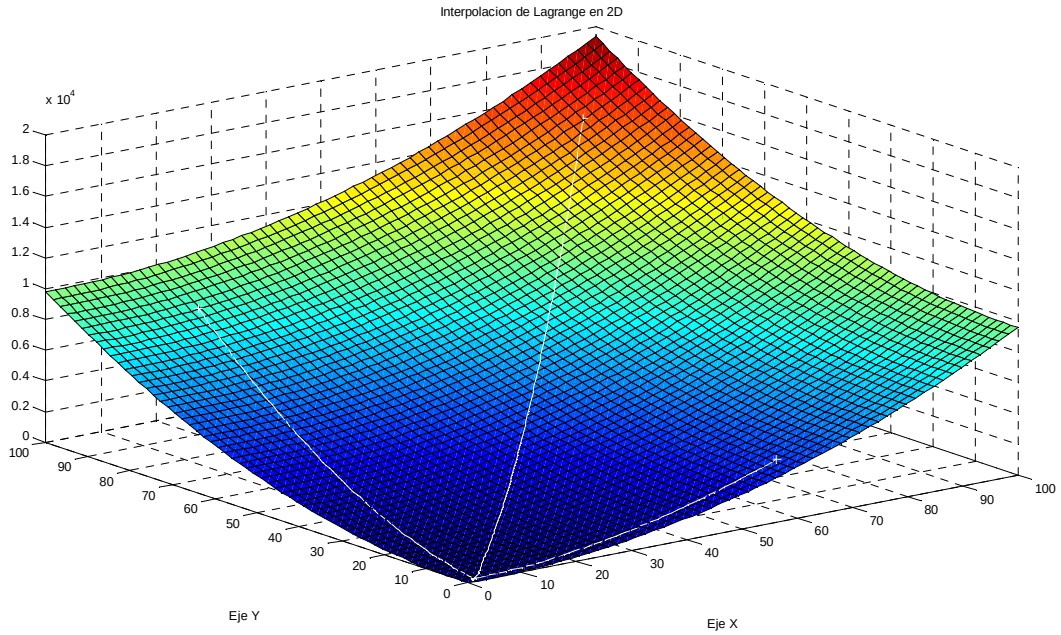
Universidad de Las Palmas de Gran Canaria
Escuela Técnica Superior de Ingenieros Industriales

"CÁLCULO DE LA TRAYECTORIA DE MÁXIMO DESCENSO DE UNA PARTÍCULA"



Mónica Gil Doreste
Mª Victoria Sánchez Henríquez
Marco A. Suárez González

Las Palmas de Gran Canaria, Junio 07

ÍNDICE

I MEMORIA

II PROGRAMACIÓN EN MAT LAB

I

MEMORIA

1. ENUNCIADO

MÉTODOS NUMÉRICOS - E.T.S.I.I.

TRABAJO DE LA ASIGNATURA - CURSO 2006-2007

Realizar un programa en MATLAB que construya un modelo tridimensional que defina la trayectoria de descenso por gravedad de una partícula (o un conjunto finito de ellas) sobre una orografía. El trabajo debe considerar los siguientes aspectos y requisitos:

a. Utilizar la interpolación de Lagrange para situar un punto sobre una topografía dada.

b. Utilizar el método de máximo descenso para definir la trayectoria.

c. Representar gráficamente tanto la superficie como las diferentes trayectorias obtenidas.

2. INTRODUCCIÓN

El objetivo de este trabajo es el de determinar la trayectoria de descenso por gravedad de una partícula sobre una orografía mediante el método del máximo descenso.

Para ello, previamente, se pedirá al usuario la ecuación que describa la superficie deseada y el punto de partida del movimiento de la partícula.

3. FUNDAMENTOS DEL PROGRAMA

El programa se dividirá en dos partes fundamentales:

- Creación de la orografía.
- Determinación de la trayectoria seguida por la partícula por el método del máximo descenso.

3.1. CREACIÓN DE LA OROGRAFÍA:

El programa pide al usuario la forma en que desea realizar la introducción de los datos, pudiendo elegir entre la entrada de éstos por medio de una función o por medio de un fichero de texto.

La creación de la orografía se hará en dos pasos. Primero se determinará el tamaño de la superficie en un plano, que se dividirá en una cuadrícula y después se le dará una altura correspondiente a cada punto que forme la misma.

Se le pide al usuario las medidas del largo y del ancho de la superficie, así como el número de puntos en que van a dividirse estas dos longitudes. Con estos datos, se crea una cuadrícula.

El siguiente paso consistirá en darle altura a los diferentes puntos que conforman la cuadrícula. Esto se hará mediante la introducción por parte del usuario de una función z= F(x, y). Se evaluará esta función matemática en cada punto resultante para así obtener la coordenada "z" para cada combinación de "x" e "y". Esta función será la que realmente dará la forma característica de la orografía.

Una vez creada la orografía requerida por el usuario, el programa se servirá del Método de Interpolación de Lagrange para crear una malla más tupida, es decir, de mayor definición. Se utilizará la malla de puntos creada previamente y solicitará nuevamente un número de puntos sobre el eje "x" e "y" para finalmente determinar las valores sobre el eje "z".

```
>> trayectoria_descenso
Formula de Interpolacion de lagrange en 2D
Elija la forma de entrada de datos:
1.  Generar datos usando una funcion F
2. Entrada desde un fichero de texto
Escriba 1 o 2
 1
Escriba la longitud de X y de Y en lineas separadas.
 50
 50
Escriba el numero de puntos en X y en Y en lineas separadas.
 20
 20
Escriba la funcion F(x,y) en terminos de x e y
Por ejemplo: y-x^2+1
 4*x-70*y^2
Elija la forma de salida de datos:
1. Pantalla
2. Fichero de texto
Escriba 1 o 2
 1
```

```
 50.00000   50.00000   20.00000   20.00000
  0.00000    0.00000    0.00000
  2.63158    0.00000   10.52632
  5.26316    0.00000   21.05263
  7.89474    0.00000   31.57895
 10.52632    0.00000   42.10526
 13.15789    0.00000   52.63158
 15.78947    0.00000   63.15789
 .......     ......    ......
 .......     ......    ......
 42.10526   50.00000 -174831.57895
 44.73684   50.00000 -174821.05263
 47.36842   50.00000 -174810.52632
 50.00000   50.00000 -174800.00000
Introduzca un nuevo numero de puntos para x y para y
 en lineas separadas mayor que los anteriores
 30
 30
```

El número de puntos introducido por segunda vez deberá ser mayor que los iniciales. En caso contrario, se producirá un error en el programa y habrá que volver a ejecutarlo. El error aparecerá por pantalla de la siguiente forma:

```
??? Error using ==> surface
X, Y, Z, and C cannot be complex.

Error in ==> C:\MATLAB6p5\toolbox\matlab\graph3d\surf.m
On line 68  ==> hh = surface(varargin{:});

Error in ==> C:\Documents and
Settings\Mavi\Escritorio\Trabajo_metodos_numericos\trayectoria_descenso.m
On line 165  ==> surf(XD,YD,ZD);
```

3.1.1. INTERPOLACIÓN DE LAGRANGE:

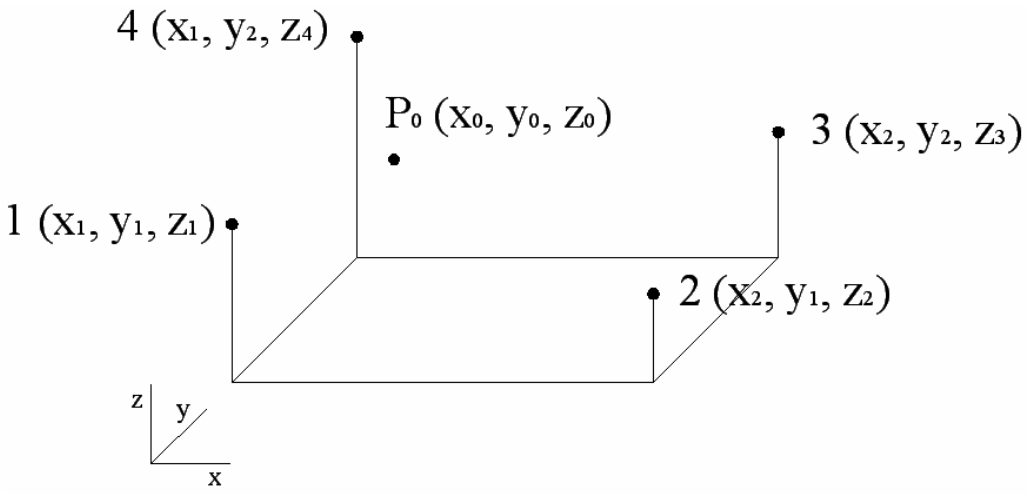
La interpolación polinómica es un método usado para conocer, de un modo aproximado, los valores que toma cierta función de la cual sólo se conocen sus valores en un número finito de puntos.

Un método de interpolación polinómica es el de Lagrange, el cual atiende a la siguiente fórmula:

$$l_j(x) = \prod_{i \neq j} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)(x - x_1)...(x - x_{j-1})(x - x_{j+1})...(x - x_n)}{(x_j - x_0)(x_j - x_1)...(x_j - x_{j-1})(x_j - x_{j+1})...(x_j - x_n)}$$

Para el caso concreto de la orografía (3 dimensiones), se tendrían cuatro puntos (P_1, P_2, P_3 y P_4) donde la función es conocida y se pretende calcularla en el punto $P_0(X_0, Y_0, Z_0)$. La ecuación quedaría de la siguiente forma:

$$Z_0 = \frac{(x_2 - x_0)\cdot(y_2 - y_0)}{(x_2 - x_1)\cdot(y_2 - y_1)} \cdot Z_1 - \frac{(x_1 - x_0)\cdot(y_2 - y_0)}{(x_2 - x_1)\cdot(y_2 - y_1)} \cdot Z_2 + \frac{(x_1 - x_0)\cdot(y_1 - y_0)}{(x_2 - x_1)\cdot(y_2 - y_1)} \cdot Z_3 + \\ + \frac{(x_0 - x_2)\cdot(y_1 - y_0)}{(x_2 - x_1)\cdot(y_2 - y_1)} \cdot Z_4$$



Del dibujo anterior se puede comprobar que las coordenadas “x” e “y” de los puntos 3 y 4 pueden expresarse como coordenadas de los puntos 1 y 2 puesto que coinciden con estos. No ocurre lo mismo con su coordenada z, que la determina el programa por medio de la función.

Determinación de P1, P2, P3 y P4:

$$nx = fix(x_0/CC)+1$$

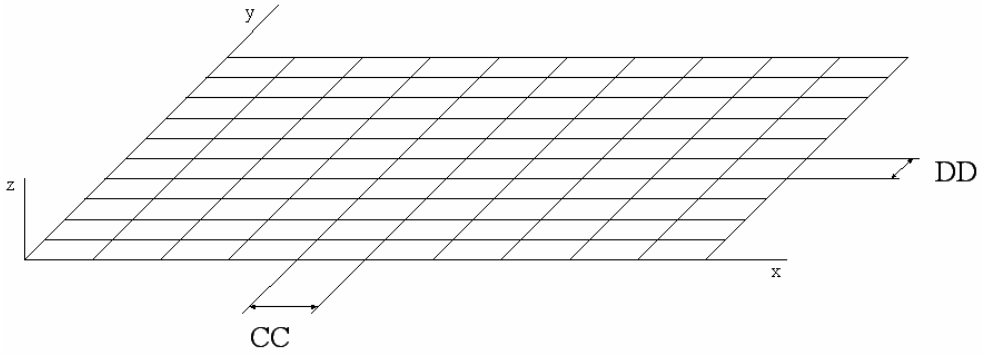
donde:

nx = es el número de intervalos sobre el eje x que existen desde el origen (0,0) hasta el cuadrante donde se encuentra el punto en cuestión
fix = orden perteneciente al programa Matlab que permite quedarnos con la parte entera de un número
x0 = Coordenada del punto inicial sobre el eje *x*
CC = Distancia existente entre dos puntos consecutivos sobre el eje *x*. Intervalo en *x*. (Es proporcionado por el programa previo).

$$ny = fix(y_0/DD)+1$$

donde:

ny = es el número de intervalos sobre el eje y que existen desde el origen (0,0) hasta el cuadrante donde se encuentra el punto en cuestión
fix = orden perteneciente al programa Matlab que permite quedarnos con la parte entera de un número
y_0 = Coordenada del punto inicial sobre el eje *y*
DD = Distancia existente entre dos puntos consecutivos sobre el eje *y*. Intervalo en *y*. (Es proporcionado por el programa previo).



Con los valores *nx* y *ny* ya se pueden determinar los valores de las coordenadas de los puntos que conforman el cuadrante según:

$$x_2 = nx \cdot CC$$

$$y_2 = ny \cdot DD$$

$$x_1 = x_2 - CC$$

$$y_1 = y_2 - DD$$

3.2. TRAYECTORIA DE LA PARTÍCULA:

En la segunda parte, el programa determinará la trayectoria seguida por la partícula.
Para facilitar la comprensión de la estructura interna del programa, se han creado una serie de funciones en archivos de matlab externos, a las que se invocará cuando sea preciso. Estas funciones realizarán los procesos necesarios para el cálculo de la trayectoria y se describirán posteriormente.

3.2.1. SITUACIÓN INICIAL DE LA PARTÍCULA

Para situar la partícula en la orografía, el usuario debe introducir las coordenadas que sitúan a la misma sobre el plano horizontal, con lo que se tendrá $P_0=(x_0, y_0)$

```
Introduzca las coordenadas X e Y del punto inicial de la trayectoria en lineas separadas.
 20
 40
```

El programa invoca a la función esta_dentro que comprobará mediante comparación entre las coordenadas del punto y la longitud de la orografía, si el punto P_0 se encuentra dentro de la cuadrícula preestablecida. En caso contrario aparecerá por pantalla el siguiente mensaje.

```
??? Error using ==> trayectoria_descenso
El punto tiene que estar dentro del intervalo de trabajo
```

** Los resultados de las siguientes funciones no saldrán por pantalla hasta el momento de la solución final.

Para darle la altura al punto $P_0=(x_0, y_0)$, se llama a la función evalua_z, que calcula la altura de z_0 mediante una interpolación lineal. Esta interpolación es una herramienta de Matlab, llamada ZI = INTERP2(X,Y,Z,XI,YI,'linear'), que devuelve el valor ZI interpolando los vectores X,Y,Z y los valores del punto XI e YI.

En conclusión, la situación inicial de la partícula queda perfectamente definida mediante sus tres coordenadas $P_0=(x_0, y_0, z_0)$ y que quedará representada en la superficie mediante una cruz color blanco.

En definitiva, lo que hasta ahora se ha conseguido es la determinación de la orografía y la situación de la partícula en el instante inicial. A continuación se describe el desarrollo del objetivo principal del trabajo: la trayectoria de la partícula.

3.2.2. DETERMINACIÓN DE LA TRAYECTORIA

Método de Máximo Descenso:

Como ya se había mencionado se seguirá el Método de Máximo Descenso cuya explicación, tomada de *Nocedal, Jorge et al. Numerical optimization, Springer-Verlag, Nueva York, 1999,* se detalla a continuación:

Dada una función diferenciable $f : \mathbb{R}^2 \to \mathbb{R}$, *hallar* $x^* \in \mathbb{R}^2$ *de modo que existe una vecindad* N *de* x^* *tal que*

$$f(x^*) \leq f(x)$$

para todo $x \in N$. *Tal punto se llama mínimo local de* f.

Los algoritmos para resolver este problema, dada una aproximación x_0 al mínimo local, generan una sucesión de iteraciones $\{x_i\}_{i=0}^{k}$ que termina cuando no se puede mejorar la aproximación o se alcanza alguna precisión prestablecida. Para moverse de un punto x_k al siguiente x_{k+1} en la iteración, se utiliza la información de f en x_k (y ocasionalmente la de los valores previos $x_0, \ldots, x_k$).

Hay dos formas de lograr esta sucesión. La estrategia que analizaremos aquí brevemente es la de búsqueda de línea. En este caso, el algoritmo escoge una dirección p_k y busca a lo largo de esta dirección desde el punto actual por un nuevo punto que reduzca el valor de f. Se necesita, además, calcular la

distancia que recorrerse en esta dirección, lo cual se puede conseguir resolviendo el siguiente problema de minimización

$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

Sin embargo, resolver este problema exactamente es muchas veces innecesario, y basta con aproximarse a una solución para calcular un nuevo punto de la sucesión.

Direcciones de búsqueda

La dirección de máximo descenso $-\nabla f_k$ es la elección más obvia para buscar el mínimo, pues es la dirección en la que f decrece más rápidamente. Este esquema nos conduce al método del descenso más pronunciado. En efecto, por el teorema de Taylor, tenemos que para cualquier dirección de búsqueda p y parámetro de tamaño de paso α, tenemos

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f_k + \frac{1}{2}\alpha^2 p^T \nabla^2 f(x_k + tp)\, p,$$

para algún $p \in (0, \alpha)$. La tasa de crecimiento de f a lo largo de p en x_k es el coeficiente de α, es decir, $p^T \nabla f_k$. Por lo tanto, la dirección unitaria p de máximo decrecimiento es la solución del problema

$$\min_{\|p\|=1} p^T \nabla f_k.$$

En virtud de que

$$p^T \nabla f_k = \|p\| \, \|\nabla f_k\| \cos\theta,$$

donde θ es ángulo entre p y ∇f_k, se infiere que el mínimo se alcanza cuando

$$p = -\frac{\nabla f_k}{\|\nabla f_k\|}.$$

Aunque con el método del descenso más pronunciado no necesita calcular segundas derivadas. De hecho, con cualquier dirección de descenso producirá un decremento en f, siempre que el tamaño de paso sea suficientemente pequeño. Para ver esto, acudimos al teorema de Taylor,

$$f(x_k + \epsilon p_k) = f(x_k) + \epsilon p_k^T \nabla f_k + O\left(\epsilon^2\right).$$

Si p_k es una dirección de descenso, el coseno del ángulo entre p_k y ∇f_k es negativo, luego $\epsilon p_k^T \nabla f_k < 0$, así que

$$f(x_k + \epsilon p_k) < f(x_k).$$

Cálculo del tamaño de paso

Tomar simplemente como dirección de descenso al vector unitario en dirección contraria al gradiente no es una buena elección pues puede no converger. En efecto, para una función tan simple como $f(x) = x^2 + y^2 + 1$, el gradiente $\nabla f = (2x, 2y)$ evaluado en $(0{,}5, 0)$ vale $(1, 0)$. Al movernos en la dirección opuesta a la este vector (es decir, buscando el mayor descenso), llegamos al punto $(-0{,}5, 0)$. Repitiendo el proceso de calcular el gradiente y moverse en dirección opuesta, nos mantendremos oscilando infinitamente entre $(0{,}5, 0)$ y $(-0{,}5, 0)$. Esto puede evitarse escogiendo un tamaño de paso para avanzar en la dirección opuesta a la del gradiente. Para calcular dicho tamaño de paso, utilizamos las condición de decrecimiento suficiente

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c\alpha_k \nabla f_k^T p_k, \tag{1}$$

donde $c_1 \in (0, 1)$ es una constante apropiada. Intuitivamente, que se cumpla esta desigualdad significa que la función

$$\phi(\alpha) = f(x + \alpha p)$$

está por debajo de la línea

$$l(\alpha) = f(x) + c\alpha \nabla f^T p,$$

la constante c se escoge generalmente como 1×10^{-4}.

Podemos utilizar el siguiente algoritmo para encontrar el tamaño de paso en el método del descenso más pronunciado.

El programa llama a la función calcula _trayectoria. Este función devuelve el valor de todos los puntos que conforman la trayectoria (en un vector T), un vector que guarda el paso para cada punto (alfas), devuelve el número de iteraciones realizadas hasta llegar a la solución, el punto (x,y) de la solución, así como el valor de z en dicho punto. La solución se obtendrá cuando se incumpla alguna de estas tres condiciones:

a. El punto esté dentro de la cuadrícula, si no lo está quiere decir que aunque no haya llegado al mínimo, la partícula ha salido de la superficie.
b. El número de iteraciones es menor que 1000, valor por el que se entiende que el programa se ha metido en un bucle infinito, en este caso se para el proceso.
c. La norma del nuevo punto menos el punto anterior es mayor que una tolerancia de 0.0001. En este caso se ha llegado a un mínimo.

Cada punto se calcula mediante la fórmula P = P + alfa*Pk, por lo que la función calcula_trayectoria obtendrá para cada punto tanto Pk con la función evaluaPk, como el tamaño del paso con la función calcula_alfa.

- Cálculo de Pk, función evaluaPk:

$$Pk=-\text{grad } F(x, y)$$

Se calcula el gradiente de la función en dicho punto. El gradiente será la derivada gráfica respecto a cada variable (x,y), asemejando la derivada a la tangente:

$$\text{Derivada respecto x = tang} = \frac{\text{EvaluaZ}(X, Y, Z, X(i+1), y) - \text{EvaluaZ}(X, Y, Z, X(i), y)}{Dx}$$

$$\text{Derivada respecto y = tang} = \frac{\text{EvaluaZ}(X, Y, Z, x, Y(j+1)) - \text{EvaluaZ}(X, Y, Z, x, Y(j))}{Dy}$$

Asumiendo constante la distancia entre dos puntos consecutivos de la cuadrícula para cada variable, Dx e Dy.

- Cálculo de alfa, función calcula_alfa:

Como se explicó en la página anterior, para el cálculo del tamaño de paso (alfa), se utiliza el siguiente algoritmo:

1. *Hacer* $\alpha \leftarrow \overline{\alpha}$
2. *Repetir 3 mientras* $f(x_k + \alpha p_k) > f(x_k) + c\alpha \nabla f_k^T p_k$.
3. *Hacer* $\alpha \leftarrow \rho\alpha$.
4. *Devolver* $\alpha_k = \alpha$.

Se asigna un valor inicial a alfa, y se repite el paso 3 mientras no se satisfaga la condición del paso 2. Dado que ρ es un valor comprendido entre (0,1), en nuestro caso $\rho = 0.1$, el valor de alfa irá disminuyendo hasta alcanzar un tamaño que haga cumplir la condición.

Finalmente, el programa saca por pantalla los resultados obtenidos: el punto inicial de la partícula con sus tres coordenadas, el número de iteraciones que fueron realizadas hasta llegar a la solución y las coordenadas del punto final.

El punto inicial de la trayectoria es (40,40,-1.119176e+005)
Numero de iteraciones: 11
Solucion: (3.999356e+001,5.000000e+001)
Valor de z: -1.748400e+005

Se unirán todos los puntos de la trayectoria, almacenados en el vector T, mediante guiones blancos para ver la trayectoria que realiza la partícula sobre la superficie dibujada anteriormente.

4. EJEMPLOS

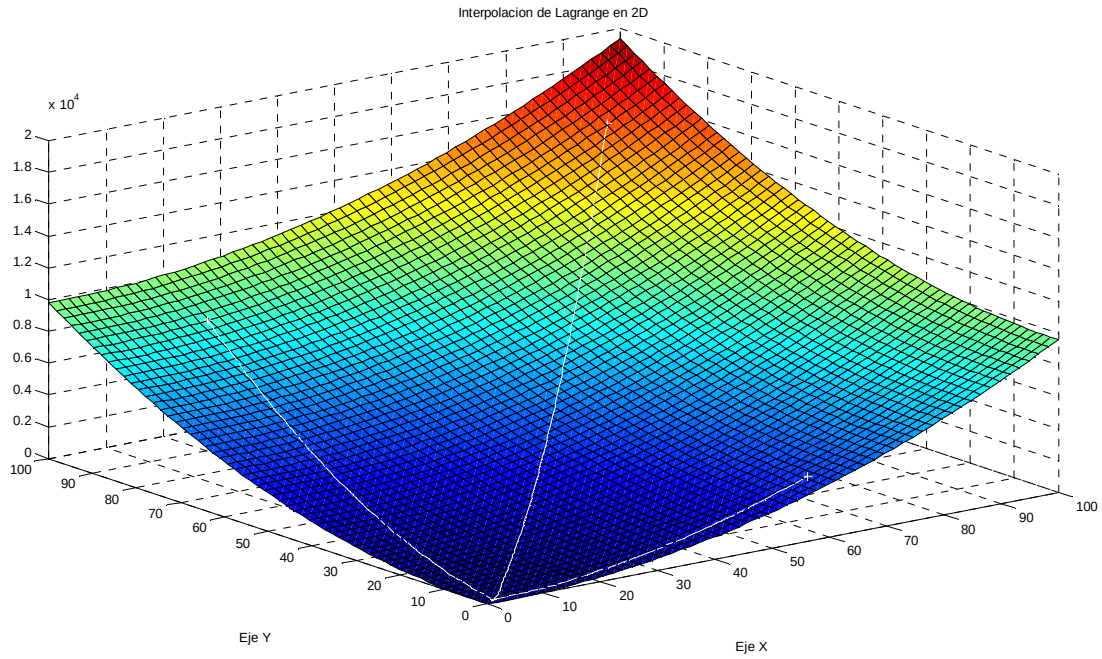
4.1. EJEMPLO 1

Formula de Interpolacion de lagrange en 2D
Elija la forma de entrada de datos:
1. Generar datos usando una funcion F
2. Entrada desde un fichero de texto
Escriba 1 o 2
1
Escriba la longitud de X y de Y en lineas separadas.
100
100
Escriba el numero de puntos en X y en Y en lineas separadas.
50
50
Escriba la funcion F(x,y) en terminos de x e y
Por ejemplo: y-x^2+1
(x-2)^2 + (y-1)^2
Elija la forma de salida de datos:
1. Pantalla
2. Fichero de texto
Escriba 1 o 2
1
100.00000 100.00000 50.00000 50.00000
0.00000 0.00000 5.00000
....
Introduzca un nuevo numero de puntos para x y para y en lineas separadas
60
60
Introduzca las coordenadas X e Y del punto inicial de la trayectoria en lineas separadas.
90
90
El punto inicial de la trayectoria es: (90,90,1.566575e+004)
Numero de iteraciones: 137
Solución: (1.695133e+000,9.924350e-001)
Valor de z: 1.720138e+000

Si repetimos el mismo proceso para varios puntos, introduciendo antes en pantalla el comando:

>> hold on

La superficie generada y trayectoria de máximo descenso para los diferentes puntos es la siguiente:



4.2. EJEMPLO 2

Formula de Interpolacion de lagrange en 2D

Elija la forma de entrada de datos:

1. Generar datos usando una funcion F

2. Entrada desde un fichero de texto

Escriba 1 o 2

1

Escriba la longitud de X y de Y en lineas separadas.

150

200

Escriba el numero de puntos en X y en Y en lineas separadas.

60

60

Escriba la funcion F(x,y) en terminos de x e y

Por ejemplo: y-x^2+1

8*x^2 + 3*x*y + 7*x^2 -25*x +31*y -29

Elija la forma de salida de datos:

1. Pantalla

2. Fichero de texto

Escriba 1 o 2

1

150.00000 200.00000 60.00000 60.00000

0.00000 0.00000 -29.00000

... ...

Introduzca un nuevo numero de puntos para x y para y en lineas separadas

70

70

Introduzca las coordenadas X e Y del punto inicial de la trayectoria en lineas separadas.

145

180

El punto inicial de la trayectoria es: (145,180,3.956370e+005)

Numero de iteraciones: 150

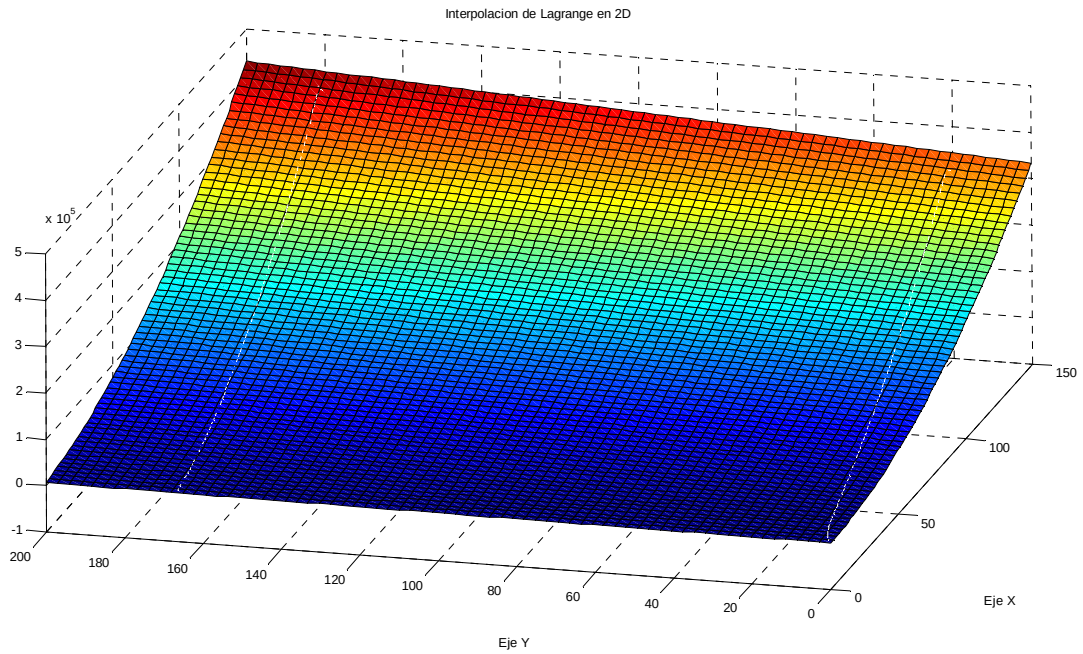
Solucion: (1.001024e-004,1.668517e+002)

Valor de z: 5.143453e+003

Si repetimos el mismo proceso para varios puntos, introduciendo antes en pantalla el comando:

>> hold on

La superficie generada y trayectoria de máximo descenso para los diferentes puntos es la siguiente:



4.3. EJEMPLO 3

Formula de Interpolacion de lagrange en 2D

Elija la forma de entrada de datos:

1. Generar datos usando una funcion F

2. Entrada desde un fichero de texto

Escriba 1 o 2

1

Escriba la longitud de X y de Y en lineas separadas.

200

150

Escriba el numero de puntos en X y en Y en lineas separadas.

50

60

Escriba la funcion F(x,y) en terminos de x e y

Por ejemplo: y-x^2+1

x^3 + y^2

Elija la forma de salida de datos:

1. Pantalla

2. Fichero de texto

Escriba 1 o 2

1

```
 200.00000   150.00000    50.00000    60.00000
   0.00000     0.00000     0.00000
   4.08163     0.00000    67.99888
   8.16327     0.00000   543.99102
      ...         ...         ...
```

Introduzca un nuevo numero de puntos para x y para y en lineas separadas

60

70

Introduzca las coordenadas X e Y del punto inicial de la trayectoria en lineas separadas.

190

75

El punto inicial de la trayectoria es: (190,75,6.867300e+006)

Numero de iteraciones: 219

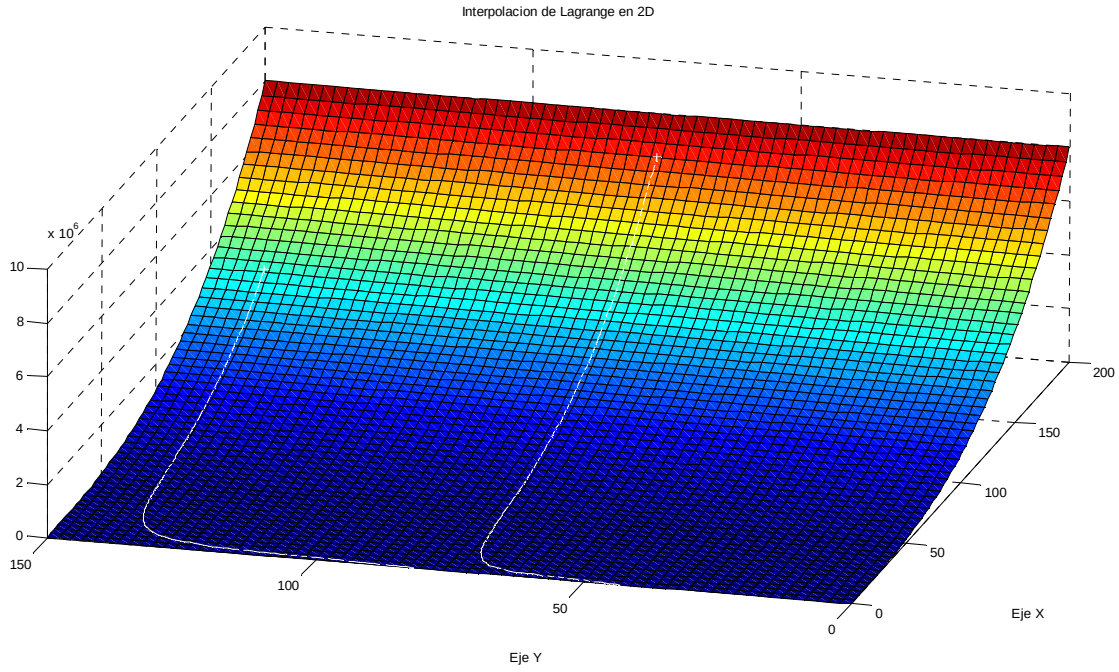
Solucion: (3.968941e-007,4.313534e+001)

Valor de z: 1.861971e+003

Si repetimos el mismo proceso para varios puntos, introduciendo antes en pantalla el comando:

>> hold on

La superficie generada y trayectoria de máximo descenso para los diferentes puntos es la siguiente:



5. CONCLUSION

El programa satisface el objetivo del presente trabajo, que es el de calcular la trayectoria que seguiría una partícula situada en una orografía por el Método de Máximo Descenso mediante el uso del programa MatLab. El correcto funcionamiento del programa queda corroborado por los ejemplos de diversa índole presentados anteriormente (apartado 4).

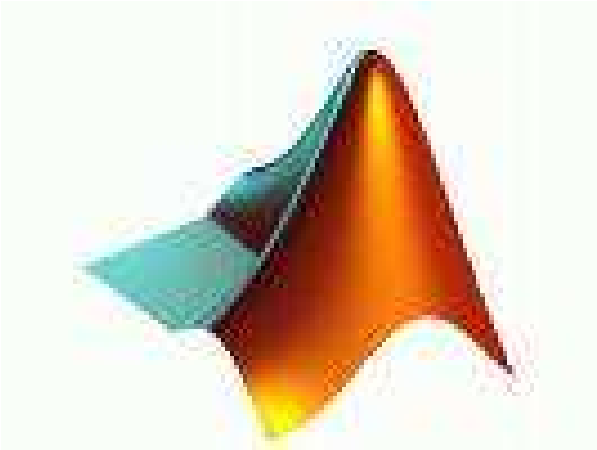
6. BIBLIOGRAFÍA

- *MATLAB. Edición de estudiante. Versión 4. Guía de usuario.Editorial Prentice Hall.*
- *Nocedal, Jorge et al. Numerical optimization, Springer-Verlag, Nueva York, 1999*

- http://www.geocities.com/octavioalberto.geo/math/practicaOpt01.pdf
- https://www.dma.uvigo.es/asignaturas/MN/files0/theList/sesion4.pdf
- http://www.doi.icai.upcomillas.es/simio/transpa/t_nlp_ar.pdf

II

PROGRAMACIÓN EN MAT LAB



```
1
2   function trayectoria_descenso
3
4   syms('F','OK','longX','longY','N','FLAG','NAME','OUP');
5   syms('M','FLAGO','DeltaX','X','s','DeltaY','XD','YD','ZD','NTOT');
6   syms('IP1','JP1','IP2','JP2','IP3','JP3','IP4','JP4');
7   syms('X1','YI','Z1','X2','Y2','Z2','X3','Y3','Z3','X4','Y4','Z4');
8   syms('D1','D2','D3','D4','N1','N2','N3','N4');
9   TRUE = 1;
10  FALSE = 0;
11  fprintf(1,'Formula de Interpolacion de lagrange en 2D\n');
12  OK = FALSE;
13  while OK == FALSE
14     fprintf(1,'Elija la forma de entrada de datos:\n');
15     fprintf(1,'1.  Generar datos usando una funcion F\n');
16     fprintf(1,'2. Entrada desde un fichero de texto\n');
17     fprintf(1,'Escriba 1 o 2 \n');
18     FLAG = input(' ');
19     if FLAG == 1 | FLAG == 2
20        OK = TRUE;
21     end
22  end
23  OK = FALSE;
24  while OK == FALSE
25     fprintf(1,'Escriba la longitud de X y de Y en lineas separadas.\n');
26     longX = input(' ');
27     longY = input(' ');
28     fprintf(1,'Escriba el numero de puntos en X y en Y en lineas separadas.\n')
29     N = input(' ');
30     M = input(' ');
31     if longX > 0 & longY > 0 & N > 0 & M > 0
32        X = zeros(N,M,3); %Almacena la los valores X,Y,Z
33        %para los puntos del espacio discretizado
34        OK = TRUE;
35     end
36  end
37  DeltaX = longX/(N-1);
38  DeltaY = longY/(M-1);
39
40  if FLAG == 1 %Generar datos usando una funcion
41     fprintf(1,'Escriba la funcion F(x,y) en terminos de x e y\n');
42     fprintf(1,'Por ejemplo: y-x^2+1\n');
43     s = input(' ','s');
44     F = inline(char(s),'x','y');
45     for j = 1 : M
46        for i = 1 : N
47           X(i,j,1) = (i-1)*DeltaX;
48           X(i,j,2) = (j-1)*DeltaY;
49           X(i,j,3) = F(X(i,j,1),X(i,j,2));
50        end;
51     end;
52     fprintf(1,'Elija la forma de salida de datos:\n');
```

```
    fprintf(1,'1. Pantalla\n');
    fprintf(1,'2. Fichero de texto\n');
    fprintf(1,'Escriba 1 o 2\n');
    FLAGO = input(' ');
    if FLAGO == 2
       fprintf(1,'Escriba el nombre del fichero '
       'de la forma - disco:\\nombre.ext\n');
       fprintf(1,'Por ejemplo   A:\\OUTPUT.DTA\n');
       NAME = input(' ','s');
       OUP = fopen(NAME,'wt');
    else
       OUP = 1;
    end;
    fprintf(OUP, '%11.5f %11.5f %11.5f %11.5f\n', longX,longY,N,M);
    for j = 1 : M
       for i = 1 : N
          fprintf(OUP, '%11.5f %11.5f %11.5f\n',
                       X(i,j,1),X(i,j,2),X(i,j,3));
       end;
    end;
    if OUP ~= 1
       fclose(OUP);
       fprintf(1,'Fichero de salida creado satisfactoriamente \n',NAME);
    end;
end;
if FLAG == 2
    fprintf(1,'Esta creado un fichero de texto'
    'con los datos en ¿TRES columnas?\n');
    fprintf(1,'donde la primera fila...?? e\n');
    fprintf(1,'Escriba Y o N\n')
    E = input(' ','s');
    if E == 'Y' | E == 'y'
       fprintf(1,'Escriba el nombre del fichero de la forma - ');
       fprintf(1,'disco:\\nombre.ext\n');
       fprintf(1,'Por ejemplo:   A:\\DATA.DTA\n');
       NAME = input(' ','s');
       INP = fopen(NAME,'rt');
       OK = FALSE;
       while OK == FALSE
          for j = 1 : M
             for i = 1 : N
                for k = 1 : 3
                   X(i,j,k) = fscanf(INP, '%f',1);
                end;
             end;
          end;
       end;
    end;
end;

%Interpolacion de Lagrange
```

```
104     XX = zeros(NN,MM,3); %almacenaremos los puntos del espacio interpolado
105     XD = zeros(1,NN);
106     YD = zeros(1,MM);
107     ZD = zeros(NN,MM);
108     OK = TRUE;
109  end
110  %nuevos valores Delta
111  CC = longX/(NN-1);
112  DD = longY/(MM-1);
113  for j = 1 : MM
114     for i = 1 : NN
115        XX(i,j,1) = (i-1)*CC;
116        XX(i,j,2) = (j-1)*DD;
117
118        IP1 = fix(XX(i,j,1)/DeltaX)+1;
119        JP1 = fix(XX(i,j,2)/DeltaY)+1;
120        if i==NN
121           IP1 = fix(XX(i,j,1)/DeltaX);
122        end;
123        if j==MM
124           JP1 = fix(XX(i,j,2)/DeltaY);
125        end;
126        IP2 = IP1+1;
127        JP2 = JP1;
128        IP3 = IP1+1;
129        JP3 = JP1+1;
130        IP4 = IP1;
131        JP4 = JP1+1;
132
133        X1 = X(IP1,JP1,1);
134        Y1 = X(IP1,JP1,2);
135        X2 = X(IP3,JP3,1);
136        Y2 = X(IP3,JP3,2);
137        Z1 = X(IP1,JP1,3);
138        Z2 = X(IP2,JP2,3);
139        Z3 = X(IP3,JP3,3);
140        Z4 = X(IP4,JP4,3);
141
142        D1 = ((X2-X1)*(Y2-Y1));
143        D2 = -D1;
144        D3 = D1;
145        D4 = D1;
146
147        N1 = (((X2-XX(i,j,1))*(Y2-XX(i,j,2)))/D1);
148        N2 = (((X1-XX(i,j,1))*(Y2-XX(i,j,2)))/D2);
149        N3 = (((X1-XX(i,j,1))*(Y1-XX(i,j,2)))/D3);
150        N4 = (((XX(i,j,1)-X2)*(Y1-XX(i,j,2)))/D4);
151
```

```
152         XX(i,j,3) = (N1*Z1+N2*Z2+N3*Z3+N4*Z4); % nuevo valor de Z
153
154         ZD(i,j)=XX(i,j,3);
155       end;
156   end;
157   ZD = ZD';
158   for i = 1 : NN
159      XD(i)=(i-1)*CC;
160   end;
161   for j = 1 : MM
162      YD(j)=(j-1)*DD;
163   end;
164   figure;
165   surf(XD,YD,ZD);
166   title('Interpolacion de Lagrange en 2D');
167   xlabel('Eje X');
168   ylabel('Eje Y');
169   hold on;
170   fprintf(1,'\n');
171
172   %Introduccion del punto inicial de la trayectoria.
173   fprintf(1,'Introduzca las coordenadas X e Y del punto inicial de'
174   'la trayectoria en lineas separadas.\n');
175   X0 = input(' ');
176   Y0 = input(' ');
177
178   if ~esta_dentro(XD,YD,[X0,Y0])
179   %con esta funcion se evalua si el punto esta dentro de los limites de la malla,
180   %el simbolo"~" es una negacion logica
181       error('El punto tiene que estar dentro del intervalo de trabajo');
182   %mensaje de error pq el punto tiene que estar dentro de la malla
183   end
184
185   Z0 = evaluaZ(XD,YD,ZD,X0,Y0);
186   %Evalua el valor de z en el terreno definido por X,Y,Z en el
187   %punto (x,y) y se guarda en Z0
188   fprintf('El punto inicial de la trayectoria es(%d,%d,%d)\n',X0,Y0,Z0);
189   %escribe en pantalla el punto inicial (X0,Y0) y la Z0 correspondiente
190   %calculada en la linea anterior
191   plot3(X0,Y0,Z0,'w+');  %dibuja una "+ blanca" en ese punto de la superficie
192
193   [T,sol,z,niter,alfas] = calcula_trayectoria(XD,YD,ZD,[X0,Y0]);
194   % minimiza la funcion expresada con X,Y,Z con el metodo del maximo descenso.
195   %Po es el punto inicial. devuelve la trayectoria T de puntos hasta llegar al minimo,
196   %la solucion y el valor de la funcion para ese punto,
197   %niter es el numero de iteraciones y alfas es un vector con
198   %los distintos alfas adoptados en cada paso del metodo
199   plot3(T(:,1),T(:,2),T(:,3),'w-');  %dibuja todos los puntos de la
200   %trayectoria uniendolos con "- blancos".
201   %Los puntos vienen representados en cada fila de la matriz T (con 3 columnas y n filas),
202   %con cada una de las variables (X,Y,Z) asociadas a una columna.
203   return;
```

FUNCIÓN ESTA_DENTRO

```
%Determina si un punto P = (x,y) esta dentro de los recintos
%de puntos definidos en X e Y
function b = esta_dentro(X,Y,P)
%cuando se ejecute la funcion quedara guardado en "b" un true o un false
    minx = X(1);
 %se guarda en minx el primer valor del vector X
 %(que es el que guarda los valores de x en cada punto)
    maxx = X(length(X));
 %se guarda en maxx el ultimo valor del vector X
 %(que es el que guarda los valores de x en cada punto);
 %length(X), calcula el numero de puntos de un vector dado
    miny = Y(1);
  %se guarda en miny el primer valor del vector Y
  %(que es el que guarda los valores de y en cada punto)
    maxy = Y(length(Y));
%se guarda en maxy el ultimo valor del vector y
%(que es el que guarda los valores de y en cada punto)

    if P(1) >= minx && P(1) <= maxx && P(2) >= miny && P(2) <= maxy
%se evalua el punto, guardado en el vector P (P=[X0,Y0]), y se devuelve
        b = true;
%un true si esta dentro de los limites y un false si no lo esta
    else
        b = false;
    end
return;
```

FUNCIÓN EVALUAZ

```
%Evalua el valor de z en el terreno definido por X,Y,Z en el punto (x,y)
function z = evaluaZ(X,Y,Z,x,y)
if length(X) ~= size(Z,2) || length(Y) ~= size(Z,1) %size(Z,2)->
    %devuelve el numero de columnas de la matriz Z; size(Z,1) ->
    %devuelve el numero de filas de la matriz Z; se compara el numero de
    %columnas de Z con el numero de elementos del vector X y se compara
    %el numero de filas de Z con el numero de elementos del vector Y
    error('La matriz Z tiene que tener las dimensiones de X x Y');
    %mensaje de error
end
if x < X(1) || x > X(length(X))
   error('x tiene que estar en el intervalo [%d,%d]',X(1),X(length(X)));
end %se evalua si el pto esta dentro de los limites de la malla
if y < Y(1) || y > Y(length(Y))
   error('y tiene que estar en el intervalo [%d,%d]',Y(1),Y(length(Y)));
end

%Calculamos la Z interpolando con los puntos que definen el terreno
z = interp2(X,Y,Z,x,y,'linear');
return;

%>> help interp2

%  INTERP2 2-D interpolation (table lookup).
%  ZI = INTERP2(X,Y,Z,XI,YI) interpolates to find ZI, the values of the
%  underlying 2-D function Z at the points in matrices XI and YI.
%  Matrices X and Y specify the points at which the data Z is given.
%  Out of range values are returned as NaN.
```

FUNCIÓN CALCULA_TRAYECTORIA

```
%minimiza la funcion expresada con X,Y,Z con el metodo del maximo
%descenso. Po es el punto inicial.
%devuelve la trayectoria T de puntos hasta llegar al minimo,
%la solucion y el valor de la funcion para ese punto
%
%niter es el numero de iteraciones y alfas es un vector con los
%distintos alfas adoptados en cada paso del metodo
%
function [T,sol,z,niter,alfas] = calcula_trayectoria(X,Y,Z,Po)

if length(X) ~= size(Z,2) || length(Y) ~= size(Z,1)
    error('La matriz Z tiene que tener las dimensiones de X x Y');
    %asegura que la matriz Z tenga las dimensiones adecuadas
end

npuntos = 0;
niter = 0;
P = Po; %vetor de tamaño 2 que almacena el pto [X0,Y0]
Panterior = P + [10,10]; %iniciamos a un numero diferente de P
TOL = 0.0001; %Tolerancia para una solucion razonable

while esta_dentro(X,Y,P) && niter < 1000 && norm(P-Panterior) > TOL
    %mientras este dentro de la malla, y numero de iteraciones no
    %sea mayor de 1000 (si es + de 1000 es que se metio en un bucle)
    %y la norma de P-Panterior (esta caminando en puntos tan proximos
    %que esta en el minimo)sea mayor que la tolerancia, se repetira el bucle
    npuntos = npuntos +1;
    T(npuntos,:) = [P(1),P(2),evaluaZ(X,Y,Z,P(1),P(2))];
    %guarda fila a fila los puntos de la trayectoria, el
    %valor de Z se obtiene con la funcion evaluaZ
    Pk = evaluaPk(X,Y,Z,P(1),P(2));
    %calculamos el "menor gradiente" en ese punto
    alfa = calcula_alfa(X,Y,Z,P(1),P(2));
    %calculamos alfa para este paso
    alfas(npuntos) = alfa;
    %almacenamos el valor de alfa para esa iteracion
    Panterior = P;
    P = P + alfa*Pk;        %nuevo pto
    niter = niter +1;
end
sol = T(npuntos,1:2);
%guarda en "sol" los valores de las 2 primeras columnas (X,Y)
%de la ultima fila de la matriz de la trayectorias, T
z = T(npuntos,3); %guarda en "z" el valor de la 3 columna y la ultima fila
fprintf('Numero de iteraciones: %d\n',niter);
fprintf('Solucion: (%d,%d)\n',T(npuntos,1),T(npuntos,2));
%muestra en pantalla las coordenadas (X,Y) del ultimo pto
fprintf('Valor de z: %d\n',T(npuntos,3));
%muestra el valor de "z" del ultimo pto
```

FUNCIÓN EVALUAPK

```
function Pk = evaluaPk(X,Y,Z,x,y)
%Pk = - grad f(x,y)

Pk = calcula_gradiente(X,Y,Z,x,y);
Pk = - Pk/norm(Pk); %vector unitario
return;
%se guarda en el vector Pk de tamaño 2
```

FUNCIÓN CALCULA_GRADIENTE

```

function g = calcula_gradiente(X,Y,Z,x,y)

%Determinamos el cuadrante en donde se encuentra el punto
[i,j] = cuadrante(X,Y,x,y);

%Asumimos que delta es constante en cada eje del mayado
Dx = X(2) - X(1);
Dy = Y(2) - Y(1);

%g = grad f(x,y)
g(1) = (EvaluaZ(X,Y,Z,X(i+1),y) - EvaluaZ(X,Y,Z,X(i),y)) / Dx;
%hacemos la derivada como la tangente = (distacia en Z
%correspondiente a los indices de las "x",
%dejando la "y" cte)/(distancia en x)
g(2) = (EvaluaZ(X,Y,Z,x,Y(j+1)) - EvaluaZ(X,Y,Z,x,Y(j))) / Dy;
%hacemos la derivada como la tangente = (distacia en Z
%correspondiente a los indices de las "y", dejando la "x" cte)/(distancia en y)
%se guarda en un vector de tamaño 2 que guarda la
%derivada de cada una de las variables
return;

```

FUNCIÓN CUADRANTE

```
%determina los valores i y j del cuadrante donde se encuentra el punto (x,y)
%
%el cuadrante seria el definido por los puntos (i,j),
%(i,j+1), (i+1,j) y (i+1,j+1)
function [I,J] = cuadrante(X,Y,x,y)

I = 0;
i = 1;
while i <= length(X) && X(i) <= x %se repetira el bucle mientras el indice
    %no se salga del tamaño del vetor X y mientras el valor del vector X
    %en ese indice sea menor que el de la variable x del punto
    I = i;
    i = i + 1;
end

J = 0;
j = 1;
while j <= length(Y) && Y(j) <= y %se repetira el bucle mientras el indice
    %no se salga del tamaño del vetor Y y mientras el valor del vector Y
    %en ese indice sea menor que el de la variable y del punto
    J = j;
    j = j + 1;
end

return;
```

FUNCIÓN CALCULA_ALFA

```
%Calcula alfa para el metodo del maximo descenso
%en funcion del punto actual y el terreno
function alfa = calcula_alfa(X,Y,Z,x,y)
    rho = 0.1; % rho entre (0,1)
    c = 1e-4; % c entre (0,1) normalmente ese valor fijo
    alfa_inicial = 1;

    gradiente = calcula_gradiente(X,Y,Z,x,y);
    %calcula el gradiente en ese punto
    p=-gradiente/norm(gradiente);
    %guarda el "menos gradiente" en un vector unitario de tamaño 2
    alfa= alfa_inicial;
    u =([x,y]+alfa*p);
    %si alfa es demasiado grande,
    %puede hacer que el nuevo punto se salga del terreno
    %corregimos si se da este caso
    while ~esta_dentro(X,Y,u)
        alfa = alfa / 2;
        u =([x,y]+alfa*p);
    end

    F = evaluaZ(X,Y,Z,u(1),u(2));
    %F(alfa) = f(Xk + alfa*Pk), valor de z en el pto
    %destino con el alfa inicial
    f = evaluaZ(X,Y,Z,x,y);%f(Xk), valor de z en el pto de partida
    %mientras f(Xk + alfa*Pk) > f(Xk) + c*alfa*grad(f)*Pk'
    while F > f+(c*alfa)*gradiente*p'   %p' por el producto de vectores
        alfa = alfa*rho;
        u =([x,y]+alfa*p);
        F = evaluaZ(X,Y,Z,u(1),u(2)); %F(alfa) = f(Xk + alfa*Pk)
    end

return;
```