longitud del tramo sea mayor que un kilómetro, se tendrá que cumplir que el número de iteraciones del método general sea menor que 1000.

```
while t0 < tfinal && (niter_hitos < sti_hitos || longitud_carretera(c,t0) >= (distancia-TOL)) && niter_hitos < 1000
```

Avanzamos hasta acercarnos suficiente al “t” solución.

```
t = fix(t0) + PASO_t;
```

Mientras la longitud del tramo sea menor que un kilómetro y “t” no sea todavía la “tfinal” hacemos lo siguiente:

```
while longitud_carretera(c,t0,t) < distancia && t < tfinal
   t = t + PASO_t;
end
t = min(t,tfinal);
```

NOTA: Por si rebasamos el límite, aplicamos Newton-Rapshon para refinar “t” tal que:

f(t)=longitud_carretera(c,t0,t)-distancia = 0
ti+1 = ti - f(ti)/f'(ti)

```
 niter = 0;
f = longitud_carretera(c,t0,t) - distancia;
```

Mientras el valor absoluto de “f” sea mayor que la tolerancia y el número de iteraciones por hito sea menor que el límite máximo, se tendrá:

```
while abs(f) > TOL && niter < MAX_NEWTON_ITER
```

Aplicamos la fórmula de la derivada de dos puntos.

```
 derivada_f = (longitud_carretera(c,t0,t+h) - longitud_carretera(c,t0,t-h))/(2*h);
   t = t - f/derivada_f;
```

Se evalúa la nueva “f”.

```
  f = longitud_carretera(c,t0,t) - distancia;
    niter = niter + 1;
end
```

Confirmamos que en la subrutina “assert” el “t” es mayor que 0 y t0.

```
assert(t>0,'t tiene que ser positiva');
assert(t>t0,'t tiene que ser mayor que t0 en cada paso');
```

Si “t” es menor o igual que la última “t” más la tolerancia, se introducen los datos finales y se sigue buscando el próximo kilómetro, escribimos un punto con cada kilómetro encontrado.

```
if t <= tfinal + TOL

   nhitos = nhitos + 1;
   T(nhitos) = t;
   Iter(nhitos) = niter;
   fprintf('.');
end
t0 = t;
niter_hitos = niter_hitos + 1;
end

fprintf('\n');
```

Se evalúa “x” e “y” para cada “t” de cada hito.

```
        puntos = punto_carretera(c,T);

return;
```

SUBRUTINA “DIBUJA_HITOS”

Con esta subrutina se dibujarán los hitos con los datos obtenidos en la subrutina “kilometra_carretera”.

```
function dibuja_hitos(puntos)

        plot(puntos(:,1),puntos(:,2),'ro');
        hold on;

return;
```

SUBRUTINA “TABLA_KILOMETROS”

Con esta subrutina se dibuja una tabla con el número de kilómetro, “t”, “x” e “y” correspondientes a ese punto y el número de iteraciones realizadas para la obtención de cada hito.

```
function tabla_kilometros(T,puntos,Iter)

        fprintf('Kilometro      t          x          y          nº iteraciones\n');
        for i = 1:length(T)
                fprintf(' %3d      %3.4f      %3.4f      %3.4f                %2d\n',i-
1,T(i),puntos(i,1),puntos(i,2),Iter(i));
        end

return;
```

SUBRUTINA “VECTOR_COLUMNA”

Comprueba que es un vector columna y si no es así la se traspone.

```
function vcolumna = vector_columna(v)
   s = size(v);
```

Se traspone si es vector fila, el número de filas es 1. Si el número de filas es distinto de 1 y además también lo es el número de columnas estamos metiendo una matriz.

```
      if s(1) == 1
      vcolumna = v';
   elseif s(2) ~= 1
      error('es una matriz. se esperaba vector');
   else
      vcolumna = v; % ya era un vector columna
   end
return;
```

SUBRUTINA "ASSERT"

Esta subrutina se caracteriza por confirmar "a" , si esto no se cumple, se lanzará un mensaje.

```
function assert(a,mensaje)

if nargin < 2
        mensaje = sprintf(' condición %s falsa',inputname(1));
end

mensaje = sprintf('assert: %s',mensaje);

if not(a)
         fprintf('\n\n');
         fprintf(mensaje);
         fprintf('\n\n');
         error(mensaje);
end

return;
```

SUBRUTINA "LONGITUD_CARRETERA"

Esta subrutina, se caracteriza por devolver la longitud total de la carretera.

```
function L = longitud_carretera(carretera,a,b)
   if nargin < 2
      a = 1;
   end
   if nargin < 3
      b = carretera.N;
   end
```

"L"es igual a la integral entre "a" y "b" del dl por la escala, tal y como se ve a continuación.

```
   L = carretera.escala * quad(@dlong,a,b,[],[],carretera);

return;
```

SUBRUTINA "DLONG"

Esta subrutina nos devuelve dl = sqrt(dx^2 + dy^2), para ello utiliza la función derivada que nos da el "dx" y "dy".

```
function valor = dlong(arg1,arg2)
   d = derivada(arg1,arg2);
   valor = sqrt(d(:,1).^2 + d(:,2).^2);
return;
```

SUBRUTINA "DERIVADA"

Esta subrutina aplica la fórmula de la derivación por dos puntos.

```
function d = derivada(arg1,arg2)
```

Para ello se identifica si el primer o segundo argumento es la estructura carretera.

```
   if isstruct(arg1)
      carretera = arg1;
      t = arg2;
   else
      carretera = arg2;
      t = arg1;
   end
```

Una vez se ha identificado la estructura, se aplica la fórmula de la derivación.

```
   h = 0.001;
   d = punto_carretera(carretera,t+h) - punto_carretera(carretera,t-h);
   d = d / (2*h);
return;
```

SUBRUTINA "PUNTO_CARRETERA"

Evalúa los polinomios interpoladores para poder luego con los valores de "x" e "y" derivar o hacer con estos datos lo necesario.

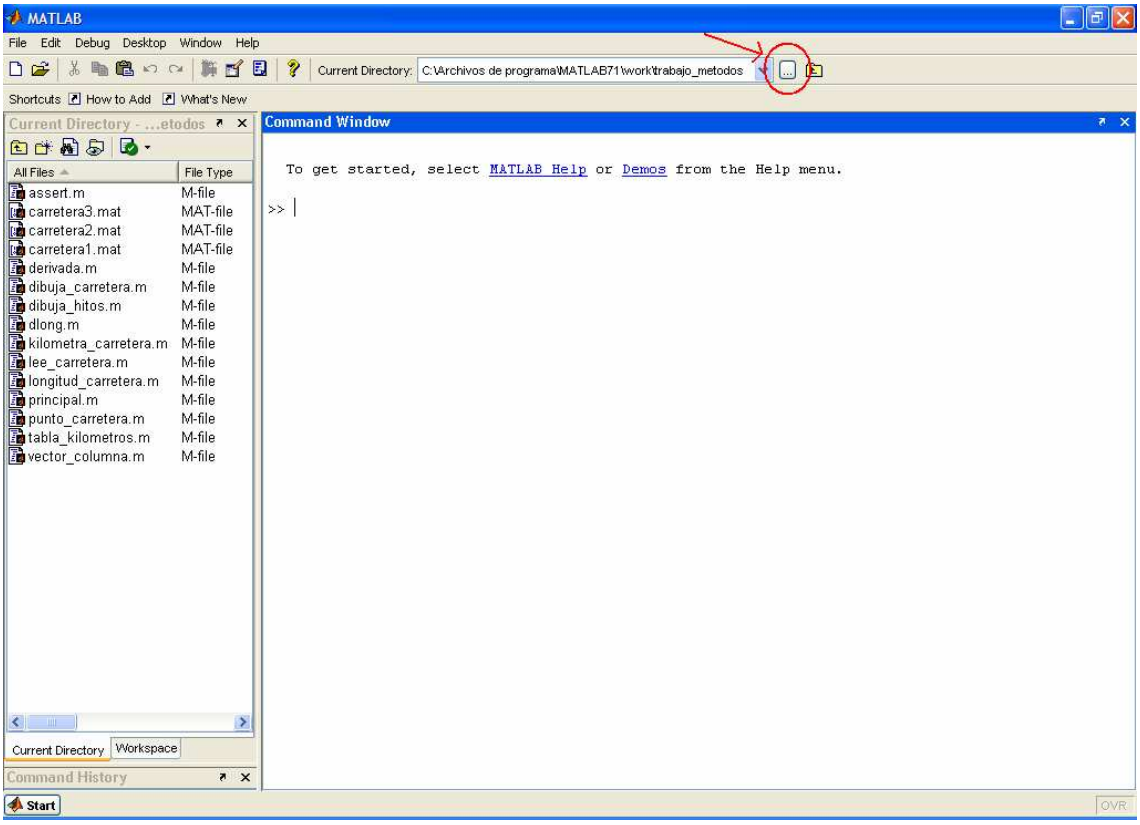
```
function p = punto_carretera(arg1,arg2)
   if isstruct(arg1)
      carretera = arg1;
      t = arg2;
   else
      carretera = arg2;
      t = arg1;
   end

   p(:,1) = ppval(carretera.splineX,t)';
   p(:,2) = ppval(carretera.splineY,t)';
return;
```

6.-APLICACIONES

6.1-EJEMPLO 1

A continuación se verán varios ejemplos siendo el primero el más ilustrativo para el fácil manejo del programa.

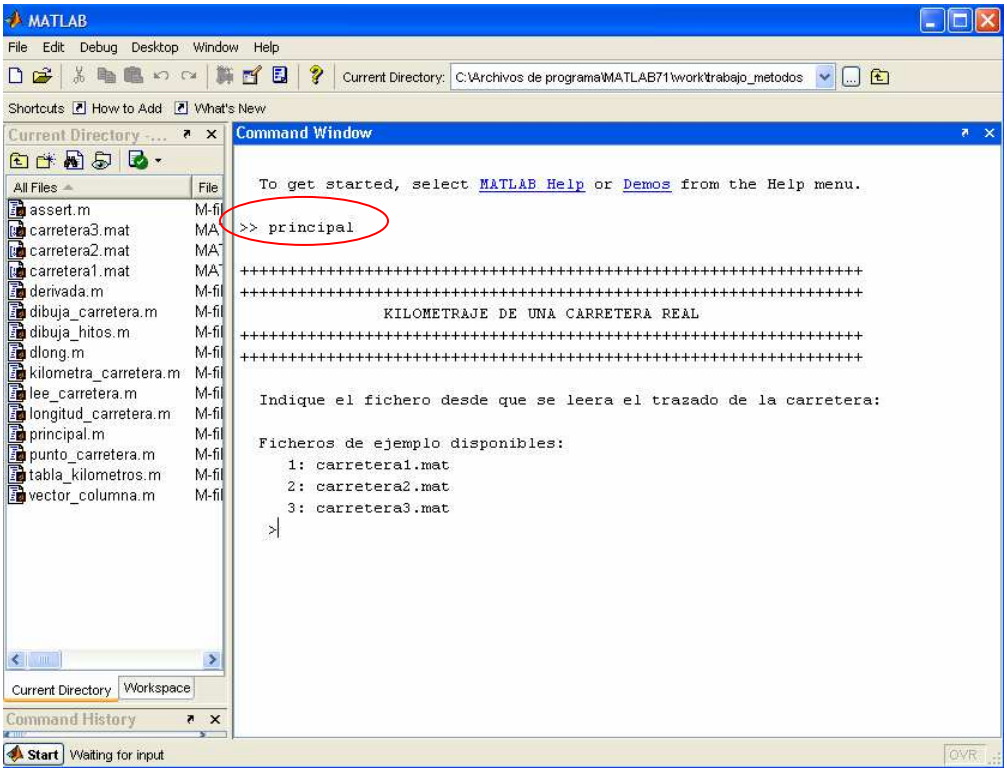
Para comenzar se procederá con la apertura del Matlab.



Ésta es la pantalla principal del programa, en donde se verá la secuencia de comandos que se estén ejecutando en ese momento.

Como se ve, en este ejemplo las subrutinas y el programa principal se encuentran a la izquierda de la ventana de comandos, dentro del *'Current directory'* facilitando el acceso a los distintos programas. En el caso muy probable de no encontrarse de esta forma, se puede acceder a ellas mediante el enlace marcado con una flecha, por el cual se pueden explorar las carpetas contenidas en el ordenador hasta encontrar su ubicación. En este caso, al encontrarse en un CD, se tendría que comenzar su búsqueda en la carpeta de 'Mi PC'.

Una vez se halla encontrado el archivo con todos los programas, en la ventana de comandos se tecleará el nombre del programa principal '*principal*', se pulsa la tecla *'enter'* y automáticamente se empezará a ejecutar el programa, tal y como se muestra en la siguiente imagen:

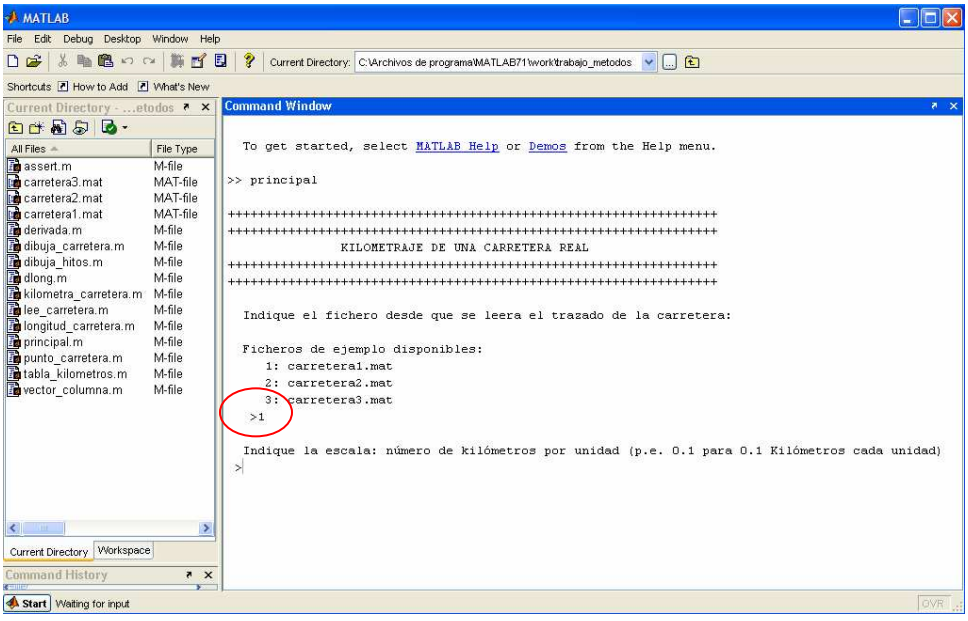


A continuación, una vez se haya abierto la ventana anterior, como se puede observar en la imagen, el programa solicita al usuario la elección de un ejemplo a resolver, tecleando *'1'* para el primer ejemplo, *'2'*, para el segundo y así sucesivamente.

En este caso se ha seleccionado el ejemplo 1.

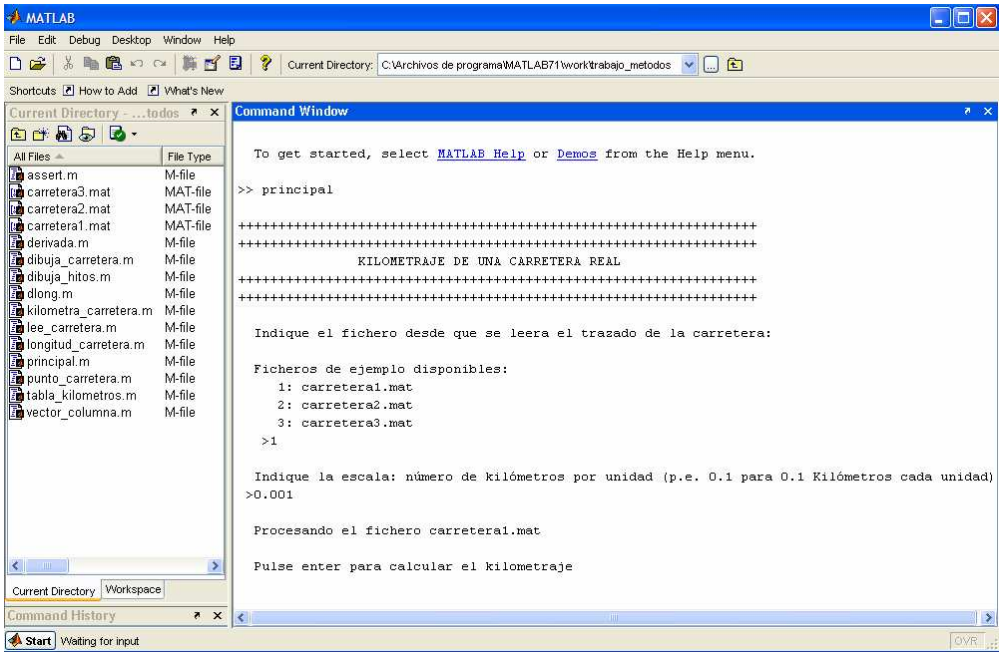
Ficheros de ejemplo disponibles:
1: carretera1.mat
2: carretera2.mat
3: carretera3.mat

Tecleamos:
>1

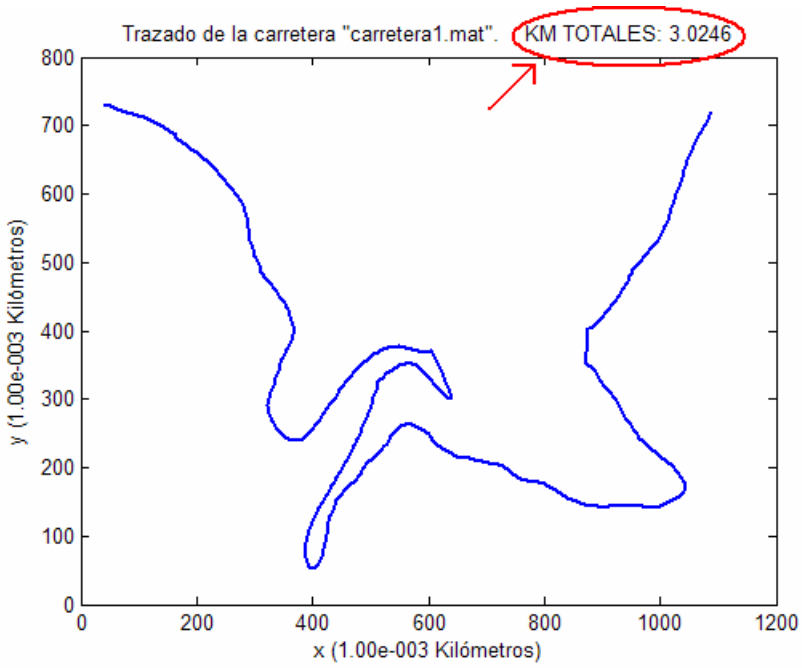


Tras introducir el ejemplo que se quiere resolver, tal y como se puede ver en la imagen superior, el programa pide la escala a la que está la carretera. Es importante tener en cuenta que el programa va a leer los datos del fichero como si estuvieran en kilómetros, así por ejemplo, como es el caso que nos ocupa, los datos están en metros, por lo que la escala a introducir mediante teclado ha de ser 0,001.

Indique la escala: número de kilómetros por unidad (p.e. 0.1 para 0.1 Kilómetros cada unidad)
>0.001



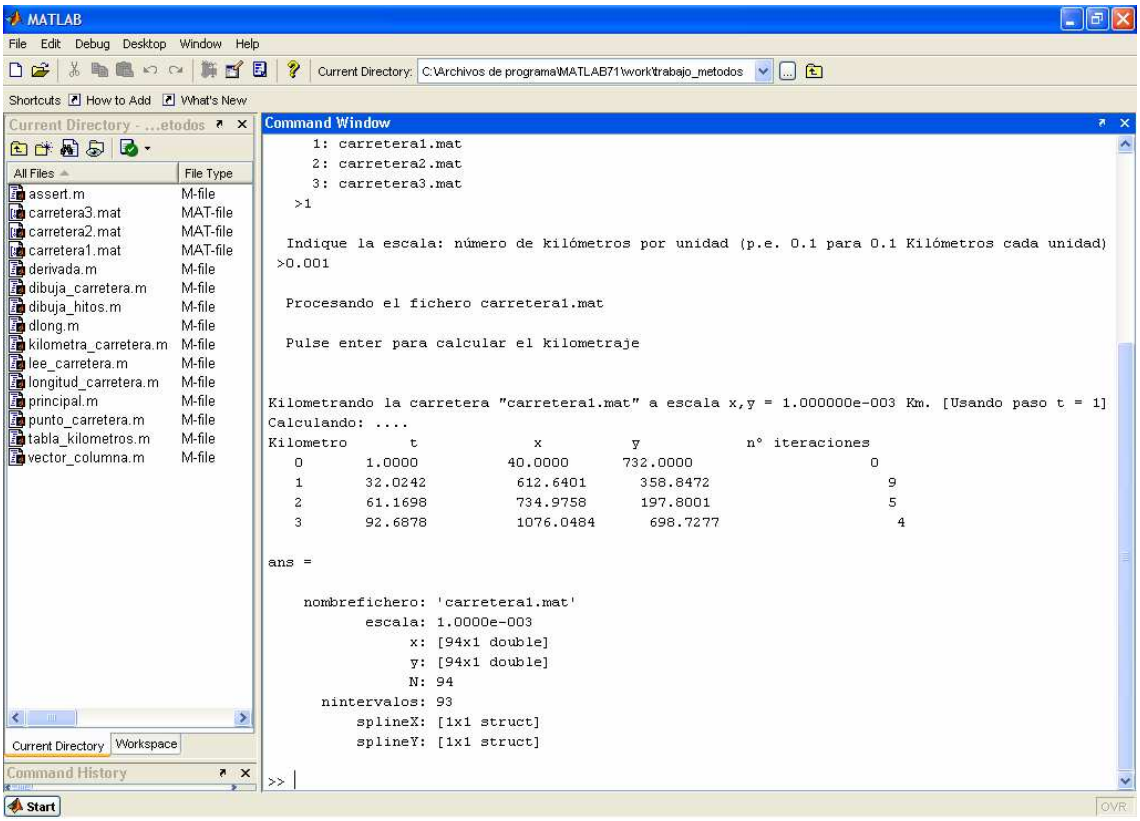
Acabado esto, el programa procesa los datos del fichero y devuelve el dibujo de la carretera con el kilometraje total, que como se puede observar en la siguiente imagen está marcado con una flecha roja.



De la figura anterior cabe destacar los ejes de coordenadas, en cuyos nombres se indica el valor de cada unidad en dicho eje.

Volviendo a la ventana de comandos, el programa solicita pulsar *'enter'* si se desea conocer el kilometraje de la carretera, es decir, marcar los puntos kilométricos, incluyendo el kilómetro cero.

Pulse enter para calcular el kilometraje



Según se pulsa 'enter' comienza el cálculo del kilometraje indicándose con 'puntos' cada kilómetro obtenido.

Kilometrando la carretera "carretera1.mat" a escala x,y = 1.000000e-003 Km. [Usando paso t = 1]
Calculando: ....

Tras el proceso de cálculo, se mostrarán por pantalla los resultados que se explicarán con detalle a continuación:

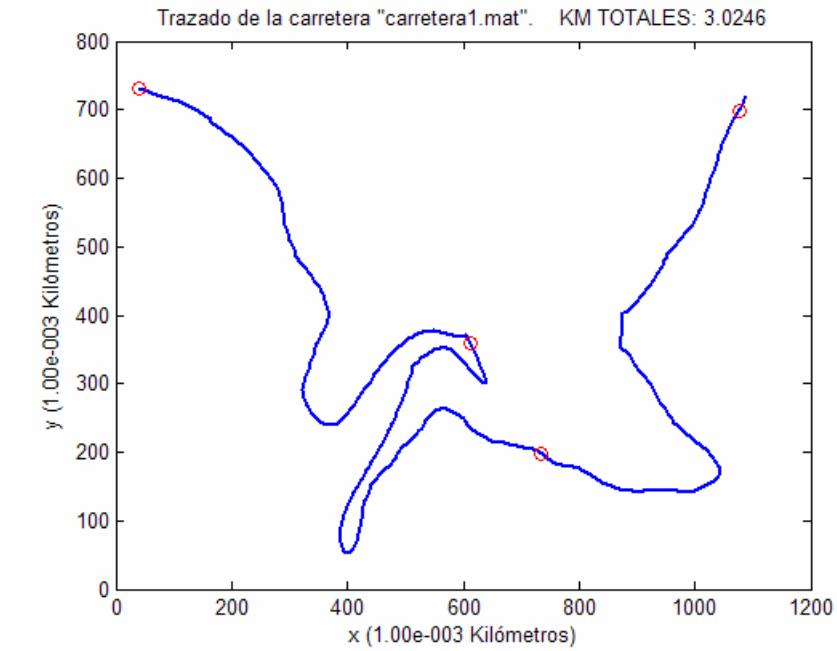
Kilómetro	*t*	*x*	*y*	*nº iteraciones*
0	*1.0000*	*40.0000*	*732.0000*	*0*
1	*32.0242*	*612.6401*	*358.8472*	*9*
2	*61.1698*	*734.9758*	*197.8001*	*5*
3	*92.6878*	*1076.0484*	*698.7277*	*4*

- La columna *kilómetro* indica el punto kilométrico.
- La segunda indica el valor del parámetro *t* para cada punto kilométrico.
- La tercera y cuarta columna indican las coordenadas *x* e *y* de cada hito kilométrico.

En lo referente a la última columna decir que muestra el número de iteraciones empleados para la obtención de cada hito kilométrico, así por ejemplo, para el cálculo del kilómetro número 2, se ha requerido de 5 iteraciones.

Además de la tabla anterior, el programa nos proporciona los datos de la carretera analizada, obtenidos a partir de la subrutina "*lee_carretera*". Dichos datos son: el nombre del fichero, la escala, las coordenadas de los puntos introducidos mediante fichero, el número de puntos totales, el número de intervalos totales, que si se observa es N-1, y finalmente, información de los polinomios interpoladores.

```
ans =

    nombrefichero: 'carretera1.mat'
          escala: 1.0000e-003
               x: [94x1 double]
               y: [94x1 double]
               N: 94
     nintervalos: 93
         splineX: [1x1 struct]
         splineY: [1x1 struct]
```



De esta imagen destacar que, sobre la obtenida en el procedimiento anterior, se han dibujado las referencias de cada kilómetro, siendo la primera la correspondiente al kilómetro cero.

6.2-EJEMPLO 2

Como la resolución de los otros ejemplos se ejecuta de la misma manera, a continuación se expondrán únicamente los resultados obtenidos, así como las gráficas finales.

La única diferencia que existe con el ejemplo resuelto anteriormente reside en la elección del ejemplo a ejecutar, que en los casos siguientes serán el 2 y el 3 respectivamente.

Indique el fichero desde que se leera el trazado de la carretera:

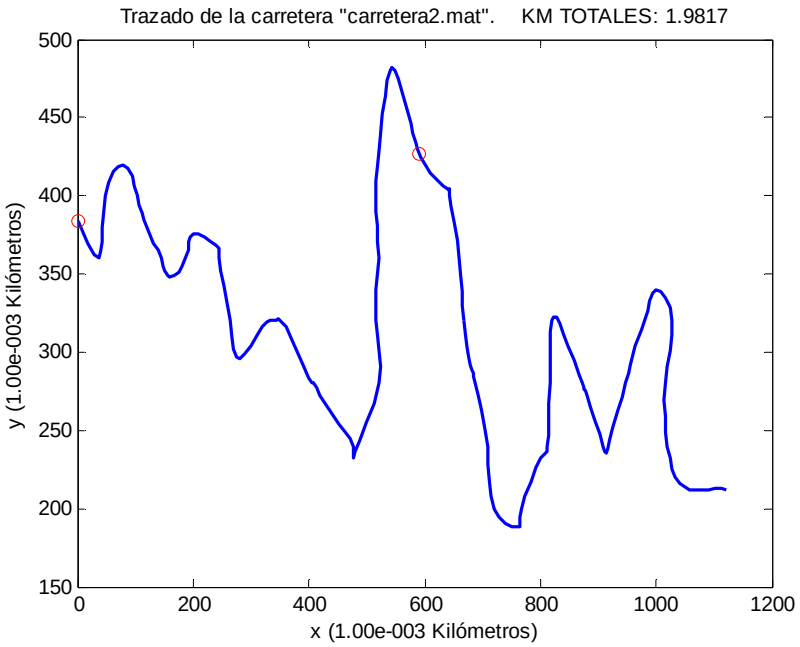
Ficheros de ejemplo disponibles:
1: carretera1.mat
2: carretera2.mat
3: carretera3.mat
>2

Indique la escala: número de kilómetros por unidad (p.e. 0.1 para 0.1 Kilómetros cada unidad)
>0.001

Procesando el fichero carretera2.mat

La tabla con las coordenadas es la mostrada a continuación:

Kilometro	*t*	*x*	*y*	*nº iteraciones*
0	*1.0000*	*0.0000*	*384.0000*	*0*
1	*30.6448*	*590.2825*	*426.7363*	*4*



6.3-EJEMPLO 3

En este tercer ejemplo se han obtenido los siguientes resultados:

Kilometro	*t*	*x*	*y*	*nº iteraciones*
0	*1.0000*	*625.0000*	*0.0000*	*0*
1	*18.7420*	*482.1708*	*416.7682*	*3*
2	*34.4401*	*1402.5019*	*446.4950*	*3*
3	*53.5648*	*1148.8700*	*794.5092*	*4*
4	*70.6393*	*388.5645*	*980.0490*	*4*