Índice

1. Introducción

Este documento se ha realizado con la intención de servir como manual de usuario para un programa en MATLAB que nos permite señalizar el kilometraje en una carretera biunívoca.

Este programa lo que nos permite es conocer en que punto de una carretera dada, siendo esta biunívoca, es decir que no retroceda a un punto anterior en el trazado de la misma, donde se encuentra cada nuevo kilómetro, pudiendo saber también el kilometraje total de la misma.

Este es un programa es de gran utilidad a efectos prácticos, ya que puede servir como sistema de apoyo en labores de tráfico, en las que se desee conocer donde se encuentra cada nuevo kilómetro de una carretera para poder establecer balizas y señalizaciones.
Además se puede extrapolar a cualquier recorrido del que se desee conocer el kilometraje, lo que puede ser útil a la hora de establecer alguna ruta turística o un nuevo sendero, etc.

Como vemos lo que nos permite conocer este programa es algo sencillo, pero que de otra forma sería laborioso de determinar.

El programa en MATLAB nos dará la solución para cada caso particular, de forma gráfica. En esta solución gráfica se podrá observar el trazado de la carretera y sobre el mismo los puntos en los que se encuentra cada kilómetro.

Para este manual, se han escogido tres ejemplos de carreteras de distintas zonas de la Isla de Gran Canaria.

Se ha optado por seleccionar carreteras de Gran Canaria, ya que es nuestro entorno más cercano, pero el programa sirve para cualquier carretera.

Las carreteras seleccionadas son las siguientes:

- Carretera C-811 que va desde Las Palmas de Gran Canaria a Mogán por el centro de la isla.

Se ha escogido el tramo de la misma que va desde la Montaña de las Vinagreras a Hoya de la Vega, rodeando la presa de Las Niñas.
En el mapa a continuación se señala la zona donde se encuentra la carretera.

Esta carretera será la que se utilice para explicar el funcionamiento del programa en MATLAB, las carretas a continuación servirán como ejemplo del correcto funcionamiento del programa.

- Carretera GC-3 que va desde la Avenida Marítima del Sur hasta Tamaraceite pasando por Jinámar, Tafira baja y Almatriche.

- La carretera GC-500 que va desde Pasito Blanco hasta Puerto Rico recorriendo toda la costa del sur de la isla y pasando por Montaña la Arena, el Llanillo, Playa de Arguineguin, Patalavaca, La Verga y Los Caideros.
-

2. Planteamiento del problema

Para poder crear un programa en MATLAB que nos permita realizar el kilometraje de una carretera, lo que va a realizar dicho programa es lo siguiente:

- Obtener un polinomio que aproxime la carretera a estudio.
- Calcular los puntos que marcan un nuevo kilómetro en la carretera.
- Mostrar una gráfica con el trazado del polinomio y el kilometraje.

Para ello los datos que se introducirán para poder ejecutar el programa serán puntos en un fichero de datos que nos describan el trazado de la carretera, a partir de estos puntos se podrá realizar el polinomio que nos aproxime dicha carretera.

Estos puntos los obtendremos de la siguiente manera:

Cogemos un mapa con la carretera a la cual queremos obtener su kilometraje. Ponemos un papel vegetal milimetrado encima y la calcamos. Una vez tengamos la carretera calcada ponemos unos ejes de coordenadas X-Y y, teniendo en cuenta la escala del mapa hacemos una tabla donde aparecerá para cada punto en el eje X su correspondiente en el eje Y. Para que la carretera sea lo más precisa posible habrá que situar muchos puntos en el eje X (por ejemplo cada 100 metros). A continuación adjuntamos los ejemplos de carreteras usados para la realización de este trabajo.

Una vez obtenidos los puntos, ya pueden ser introducidos en el fichero de datos para la realización de su kilometraje.

3. Fundamentos teóricos

Para realizar el programa en MATLAB que nos de el kilometraje de una carretera, nos hemos apoyado en los métodos numéricos que hemos considerado más apropiados para resolver los problemas matemáticos que implica esta tarea.

A continuación se exponen de forma teórica los métodos utilizados.

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 entonces 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:

$$X \quad x_0 \; x_1 \; x_2 \ldots\ldots \; x_n$$

$$Y \quad y_0 \; y_1 \; y_2 \ldots\ldots \; y_n$$

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

- 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 las 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 hasta de 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 $[t_0, t_n], [t_1, t_2], \ldots, [t_{n-1}, t_n]$, S está definido por un polinomio cúbico diferente. Sea S_i el polinomio cúbico que representa a 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:

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

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 del 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 $s_i''(t_i) = z_i$, a su vez, $s_i''(t_{i+1}) = z_{i+1}$.

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

La expresión resultante es:

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

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

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

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

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:

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

En donde:

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

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

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

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

Lo que vamos a hacer con el programa MATLAB, es utilizar un algoritmo que nos encuentre los coeficientes *z* del spline cúbico natural.

3.2. Método de derivación numérica

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

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

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

Para la realización del programa en MATLAB que nos de el kilometraje de una carretera, 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*, así obtenemos la derivada de la siguiente forma:

c-h , c+h $\qquad f'(c) = \frac{f(c+h) - f(c)}{h}$

Siendo el error cometido en el cálculo el siguiente:

$$R(f) = -\frac{h}{2} f''(\zeta)$$

Cuanto mayor sea el número de puntos seleccionado, mayor será el error.

3.3. Integración numérica

En el programa en MATLAB, las integrales se resuelven mediante un método de integración numérica, en concreto, mediante una fórmula de cuadratura de Gauss con función de peso.

Estos métodos funcionan de la siguiente manera. Suponemos primero una integral de la siguiente forma:

$$\int_a^b w(x) f(x) dx$$

Donde *w(x)* es la función de peso, la cual, es continua y estrictamente positiva en el intervalo *(a,b)* donde se realiza la integral.

Hay diferentes tipos de fórmulas de Gauss, en este caso se va a utilizar la de Gauss-Legendre, donde la función de peso es la unidad, así integramos la función que nos interesa *f(x).*

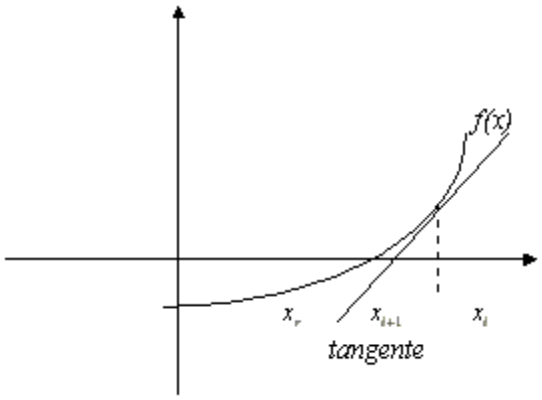
$$w(x) = 1 \Rightarrow \int_{-1}^{1} f(x)dx$$

Como vemos, este método, funciona para la resolución de integrales en el intervalo *(-1,1).*

3.4. Fórmula de Newton-Raphson

Este método es un método iterativo de los más usados y efectivos.

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



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 $m=f'(x_i)$.

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

$$y-f(x_i)=f'(x_i)(x-x_i)$$

Si en la fórmula anterior hacemos $y=0$, y despejamos x, obtenemos la fórmula iterativa de Newton-Raphson.

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

Esta fórmula iterativa nos permite calcular la siguiente aproximación:

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

Note 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 observe 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

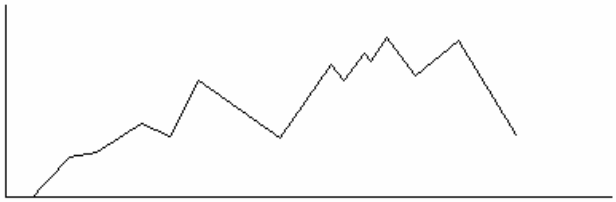
Una vez definidas las funciones a utilizar, se va a exponer el planteamiento del problema en MTLAB, con las características particulares del programa del kilometraje.

4.1. Definición de la función que representa a la carretera

En un primer paso lo que se necesita, es definir un polinomio que nos aproxime la carretera. Como ya se ha indicado en apartados anteriores, se usará para ello un polinomio de interpolación.

Aunque se recomendó el uso de un método de diferencias divididas como es el de Interpolación de Newton, se ha optado por utilizar un método de interpolación polinomial a trozos, como es el de la spline cúbica.

Esto se debe a que como se quería ajustar lo máximo posible la función a la realidad de la carretera, se tomaron muchos puntos para definir la carretera. Si no hubiese sido así nos hubiese quedado un resultado lejano a la realidad como el que se muestra a continuación.



En comprobaciones anteriores, se observó como el polinomio de Newton oscilaba para tantos datos de partida, es decir no convergía en ningún resultado, y en el mejor de los casos no daba un resultado ajustado. Por ello se consideró el método de la spline cúbica natural.

Por otro lado, la función spline utilizada es la que viene definida en la librería del programa MATLAB, en combinación con otra función también de esta librería.

Primero se ejecutará la función PP = SPLINE(X,Y). Esta función nos devuelve la forma polinomial a trozos del spline cúbico interpolador para los puntos (x, y) introducidos que nos definen la carretera. Para evaluarlo,hay que aplicarle la función PPVAL(PP,XX), también de la librería del MATLAB. Esta función PPVAL se ejecutará más adelante, en otro punto del programa en el que nos interesa dibujar la función que corresponde a nuestra carretera.

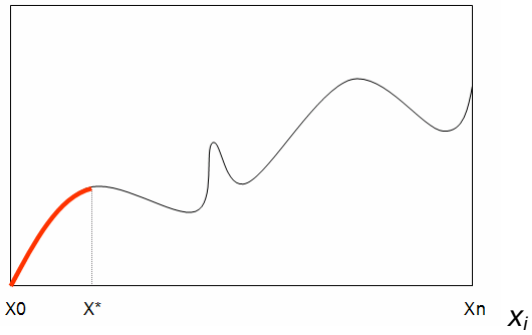
Podemos resumir entonces, que para empezar el programa necesitamos introducir en un fichero los puntos que representan a nuestra carretera, se realizará en dos columnas, una para los valores de x y otra para los valores de y.

Una vez hecho esto, con la función spline, conseguiremos construir el polinomio que represente a nuestra carretera.

4.2. Cálculo del kilometraje

Este primer paso, es realmente un paso previo necesario para que el programa nos de el resultado esperado. Este resultado es el kilometraje de la carretera.

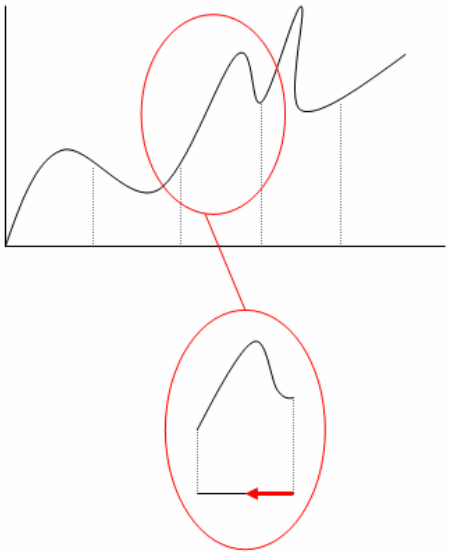
Para conseguir esto, estudiaremos que puntos del polinomio obtenido, corresponden a un nuevo kilómetro.



En la gráfica anterior, podemos ver una representación de un polinomio, y en rojo, está marcado el primer kilómetro del mismo desde su origen en x0.

Para la resolución de este problema utilizaremos un método iterativo como es el de Newton-Raphson, que nos indique donde se encuentra un kilómetro, y a partir de este el siguiente, hasta alcanzar el final de la carretera.

Necesitamos ir recorriendo la función que describe el polinomio por intervalos hasta encontrar un nuevo kilómetro. Dentro de cada uno de esos intervalos se irá estrechando el cerco desde el límite final del intervalo hacia el inicial, hasta encontrar el punto en el que este ese kilómetro.



La longitud de una curva viene definida por la siguiente expresión:

$$\int_a^b \sqrt{1+(P'(x)^2}\,dx$$

Si nos fijamos en el gráfico del polinomio, vemos como los límites de integración en nuestro caso son x_i y x^*, es decir:

a= para i entre 0 y n
*b= x**

Esto quiere decir, que cuando:

$$\int_{x_i}^{x^*} \sqrt{1+(P'(x)^2}\,dx = 1$$

La longitud del tramo de curva entre un punto x_i y otro punto x^* será de un kilómetro, siempre y cuando la escala utilizada en la descripción corresponda a un kilómetro, si no es así, habría que ajustar el resultado a su escala.

Como ya se ha visto, la fórmula de Newton-Raphson es:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Esta fórmula se puede adaptar a nuestro problema considerando que la función *f(x)* es la siguiente:

$$\int_{x_i}^{x^*} \sqrt{1+(P'(x)^2}\,dx - 1 = 0$$

De esta forma nos queda la siguiente expresión:

$$x_{i+1}^{*} = x_i^{*} - \frac{f(x_i^{*})}{f'(x_i^{*})} = x_i^{*} - \frac{\int_{x_i}^{x^{*}} \sqrt{1+P'(x)^2}\,dx - 1}{\frac{d}{dx}\int_{x_i}^{x^{*}} \sqrt{1+P'(x)^2}\,dx - 1}$$

Para conseguir nuestro objetivo lo que haremos será crear un bucle que siga poniendo en funcionamiento la fórmula anterior, que nos da la situación en *x* de cada nuevo kilómetro, y se irán modificando los índices a la misma cada vez que se encuentre un kilómetro nuevo.

Como se puede observar, dentro de esta función, necesitamos resolver, una integral y una derivada. Para resolver la primera se aplicará el método de integración numérica Gauss- Legendre, ya definido, y para el segundo el de derivación numérica.

4.3. Representación gráfica

Por último, para que el programa nos de una representación gráfica de las soluciones a cada problema particular, si que evaluaremos el resultado obtenido por la función spline, para así poder dibujar la carretera. También se representarán los puntos que marcan el kilometraje sobre la misma, así como se representará el kilometraje total de la misma.

5. Programa

"Cálculo del Kilometraje de una Carretera Real"

Este programa realiza el kilometraje de una carretera real cuyas medidas han sido pasadas en un fichero de dos columnas tal que en la primera se muestran las medidas del eje de abcisas (X) y en la segunda las medidas del eje de ordenadas (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, a partir del cual y usando la función:

- *ppval(PP,XX)*

Podremos evaluar dichos splines en los puntos indicados en XX.

5.2. Subrutinas empleadas

- *longCurva* : A su vez engloba a las siguientes:
    - *Derivación* DerivacionNumerica
    - *Integración* GaussLegendre

5.3. Variables

way Matriz que en la que se cargaran las medidas desde el fichero
N Número de medidas tomadas. Las obtendremos con 'size'
ESCALA Escala de las medidas tomadas
pp Polinomio especial que contendrá los coeficientes de interpolación de los splines
cúbicos obtenidos del trazado de la carretera con la función spline

km Puntos de kilometraje encontrados
x0,x1 Limites de integración que estamos buscando para determinar cada km
xi Aproximación inicial para cada intervalo xi = x0 + 1
u x1* = x1 – u , siendo i = integral(sqrt(1+p'^2));
u = (i - 1*(unKM/ESCALA))/ derivada (i);
uAnt Nos permitirá determinar si se están produciendo oscilaciones, comparando la
variación actual con la anterior.
error Error permitido a la hora de determinar un KM

5.4. Secuencia del programa

A continuación se va a comenzar el programa, definiendo las variables e introduciendo el nombre del programa, también se va a solicitar que se introduzcan los datos al usuario del programa, es decir que se indique la dirección del fichero con los datos de entrada y la escala en que están dichos datos.

```
clear km pp x0 x1 yy u uAnt

syms('NOMBRE', 'N', 'way', 'ESCALA', 'pp', 'x0','x1','xi','u','uAnt', 'error');
syms('L','I','J','v','y','yy', 'kk', 'unKM');

unKM = 100000; (porque 1km = 100.000 cm)

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');
NOMBRE = input(' ','s');
fprintf(1,'\n  Indique la escala de las medidas realizadas(cm) (ej: 1cm / 0.5 km =
1/50000 :  \n');
ESCALA = input(' ');
```

Ahora cargamos los datos del trazado desde el fichero indicado por el usuario.

```
way = load(NOMBRE, '-ascii');
N = size(way);
```

La primera columna de way será => way(:,1) y en ella encontraremos los valores de X(I).
La segunda columna de way será => way(:,2) y en ella encontraremos los valores de F(X(I)).

A continuación obtenemos los splines que interpolan al trazado en la variable pp.

```
        spline( X(I)      , F(X(I)) )
pp = spline( way(:,1), way(:,2));
```

Ahora debemos comenzar a buscar los puntos de kilometraje de la carretera.

cm/km = 100000 / ESCALA = numero de centímetros por kilómetro (en línea recta)

Para ello iremos recorriendo la función que describe la carretera por intervalos, por ello se requiere un bucle que vaya realizando iteraciones a lo largo de dicha función. Nuestro incremento bruto entre intervalos (u) va a ser de 50m en la escala real.
El incremento fino lo situaremos en 1m.

También definiremos un margen de error para este proceso. Para una escala de 1km el error será de 1m de error, y para otra escala será un error equivalente.

```
error = 100*ESCALA;
```

Definiremos también el primer punto para comenzar esta iteración, el cual será el principio de carretera.

```
x0 = way(1,1);
```

Y cargaremos el vector de salida, con los resultados, con este valor inicial para la iteración.

```
km(1) = x0;
```

Situamos el otro extremo de nuestra aproximación bien lejos (1km en línea recta, es decir en x) y luego iremos estrechando el cerco, es decir, caminando hacia atrás en el eje de abcisas hasta dar con el kilómetro.

```
xi = x0 + 1*(unKM*ESCALA);
x1 = x0;
I = 2;

while x1 <= way( N(1), 1 )
```

La expresión de la derivada con respecto a X de la longitud de la curva es:

```
   L = longCurva(pp,x0,xi);
      d = L /(xi - x0);
   u = (L - 1*(unKM*ESCALA))/d;

   errorL = abs(L - 1*(unKM*ESCALA));
```

Cuando alcancemos el error o el final de la carretera paramos, si no es así seguimos como se indica a continuación, modificando los índices para hacer nuevas iteraciones en el siguiente intervalo.

```
 while (errorL > error)
         x1= xi - u;
         xi = x1;
```

Actualizamos también el valor de u antes de volver a ejecutarlo.

```
         L = longCurva(pp,x0,xi);
```

Lo que se ha hecho es llamar a la subrutina longcurva. Esta subrutina calcula la longitud de la curva encerrada bajo los splines interpoladores de unos puntos dados, entre el intervalo [A,B]

Para ello mezclaremos los métodos de Gauss - Legendre y Derivación Numérica

```
function L = longCurva(pp, A ,B)

 syms('DXP','N','H','XI0','XI','WI','NN','I','X','JX','s','x','K1','K2');

 N = 5;
```

Inicializamos los coeficientes de los polinomios de Legendre hasta grado5

```
 XI = zeros(4,5);
 WI = zeros(4,5);

   XI(1,1) = 0.5773502692;
   XI(1,2) = -XI(1,1);
   WI(1,1) = 1.0;
   WI(1,2) = 1.0;
   XI(2,1) = 0.7745966692;
   XI(2,2) = 0.0;
   XI(2,3) = -XI(2,1);
   WI(2,1) = 0.5555555556;
   WI(2,2) = 0.8888888889;
   WI(2,3) = WI(2,1);
   XI(3,1) = 0.8611363116;
   XI(3,2) = 0.3399810436;
   XI(3,3) = -XI(3,2);
   XI(3,4) = -XI(3,1);
   WI(3,1) = 0.3478548451;
   WI(3,2) = 0.6521451549;
   WI(3,3) = WI(3,2);
   WI(3,4) = WI(3,1);
   XI(4,1) = 0.9061798459;
   XI(4,2) = 0.5384693101;
   XI(4,3) = 0.0;
   XI(4,4) = -XI(4,2);
   XI(4,5) = -XI(4,1);
   WI(4,1) = 0.2369268850;
   WI(4,2) = 0.4786286705;
   WI(4,3) = 0.5688888889;
```

```
   WI(4,4) = WI(4,2);
   WI(4,5) = WI(4,1);
```

Primer paso:

```
 K1 = (B-A)/2;
 K2 = (B+A)/2;
 JX=0;
```

Variación para el cálculo de la derivada

```
 h = 0.01;
```

Paso dos

```
 for I = 1:N
```

Paso tres

```
   X = K1*XI(N-1,I)+K2;
```

Paso cuatro. Calculamos la derivada primera usando derivación numérica

```
   DXP= (ppval(pp,X+h) - ppval(pp,X)) / (h);
```

Paso cinco

```
   Q = sqrt(1 + DXP^2);
   JX = JX + WI(N-1,I)*Q;

 end
 JX=JX*K1;
```

La integral de F desde A hasta B es JX

```
 L= JX;
```

```
      d = L /(xi - x0);
    uAnt = u;
    u = (L - 1*(unKM*ESCALA))/d;
```

Se ha comprobado que el programa podía oscilar en algunas partes de la función, sobretodo en aquellas donde la función tiene mucha pendiente. Para evitar oscilaciones en el caso de grandes pendientes, lo q hacemos es reducir los incrementos en los que se va acortando el intervalo hasta alcanzar el kilómetro.

```
        if(abs(u) >= abs(uAnt))
            u = sign(u)*(abs(u) - abs(uAnt))/3;

        end

        errorL = abs(L - 1*(unKM*ESCALA));
    end
```

Por si la primera aproximación ha sido correcta:

```
    x1 = xi;
```

Para cuando finalicemos y nos hayamos salido del trazado:

```
    if (x1 <= way(N(1),1))
        km(l) = x1;

        x0 = x1;
        xi = x0 + 1*(unKM*ESCALA);
        l = l + 1;
    end


end
```

Ahora dibujaremos la curva y sus puntos de kilometraje

```
 yy = zeros(1,101*(N(1)-1));
 v = zeros(1,101*(N(1)-1));

 for J = 1:N(1)-1
 for I = 1:101
    v(101*(J-1)+I)=way(J,1) + (I-1)*(way(J+1,1)-way(J,1))/100;
 end
 end
```

Como se había comentado hay que llevar a cabo una valuación de la función para dibujar el trazado

```
yy = ppval(pp,v);
```

Obtenemos la evaluación de los puntos kilométricos

```
kk = ppval(pp,km);
```

Calculamos los kilómetros totales del trazado

```
kmTotal = size(km);
kmTotal = kmTotal(2) - 1;
kmTotal = kmTotal + longCurva(pp,km(kmTotal + 1), way(N(1),1));
```

Creamos la grafica e introducimos los datos obtenidos

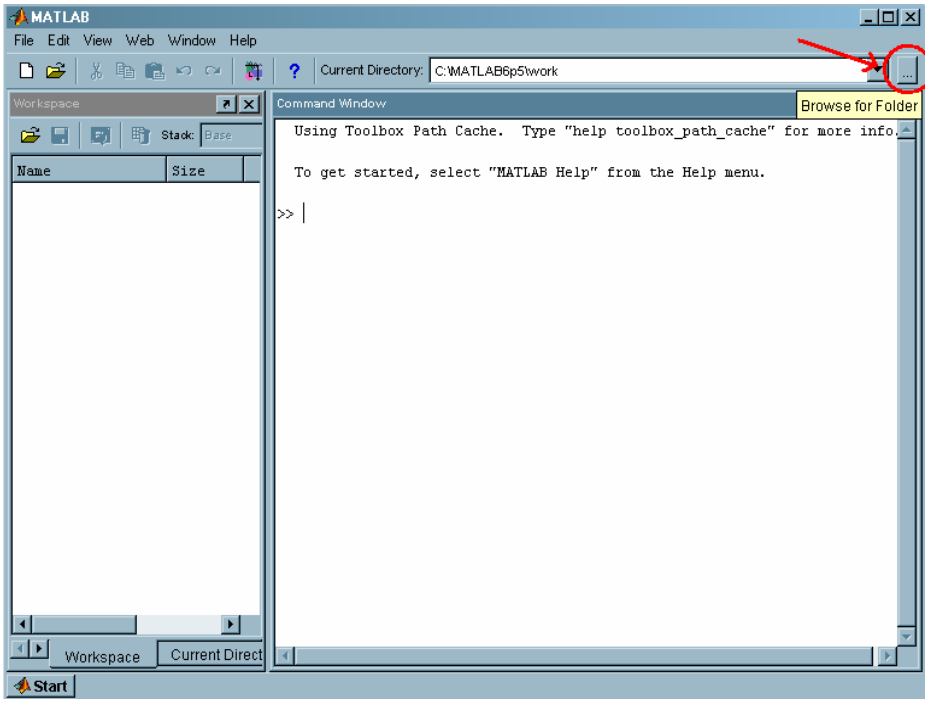
```
figure(1);
plot(v(:),yy(:),'-r',km(:), kk(:), 'b+')
title(sprintf('Trazado de la carretera. KM TOTALES: %3.4f',kmTotal));
xlabel(sprintf('Km. ( Escala : 1 CM : %i KM  )',1/(ESCALA*unKM)));
ylabel('Km');

fprintf(1,'\n\n Los puntos kilométricos los podemos localizar en las siguientes
coordenadas de abcisas\n\n');
for I= 1:ceil(kmTotal)
   fprintf(1,' KM %3i  ->  X = %f \n', I - 1, km(I));
end
   fprintf(1,'\n     Fin de Trazado en KM = %3.4f\n', kmTotal);
```

6. Aplicaciones

Una vez que hemos abierto el programa MATLAB podemos utilizar el programa. Probablemente en su escritorio encontrará un icono similar a este, pulse sobre él y abrirá el programa MATLAB.

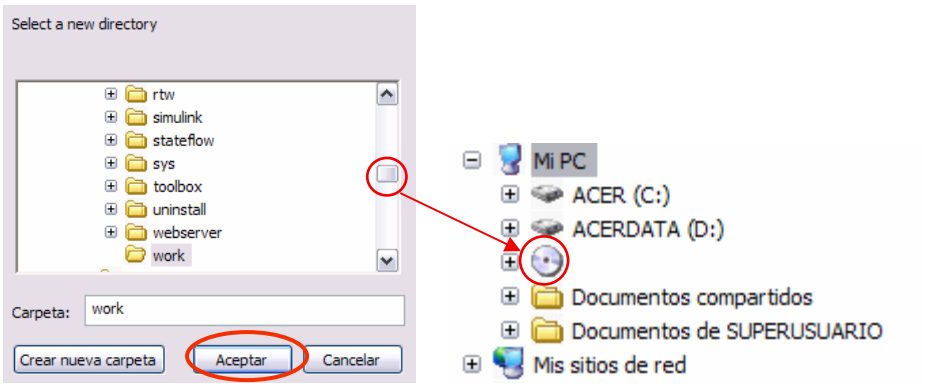
La primera pantalla que aparece al abrir el MATLAB es la siguiente:



Debemos ir a donde indica la flecha para fijar la dirección donde se encuentra nuestro programa para el cálculo del kilometraje de una carretera real llamada “kilometraje”.

En este momento aparecerá una ventana como la siguiente en la que se seleccionará la dirección donde se encuentra nuestro programa kilometraje, y pulsaremos en aceptar.

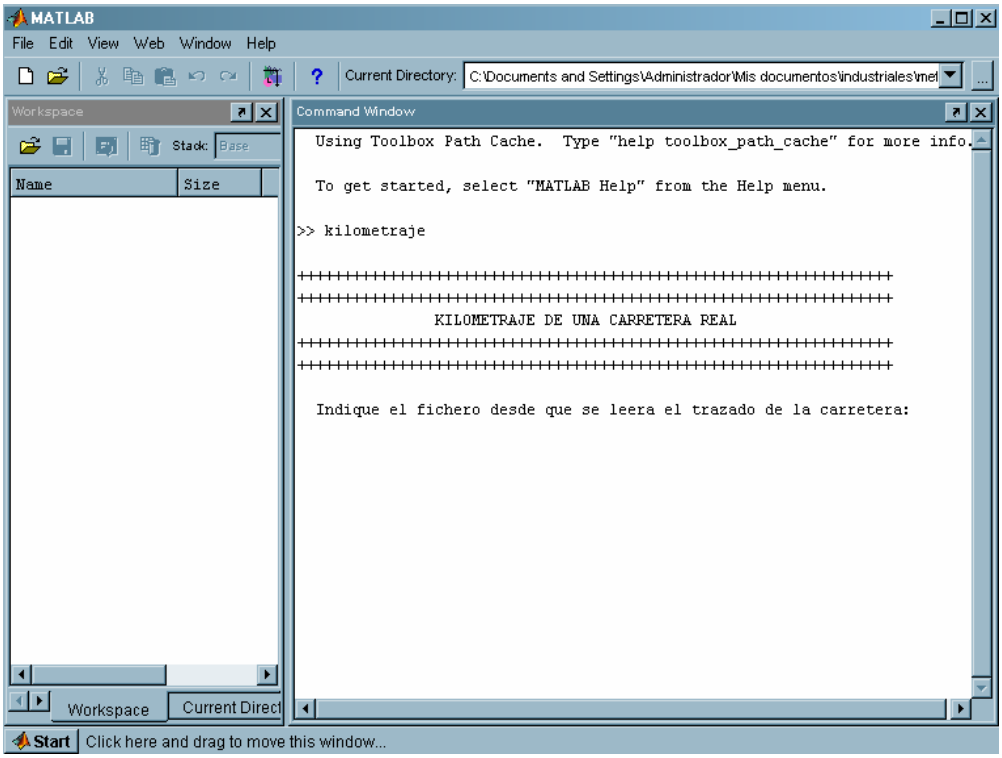
Para encontrar el programa moveremos el cursor de la derecha y localizaremos la carpeta en que se encuentra. Probablemente si lo está ejecutando desde el CD que viene adjunto a este documento, tendrá que localizar dicho icono, dentro de MiPC.



Una vez que haya pulsado aceptar, volverá a l pantalla inicial. Ahora ya se puede empezar a ejecutar el programa. Para ello debe escribir en la pantalla el nombre del mismo.

>> Kilometraje

Tras pulsar la tecla ‘’enter’’, le aparecerá lo que se muestra a continuación en la pantalla.

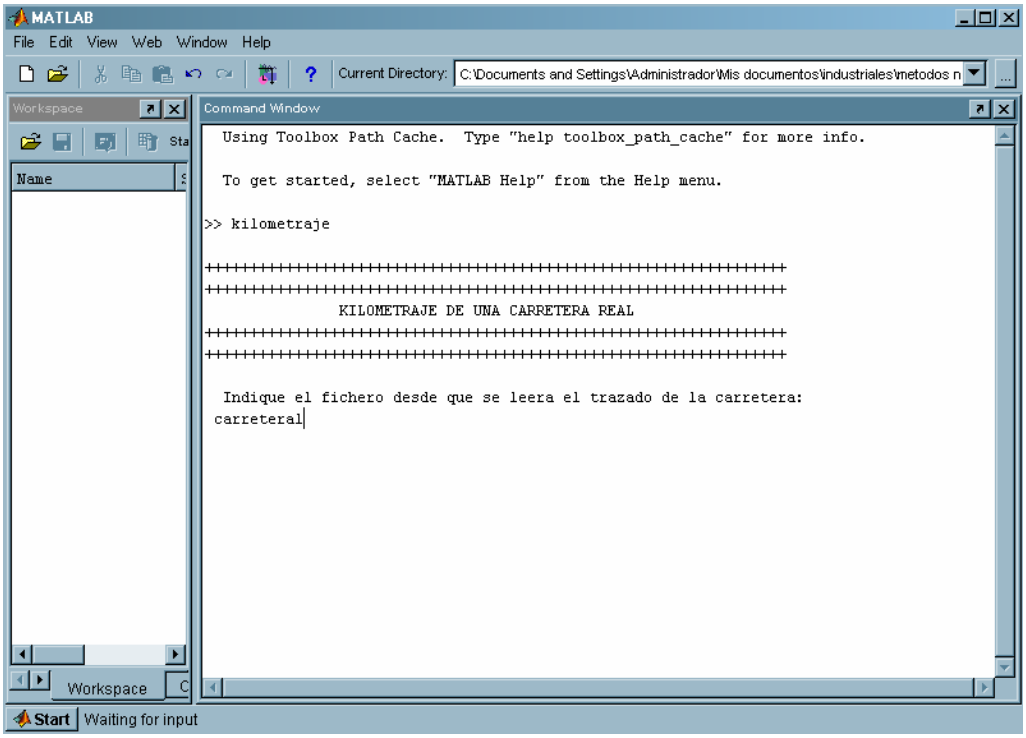


Una vez tengamos el algoritmo en funcionamiento nos pedirá que introduzcamos el fichero de entrada que contenga los puntos que definen la carretera que el usuario quiere obtener el kilometraje. En nuestro caso se llama carretera1.

Introduciremos:

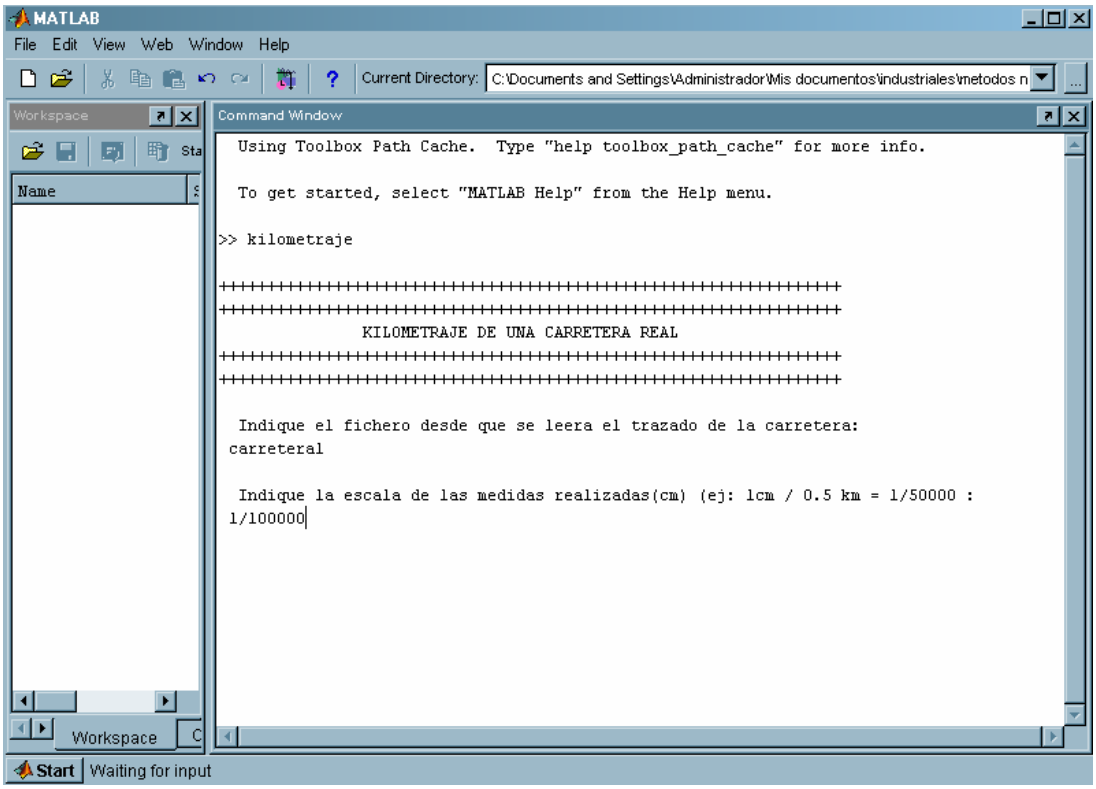
Carretera1

Tras haber escrito esto pulsaremos el ''enter.''



Una vez hecho esto, lo último que nos pedirá el programa será introducir la escala de la carretera. El ejemplo que pone el programa es que un centímetro del dibujo, es medio kilómetro, la escala a introducir es 1/50.000.
En nuestro caso es 1 / 100000, ya que la escala de los datos del fichero están a escala 1 cm.: 1 Km, ya que el mapa del que se tomaron los datos estaba a dicha escala.

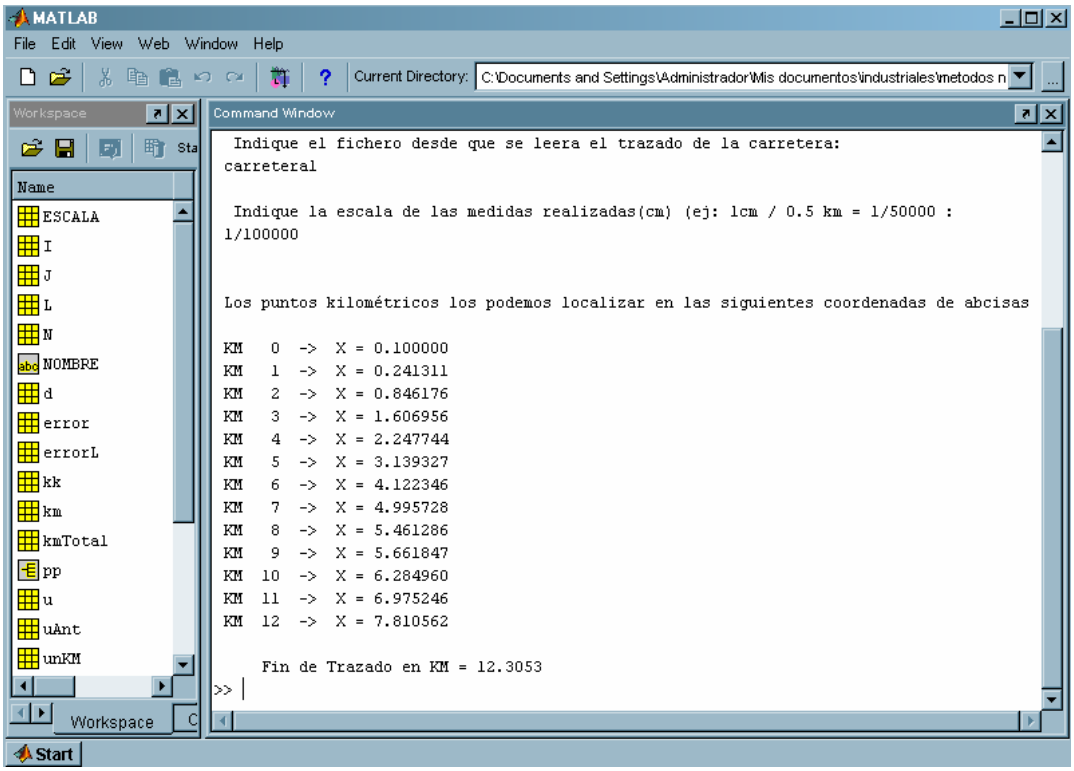
Tras introducir el dato, se pulse de nuevo la tecla ''enter.''



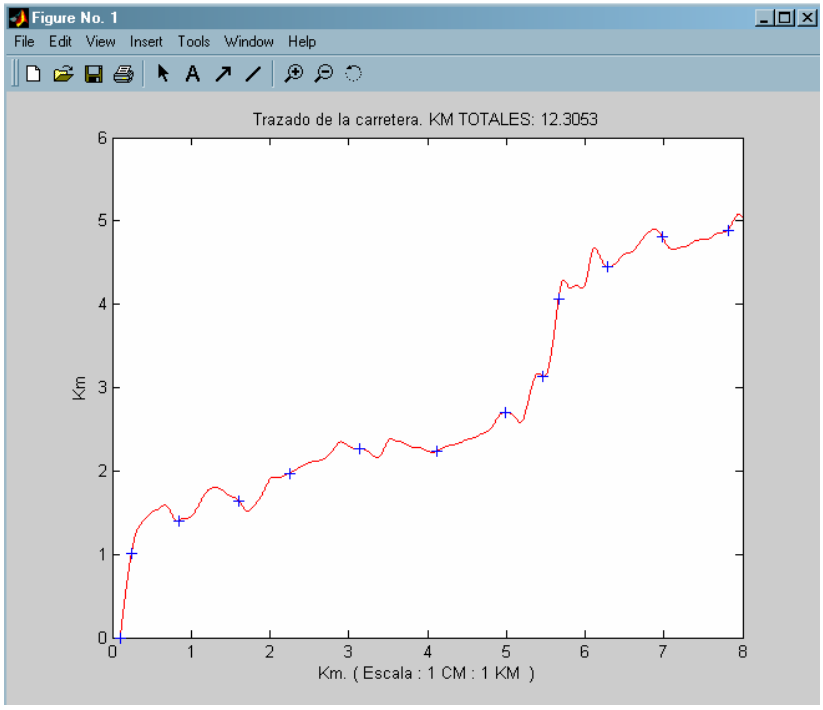
A partir de aquí el programa trabajará por si solo. Si ve que tarda demasiado, o que da error, puede que haya errores en los datos del fichero de entrada, o que algún paso anterior no se ha ejecutado bien.

En unos segundos, tras el último paso, nos aparecerá el resultado de dos maneras diferentes:

1. Nos saldrá en pantalla donde se localiza cada kilómetro en el eje de abcisas y el dato del kilometraje total de la carretera.

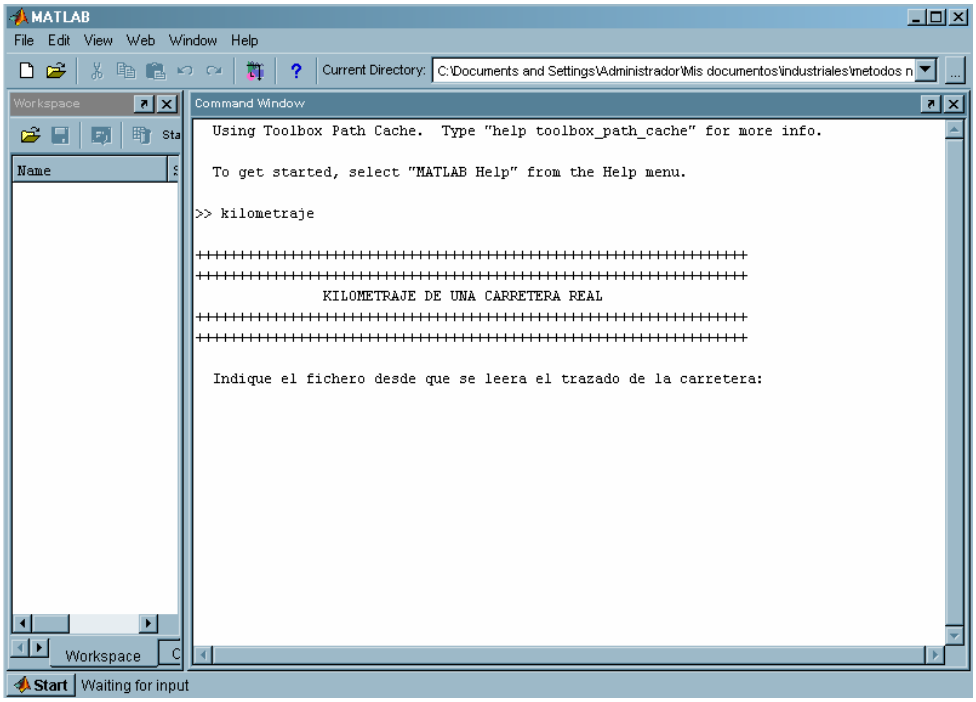


2. Nos aparecerá una gráfica con el trazado de la carretera, con todos sus kilómetros señalizados con cruces de color azul, el número de kilómetros totales y la escala original de los datos introducidos.



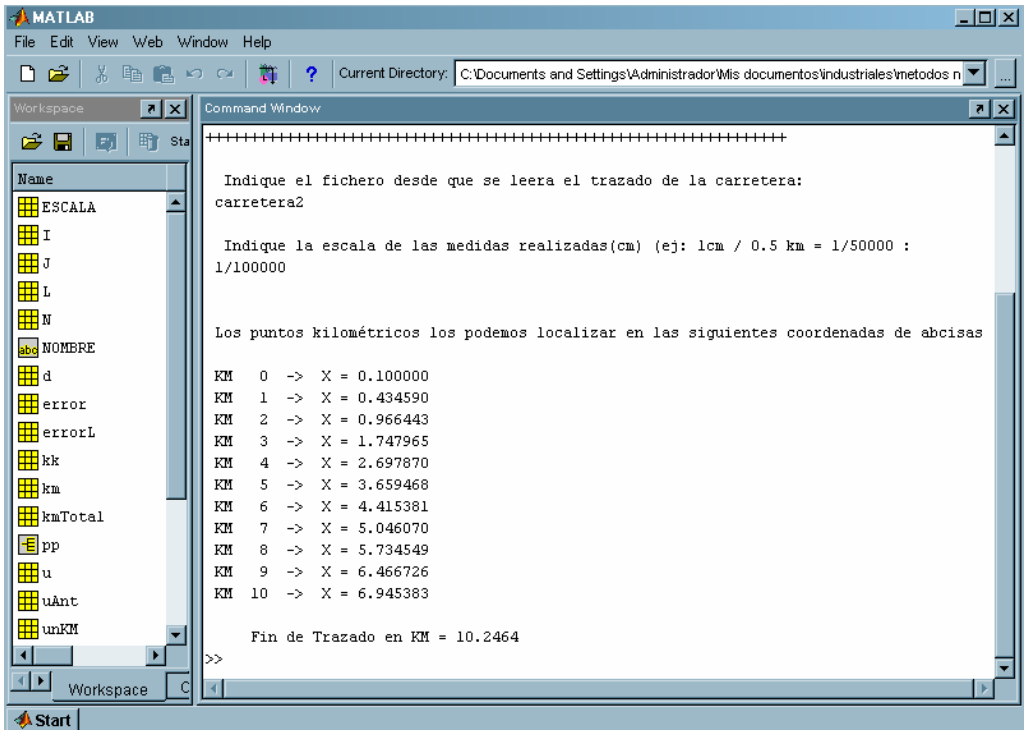
6.1. Ejemplo1

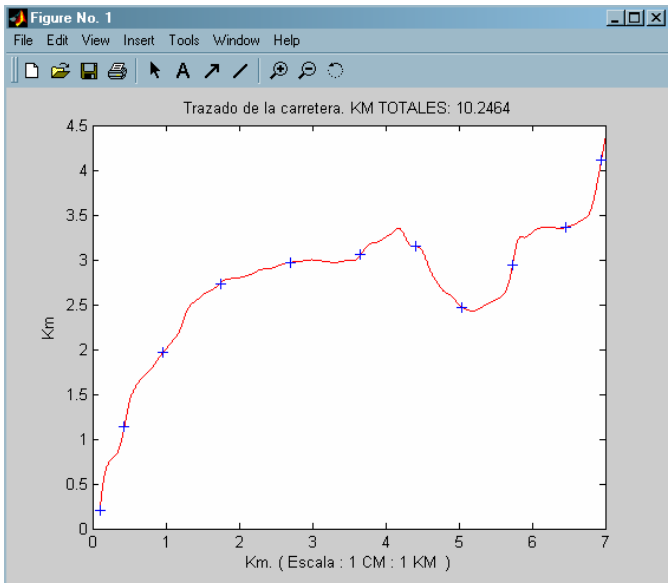
Vamos a repetir el procedimiento anterior con otras carreteras.



Ahora introduciremos el nombre del fichero que contiene los puntos de la segunda carretera llamado “carretera2”. La escala en este caso es igual que en el ejemplo anterior 1/100000.

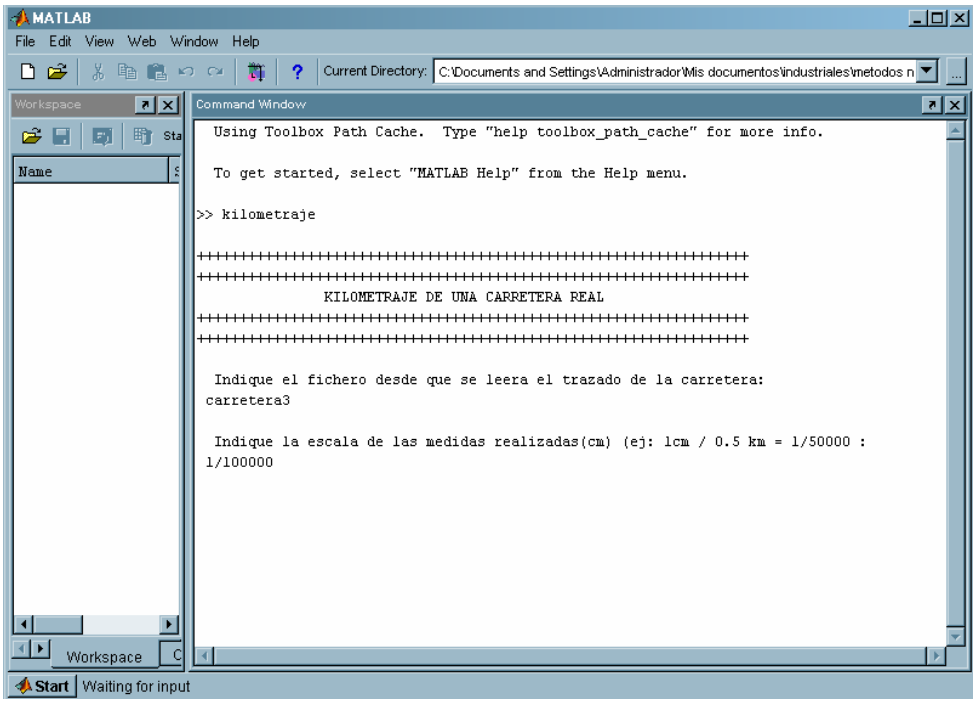
Le aparecerán en este caso los siguientes resultados.



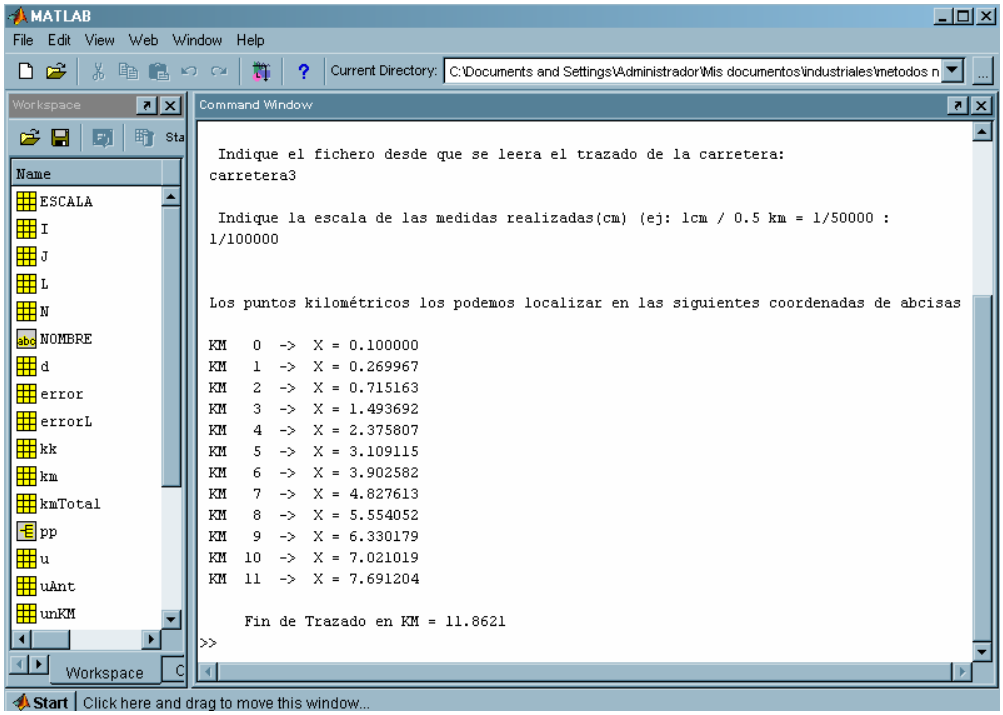


6.2. Ejemplo2

Repetiremos lo mismo para el caso de la última carretera cuyo fichero de entrada se llama “carretera3”, con escala 1/100000.



Donde obtenemos los siguientes resultados:



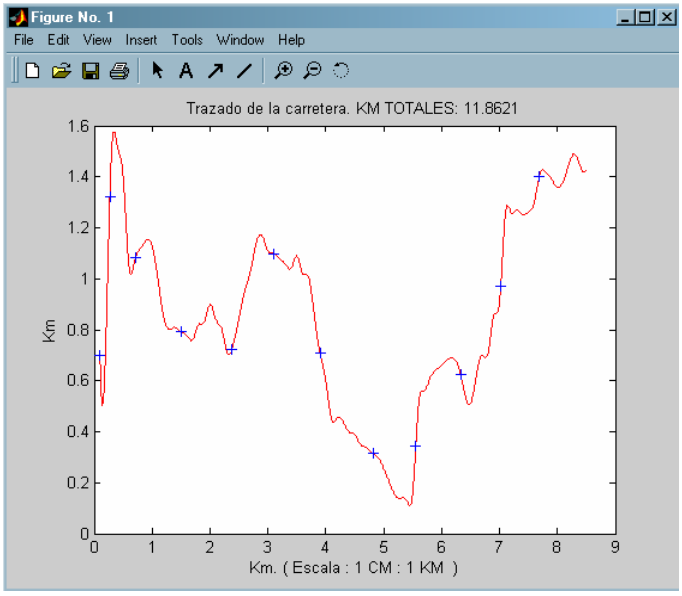


Figure No. 1
File Edit View Insert Tools Window Help
Trazado de la carretera. KM TOTALES: 11.8621
Km
Km. (Escala : 1 CM : 1 KM)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0 1 2 3 4 5 6 7 8 9