Práctica 5:
Funciones de varias variables: cálculo diferencial y optimización con *Mathematica*

Análisis Matemático

Grado en Administración y Dirección de Empresas

J. Manuel Cascón Barbero (casbar@usal.es)
María Dolores García Sanz (dgarcia@usal.es)
Bernardo García-Bernalt Alonso (bgarcia@usal.es)
María Aurora Manrique García (amg@usal.es)
Gustavo Santos García (santos@usal.es)

1. Introducción

Dedicaremos esta práctica al análisis de funciones de varias variables con *Mathematica*. Comenzaremos recordando cómo se pueden definir. Por ejemplo, dada la función: $f:\mathbb{R}^3\longrightarrow\mathbb{R}$ descrita por $f(x,y,z)=e^{xy}+\cos(z)$, escribiríamos en *Mathematica*:

In[•]:=
```
f[x_, y_, z_] := Exp[x y] + Cos[z]
                 exponencial  coseno
```

In[•]:=
```
f[x, y, z]
```

Out[•]=
$$e^{x\,y}+\text{Cos}[z]$$

In[•]:=
```
f[1, 0, 3]
```

Out[•]=
$$1+\text{Cos}[3]$$

Si en cambio, queremos definir una función que toma valores vectoriales, por ejemplo $h:\mathbb{R}^2\longrightarrow\mathbb{R}^2$, $h(x,y)=(x^2+y^2,\ \log xy)$, se haría como sigue:

In[•]:=
```
h[x_, y_] := {x^2 + y^2, Log[x y]}
                          logaritmo
```

In[•]:=
```
h[x, y]
```

Out[•]=
```
{x^2 + y^2, Log[x y]}
```

Notemos que se han usado los operadores { }, para indicar que la función devuelve un vector (en este caso, de dimensión 2).

Este documento está organizado como sigue. En la sección 2 presentaremos las herramientas que permiten dibujar funciones de dos variables con valores reales. A continuación en la sección 3 calcularemos los límites reiterados y direccionales de funciones de dos variables. La sección 4 estará dedicada al cálculo diferencial. Con el objetivo de hacer esta práctica autocontenida, dedicaremos la sección 5 a recordar los comandos necesarios para el proceso de optimización. En las últimas secciones estudiaremos la optimización sin restricciones (o en abiertos), y con restricciones.

2. Representación gráfica

La función **Plot3D** permite representar funciones de dos variables con valores reales:

In[•]:=
```
? Plot3D
```

Out[•]=

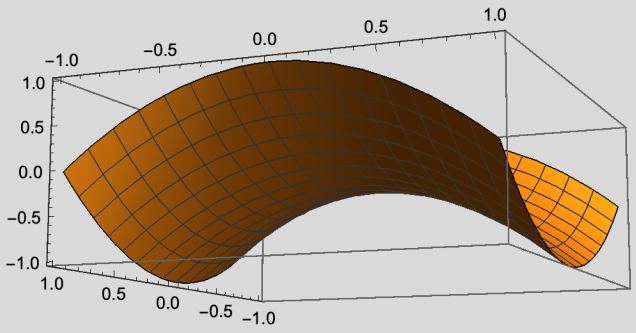
Symbol

Plot3D[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a three–dimensional plot of f as a function of x and y.
Plot3D[{f_1, f_2, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several functions.
Plot3D[{…, w[f_i], …}, …] plots f_i with features defined by the symbolic wrapper w.
Plot3D[…, {x, y} ∈ reg] takes variables {x, y} to be in the geometric region reg.

Por ejemplo, para dibujar la función f(x, y) = x^2 − y^2, en la región [-1,1]×[-1,1], escribiríamos:

In[•]:=
```
Plot3D[x^2 - y^2, {x, -1, 1}, {y, -1, 1}]
```
representación gráfica 3D

Out[•]=



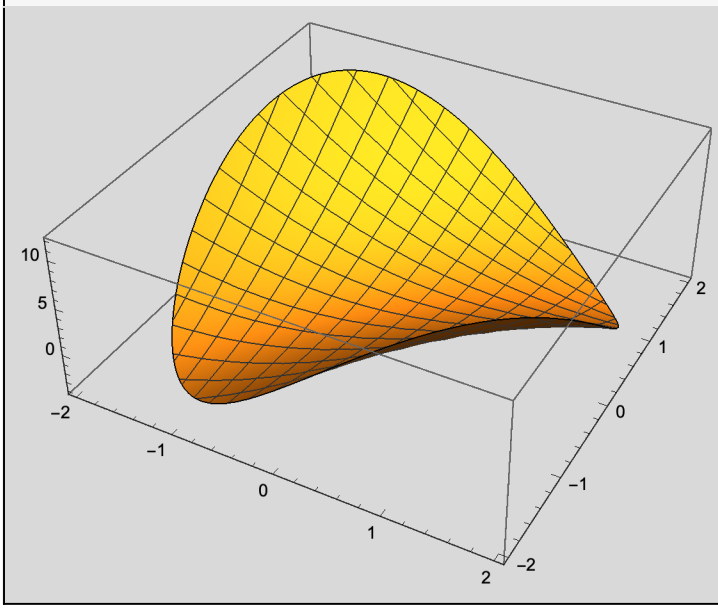
Observa que si localizas el cursor sobre el gráfico, *Mathematica* permite girarlo. Asimismo es posible modificar el tamaño del gráfico actuando sobre el marco del mismo.

La función **Plot3D** permite multitud de opciones. Como en otras ocasiones recomendamos consultar la ayuda para descubrirlas. Una de ellas, que resultará muy útil en los problemas de opti-

mización, es la funcionalidad que proporciona **RegionFunction** que permite dibujar la función solo en un recinto determinado. Por ejemplo, la instrucción:

In[ ]:=
```
Plot3D[x^2 + y^2 - 4 x y, {x, -2, 2}, {y, -2, 2},
 representación gráfica 3D
  RegionFunction → Function[{x, y, z}, x^2 + y^2 ≤ 4]]
 función de región        función
```

Out[ ]=

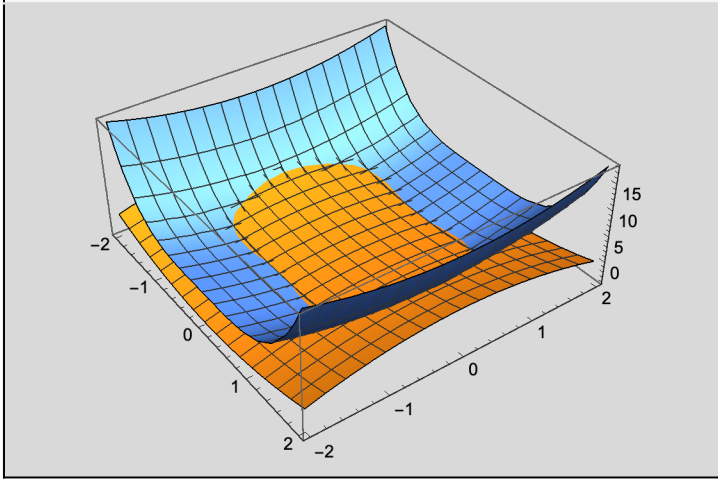


permite dibujar la función f(x,y) = $x^2 + y^2$ - 4 x y en el círculo $x^2 + y^2 \leqslant 4$.

Al igual que su homóloga para dos dimensiones, la función **Plot3D** permite dibujar varias superficies de forma simultánea. Basta con agruparlas entre llaves. Por ejemplo:

In[ ]:=
```
Plot3D[{x^2 - y^2, x^4 + y^2 - 2}, {x, -2, 2}, {y, -2, 2}, PlotRange → All]
 representación gráfica 3D                    rango de repr···  todo
```

Out[ ]=



Si las superficies vienen dadas en forma explícita F(x,y,z) = 0, se puede utilizar la función **ContourPlot3D**,

In[ ]:=
```
? ContourPlot3D
```

Out[ ]=

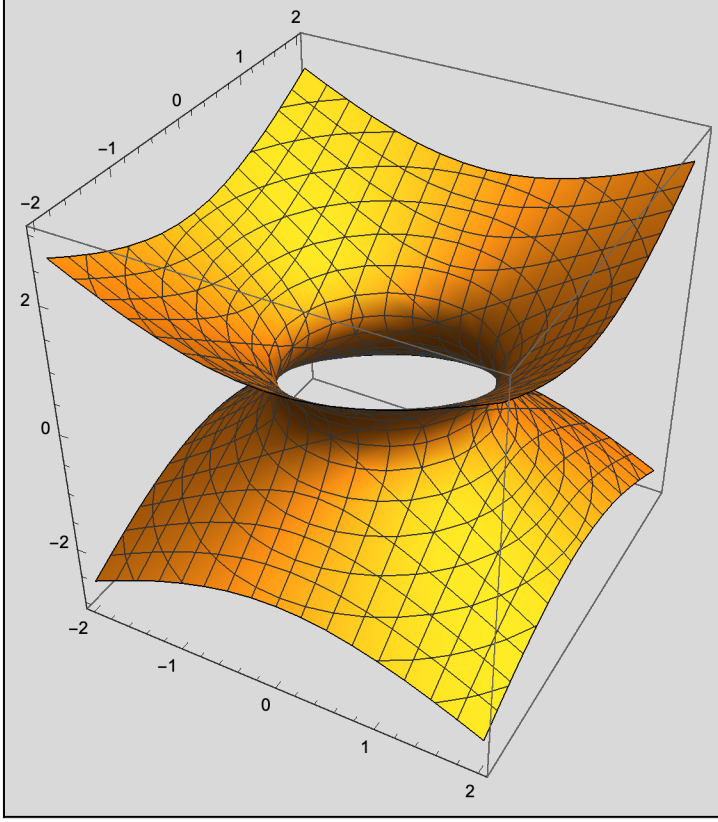
Symbol

ContourPlot3D[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}, {z, z_{min}, z_{max}}]
produces a three–dimensional contour plot of f as a function of x, y, and z.
ContourPlot3D[$f == g$, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}, {z, z_{min}, z_{max}}]
plots the contour surface for which f = g.

Ejemplos. Representar las superficies: $x^2 + y^2 - z^2 = 1$, $x^2 + 2xy - z^2 = 1$

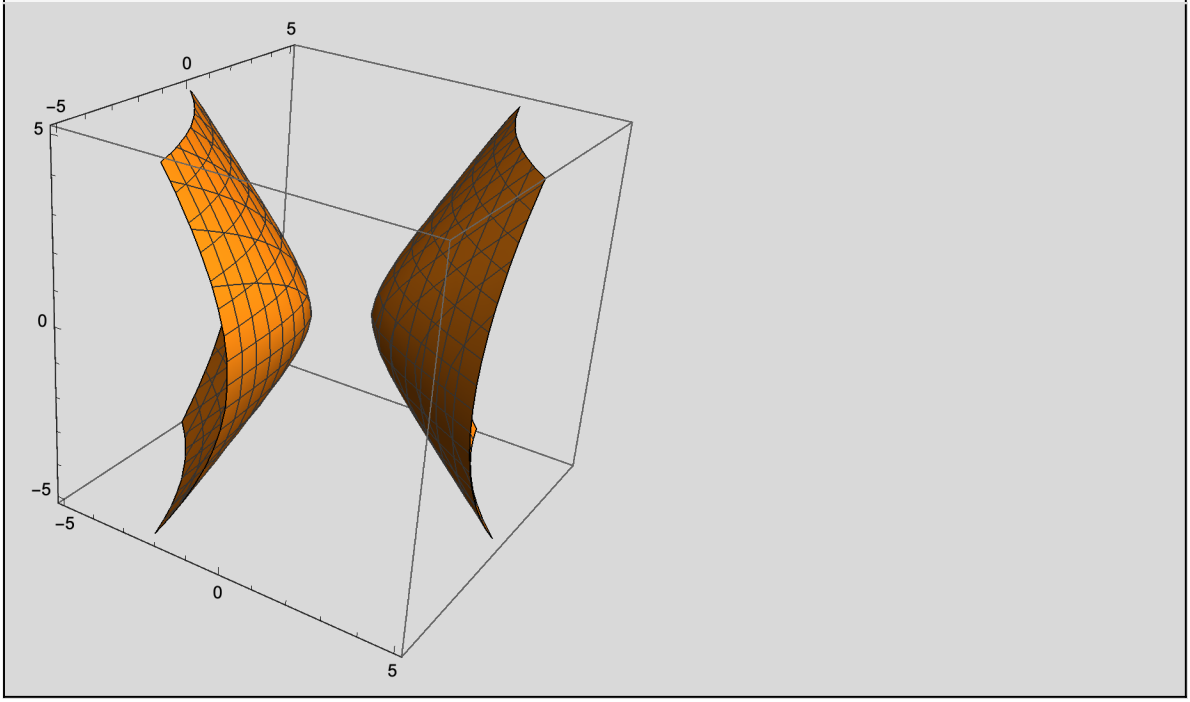
In[ ]:=
```
ContourPlot3D[x^2 + y^2 - z^2 == 1, {x, -2, 2}, {y, -2, 2}, {z, -3, 3}]
```
representación 3D de contornos

Out[ ]=



In[ ]:=
```
ContourPlot3D[x^2 + 2 x y  - z^2 == 1, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}]
```
representación 3D de contornos

Out[ ]=



3. Cálculo de límites

En esta sección veremos cómo calcular límites reiterados y direccionales con *Mathematica*. Como en ambos casos en definitiva se trata de hallar un límite en una variable, basta con utilizar la instrucción **Limit**:

In[ ]:=
```
? Limit
```

Out[ ]=

Symbol

Limit[$f[x], x \to x^*$] gives the limit $\lim_{x \to x^*} f(x)$.

Limit[$f[x_1, \ldots, x_n], \{x_1 \to x_1^*, \ldots, x_n \to x_n^*\}$] gives the nested limit $\lim_{x_1 \to x_1^*} \cdots \lim_{x_n \to x_n^*} f(x_1, \ldots, x_n)$.

Limit[$f[x_1, \ldots, x_n], \{x_1, \ldots, x_n\} \to \{x_1^*, \ldots, x_n^*\}$] gives the multivariate limit $\lim_{\{x_1, \ldots, x_n\} \to \{x_1^*, \ldots, x_n^*\}} f(x_1, \ldots, x_n)$.

3.1 Límites reiterados

Veamos con un ejemplo cómo se pueden calcular los límites reiterados con *Mathematica*. Consideramos la función:

f(x,y) = $\frac{x^2 - y^2}{x^2 + y^2}$

Para calcular $\lim_{x \to 0} \lim_{y \to 0}$ f(x,y) procedemos:

In[ ]:=
```
f[x_, y_] := (x^2 - y^3) / (x^2 + y^2)
```

In[•]:=
```
Limit[Limit[f[x, y], y → 0], x → 0]
límite  límite
```

Out[•]=
```
1
```

por otro lado, $\lim_{y\to 0}\lim_{x\to 0} f(x,y)$

In[•]:=
```
Limit[Limit[f[x, y], x → 0], y -> 0]
límite  límite
```

Out[•]=
```
0
```

De los resultados anteriores podemos concluir que la función f(x,y) no es continua en el punto (0,0).

3.2 Límites direccionales

Sea f(x,y)= $\frac{2xy^2-x^3}{x^2+y^4}$, para calcular los límites direccionales según la familia de rectas y = mx en el punto (0,0), se procede:

In[•]:=
```
f[x_, y_] := (2 x y^2 - x^3) / (x^2 + y^4)
```

In[•]:=
```
Limit[f[x, m x], x → 0]
límite
```

Out[•]=
```
0
```

Del resultado anterior no es posible decidir si la función f(x, y) es o no continua.Sin embargo, si calculamos el límite según la curva $x=y^2$ obtenemos,

In[•]:=
```
Limit[f[y^2, y], y → 0]
límite
```

Out[•]=
```
1
```

y por tanto, concluiremos que la función f(x,y) no es continua en (0,0), porque si lo fuera el límite debería ser 0.

4. Cálculo diferencial

4.1 Cálculo de derivadas parciales

Para el cálculo de derivadas parciales *Mathematica* dispone el comando **D** (el mismo que utilizábamos en el cálculo en una variable):

In[•]:=
```
? D
```

Out[•]=

Symbol

D[f, x] gives the partial derivative $\partial f/\partial x$.
D[f, {x, n}] gives the multiple derivative $\partial^n f/\partial x^n$.
D[f, x, y, …] gives the partial derivative $\cdots (\partial/\partial y)(\partial/\partial x) f$.
D[f, {x, n}, {y, m}, …] gives the multiple partial derivative $\cdots (\partial^m/\partial y^m)(\partial^n/\partial x^n) f$.
D[f, {{x_1, x_2, …}}] for a scalar f gives the vector derivative $(\partial f/\partial x_1, \partial f/\partial x_2, \ldots)$.
D[f, {$array$}] gives an array derivative.

Por ejemplo dada la función f (x, y, z) = x cos(x y z) – 3 z^2sen(x z), vamos a calcular: $\frac{\partial f}{\partial x}$, $\frac{\partial^2 f}{\partial x \partial y}$, $\frac{\partial^5 f}{\partial x^3 \partial y \partial z}$

In[•]:=
```
f[x_, y_, z_] := x Cos[x y z] - 3 z^2 Sin[x z]
```
coseno (Cos) · seno (Sin)

$\frac{\partial f}{\partial x}$:

In[•]:=
```
D[f[x, y, z], {x, 1}]
```
deriva (D)

Out[•]=
```
-3 z^3 Cos[x z] + Cos[x y z] - x y z Sin[x y z]
```

$\frac{\partial^2 f}{\partial x \partial y}$:

In[•]:=
```
D[f[x, y, z], {x, 1}, {y, 1}]
```
deriva (D)

Out[•]=
```
-x^2 y z^2 Cos[x y z] - 2 x z Sin[x y z]
```

$\frac{\partial^5 f}{\partial x^3 \partial y \partial z}$:

In[•]:=
```
D[f[x, y, z], {x, 3}, {y, 1}, {z, 1}]
```
deriva (D)

Out[•]=
```
-12 y z Cos[x y z] + 10 x^2 y^3 z^3 Cos[x y z] + 24 x y^2 z^2 Sin[x y z] - x^3 y^4 z^4 Sin[x y z]
```

4.2 Gradiente, jacobiano, diferencial

Los tres conceptos que trataremos en esta sección están relacionados como las derivadas primeras de una función, y se calculan del mismo modo en *Mathematica*. El nombre en realidad depende de cómo se interpreten. Así:

- Gradiente: En el caso de funciones a valores reales es un vector que índica la dirección de mayor variación.
- Jacobiano: Es la matriz asociada a la aplicación (lineal) diferencial.
- Diferencial: Es la aplicación diferencial.

El comando que usaremos en *Mathematica* para el cálculo de los operadores anteriores es **Grad**:

Grad[f, {x_1, ..., x_n}]
gives the gradient $(\partial f / \partial x_1, \ldots, \partial f / \partial x_n)$.

En general es necesario evaluar el jacobiano/gradiente/diferencial en un determinado punto para ello, y como ya vimos en la práctica anterior, es conveniente definir una nueva función utilizando el operador de asignación estática (=), en lugar del de asignación dinámica (:=), pues la definición involucra derivadas.

Veamos cómo funcionan estos comandos con varios ejemplos :

- Dada la función f(x,y,z) = x^2 –3 y+z–log xyz, calcular el ∇f en un punto genérico, y ∇f (1,1,1).

Definimos la función:

In[•]:=
```
f[x_, y_, z_] := x^2 - 3 y + z - Log[x y z]
                                 logaritmo
```

Calculamos el gradiente:

In[•]:=
```
Gradf[x_, y_, z_] = Grad[f[x, y, z], {x, y, z}]
                    gradiente
```

Out[•]=
$$\left\{-\frac{1}{x}+2\,x,\ -3-\frac{1}{y},\ 1-\frac{1}{z}\right\}$$

Evaluamos la función en el punto:

In[•]:=
```
Gradf[1, 1, 1]
```

Out[•]=
```
{1, -4, 0}
```

- Sea f : $\mathbb{R}^2 \to \mathbb{R}^3$ la función: f(x,y) = (x cos y, x sen y, x cos y sen y)

1) Calcular la matriz jacobiana de f en el punto (x0, y0)
2) Hallar la diferencial de f en (π, π/2)
3) Calcular df(π,π/2)(3, 2)

Comenzamos definiendo la función vectorial:

In[•]:=
```
f[x_, y_] := {x Cos[y], x Sin[y], x Cos[y] Sin[y]}
               coseno     seno     coseno  seno
```

Calculamos el jacobiano:

In[•]:=
```
Jf[x_, y_] = Grad[f[x, y], {x, y}]
             gradiente
```

Out[•]=
```
{{Cos[y], -x Sin[y]}, {Sin[y], x Cos[y]}, {Cos[y] Sin[y], x Cos[y]^2 - x Sin[y]^2}}
```

Para mostrarlo en forma matricial usamos **MatrixForm**:

In[•]:=
```
MatrixForm[Jf[x0, y0]]
```
forma de matriz

Out[•]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[y0] & -x0\,\text{Sin}[y0] \\ \text{Sin}[y0] & x0\,\text{Cos}[y0] \\ \text{Cos}[y0]\,\text{Sin}[y0] & x0\,\text{Cos}[y0]^2 - x0\,\text{Sin}[y0]^2 \end{pmatrix}$$

Para obtener la diferencial en(π, π/2), df(π,π/2)(x,y) , basta multiplicar el Jacobiano evaluado en el punto (π,π/2) por el vector (x,y):

In[•]:=
```
dfp[x_, y_] = Jf[Pi, Pi / 2].{x, y}
```
n··· número pi

Out[•]=
$$\{-\pi\, y,\ x,\ -\pi\, y\}$$

Por último evaluamos df(π,π/2) (x,y) en el punto (3,2):

In[•]:=
```
dfp[3, 2]
```

Out[•]=
$$\{-2\,\pi,\ 3,\ -2\,\pi\}$$

4.3 Derivada direccional

Si la función es de clase C ^{1}, la derivada direccional según un vector (v1 , . . . , vn) puede calcularse mediante la fórmula:

$D_{(v1,\ldots,vm)}$ f (x1, . . . , xn) = ∇ f (x1, . . .xn)·(v1, . . . , vn)

De modo que, de nuevo, la instrucción **Grad** nos permite calcular la derivada direccional de una función diferenciable. Por ejemplo, si f (x, y) = x^2 − 3 y + log (x y), para calcular la derivada direccional de f según el vector (1, 1,) en el punto (1, 2) escribiríamos:

In[•]:=
```
f[x_, y_] := x^2 - 3 y + Log[x y]
```
logaritmo

In[•]:=
```
Gradf[x_, y_] = Grad[f[x, y], {x, y}]
```
gradiente

Out[•]=
$$\left\{\frac{1}{x} + 2\,x,\ -3 + \frac{1}{y}\right\}$$

In[•]:=
```
Gradf[1, 2].{1, 1}
```

Out[•]=
$$\frac{1}{2}$$

4.4 Hessiano

Mathematica no dispone de una función para el cálculo del hessiano. No obstante no es difícil de implementarla mediante la instrucción **D**:

In[1]:=
```
HessianH[f_, x_List?VectorQ] := D[f, {x, 2}]
```

La celda anterior es una celda de inicialización, esto quiere decir que cada vez que se inicia una sesión y se evalúa una celda, *Mathematica* también evaluará esta celda. Esto nos garantiza tener definida la función que hemos creado, **HessianH**, en todo momento.

Por ejemplo, si queremos calcular el hessiano de la función: f (x, y, z) = sen(x) cos(y) e^z en el punto (π/2, π, 0) procederíamos del siguiente modo:

In[ ]:=
```
f[x_, y_, z_] := Sin[x] Cos[y] Exp[z]
```

In[ ]:=
```
Hessianf[x_, y_, z_] = HessianH[f[x, y, z], {x, y, z}]
```

Out[ ]=
```
{{-E^z Cos[y] Sin[x], -E^z Cos[x] Sin[y], E^z Cos[x] Cos[y]},
 {-E^z Cos[x] Sin[y], -E^z Cos[y] Sin[x], -E^z Sin[x] Sin[y]},
 {E^z Cos[x] Cos[y], -E^z Sin[x] Sin[y], E^z Cos[y] Sin[x]}}
```

En forma matricial:

In[ ]:=
```
MatrixForm[HessianH[f[x, y, z], {x, y, z}]]
```

Out[ ]//MatrixForm=
```
( -E^z Cos[y] Sin[x]   -E^z Cos[x] Sin[y]   E^z Cos[x] Cos[y]
  -E^z Cos[x] Sin[y]   -E^z Cos[y] Sin[x]   -E^z Sin[x] Sin[y]
   E^z Cos[x] Cos[y]   -E^z Sin[x] Sin[y]   E^z Cos[y] Sin[x] )
```

El hessiano en el punto (π/2, π, 0) es:

In[ ]:=
```
Hessianf[Pi / 2, Pi, 0]
```

Out[ ]=
```
{{1, 0, 0}, {0, 1, 0}, {0, 0, -1}}
```

In[ ]:=
```
MatrixForm[Hessianf[Pi / 2, Pi, 0]]
```

Out[ ]//MatrixForm=
```
( 1  0   0
  0  1   0
  0  0  -1 )
```

4.5 Polinomio de Taylor

El polinomio de Taylor permite obtener aproximaciones polinómicas de una función en un entorno del punto estudiado. La función que permite su cálculo en *Mathematica* es **Series** cuya sintaxis ya habíamos estudiado en el caso de funciones de una variable, y que ahora recordamos:

In[•]:=
```
? Series
```

Out[•]=

Symbol

Series[f, {x, x_0, n}] generates a power series expansion for f about the point $x = x_0$ to order $(x - x_0)^n$.
Series[f, {x, x_0, n_x}, {y, y_0, n_y}, …] successively finds series expansions with respect to x, then y, etc.

Por ejemplo, para calcular el polinomio de Taylor de la función f (x, y) = sin (x) cos (y) en el punto (0, 0) de orden 3, escribiríamos :

In[•]:=
```
Series[Sin[x] Cos[y], {x, 0, 3}, {y, 0, 3}]
```
serie seno coseno

Out[•]=
$$\left(1 - \frac{y^2}{2} + 0[y]^4\right) x + \left(-\frac{1}{6} + \frac{y^2}{12} + 0[y]^4\right) x^3 + 0[x]^4$$

Si queremos eliminar el resto (término O) y ademas expandir el polinomio, podemos usar:

In[•]:=
```
Expand[Normal[Series[Sin[x] Cos[y], {x, 0, 3}, {y, 0, 3}]]]
```
expande·· normal serie seno coseno

Out[•]=
$$x - \frac{x^3}{6} - \frac{x\,y^2}{2} + \frac{x^3\,y^2}{12}$$

En particular, podemos utilizar esta instrucción para calcular el plano tangente a una superficie (polinomio de grado 1).
Por ejemplo, dada la función
$z = 1 - x^2 - y^2$
, calcularemos el plano tangente en el punto (1, 1) *y* dibujaremos función *y* plano tangente :

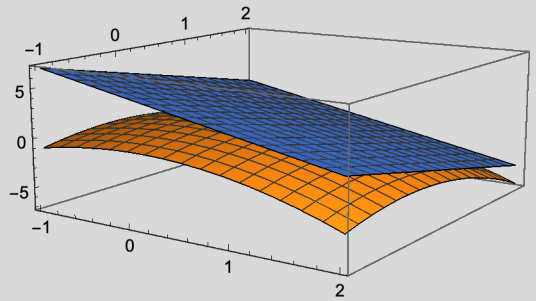
In[•]:=
```
f[x_, y_] := 1 - x^2 - y^2
```

In[•]:=
```
P[x_, y_] = Normal[Series[f[x, y], {x, 1, 1}, {y, 1, 1}]]
```
normal serie

Out[•]=
$$-1 - 2\,(-1 + x) - 2\,(-1 + y)$$

In[•]:=
```
Plot3D[{f[x, y], P[x, y]}, {x, -1, 2}, {y, -1, 2}]
```
representación gráfica 3D

Out[•]=



5. Comandos útiles

Con el objetivo de hacer esta práctica autocontenida dedicaremos esta sección a recordar algunos comandos necesarios para el proceso de optimización que hemos visto en prácticas anteriores.

- Resolución de sistemas de ecuaciones: **Solve**, **NSolve**

Para resolver sistemas de ecuaciones *Mathematica* dispone de varios comandos. Entre los más útiles tenemos **Solve** y **NSolve**. Algunos ejemplos:

In[•]:=
```
Solve[{x^2 - y^2 == 0, x^2 + 2 y^2 - x - y == 1}, {x, y}]
```
resuelve

Out[•]=
$$\left\{\left\{x \to -\frac{1}{3}, y \to -\frac{1}{3}\right\}, \{x \to 1, y \to 1\}, \left\{x \to -\frac{1}{\sqrt{3}}, y \to \frac{1}{\sqrt{3}}\right\}, \left\{x \to \frac{1}{\sqrt{3}}, y \to -\frac{1}{\sqrt{3}}\right\}\right\}$$

Dado que en nuestro caso solo nos interesan las soluciones reales, es conveniente añadir esta opción al resolver (de hecho, esto ayudará a *Mathematica*). Por ejemplo, observa la diferencia entre:

In[•]:=
```
Solve[{3 x^2 + 6 x y == 0, 3 x^2 - 4 y^3 - 4 y (1 + y^2) == 0}, {x, y}]
```
resuelve

Out[•]=
$$\left\{\{x \to 0, y \to 0\}, \left\{x \to -1, y \to \frac{1}{2}\right\}, \{x \to -2, y \to 1\}, \left\{x \to 0, y \to -\frac{i}{\sqrt{2}}\right\}, \left\{x \to 0, y \to \frac{i}{\sqrt{2}}\right\}\right\}$$

y resolver añadiendo la opción **Reals**:

In[•]:=
```
Solve[{3 x^2 + 6 x y == 0, 3 x^2 - 4 y^3 - 4 y (1 + y^2) == 0}, {x, y}, Reals]
```
resuelve números r

Out[•]=
$$\left\{\{x \to -2, y \to 1\}, \left\{x \to -1, y \to \frac{1}{2}\right\}, \{x \to 0, y \to 0\}\right\}$$

- Cálculo de valores propios: **Eigenvalues**

Para estudiar el carácter de una forma cuadrática disponemos de la instrucción **Eigenvalues** que proporciona los valores propios. Por ejemplo, dada la matriz A:

In[•]:=
```
A = {{4, 0, 0}, {0, 2, 2}, {0, 2, 2}}
```

Out[•]=
$$\{\{4, 0, 0\}, \{0, 2, 2\}, \{0, 2, 2\}\}$$

In[•]:=
```
MatrixForm[A]
```
forma de matriz

Out[•]//MatrixForm=
$$\begin{pmatrix} 4 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 2 & 2 \end{pmatrix}$$

Sus valores propios vienen dados por:

In[•]:=
```
Eigenvalues[A]
```
autovalores

Out[•]=
```
{4, 4, 0}
```

Como consecuencia podemos asegurar que A es una matriz semidefinida positiva.

- Cálculo del núcleo de una aplicación lineal

Para determinar el núcleo de una aplicación lineal podemos emplear el comando **NullSpace**, al que proporcionaremos la matriz asociada.

In[•]:=
```
? NullSpace
```

Out[•]=

Symbol

NullSpace[m] gives a list of vectors that forms a basis for the null space of the matrix m.

Por ejemplo, si f (x, y, z) = xyz, para calcular una base del núcleo de la aplicación diferencial en el punto (1, 1, 1), se procede del siguiente modo:

In[•]:=
```
f[x_, y_, z_] := x y z
```

In[•]:=
```
Jf[x_, y_, z_] = Grad[f[x, y, z], {x, y, z}]
```
gradiente

Out[•]=
```
{y z, x z, x y}
```

In[•]:=
```
NullSpace[{Jf[1, 1, 1]}]
```
espacio nulo

Out[•]=
```
{{-1, 0, 1}, {-1, 1, 0}}
```

Observa que para que *Mathematica* interprete que Jf es una matriz, hemos tenido que incluir una llave adicional.

6. Optimización en abiertos (sin restricciones)

6.1 Extremos locales

Optimizar $f(x_1, x_2, \ldots, x_n)$

Para resolver este tipo de problemas procedemos del siguiente modo:

1) Definimos la función objetivo.
2) Calculamos el gradiente / derivadas parciales.
3) Calculamos los puntos críticos (CPO - Condición de Primer Orden).

4) Clasificamos los puntos críticos (CSO - Condición de Segundo Orden).

4.1) Calculamos el hessiano (función)

4.2) Estudiamos el carácter de la evaluación del hessiano en cada uno de los puntos críticos:

Definido positivo --> mínimo local

Definido negativo --> máximo local

Indefinido ----> Punto silla

Semidefinido --> La CSO no decide

Ejemplo. Determinar los extremos relativos de la siguiente función:

f(x,y) = $x^3 + 3x^2y - 2y^2(1+y^2)$

1) Definimos la función objetivo:

In[•]:=
```
f[x_, y_] := x^3 + 3 x^2 y - 2 y^2 (1 + y^2)
```

2) Calculamos el gradiente:

In[•]:=
```
Gradf[x_, y_] = Grad[f[x, y], {x, y}]
                gradiente
```

Out[•]=
$$\{3x^2 + 6xy,\ 3x^2 - 4y^3 - 4y(1+y^2)\}$$

3) Calculamos los puntos críticos de f:

In[•]:=
```
Solve[Gradf[x, y] == {0, 0}, {x , y}, Reals]
resuelve                                números r
```

Out[•]=
$$\left\{\{x \to -2, y \to 1\}, \left\{x \to -1, y \to \frac{1}{2}\right\}, \{x \to 0, y \to 0\}\right\}$$

4.1) Calculamos el Hessiano:

In[•]:=
```
Hf[x_, y_] = HessianH[f[x, y], {x, y}]
```

Out[•]=
$$\{\{6x + 6y, 6x\}, \{6x, -20y^2 - 4(1+y^2)\}\}$$

4.2) Estudiamos el carácter del hessiano en cada punto crítico:

- (0,0):

In[•]:=
```
MatrixForm[Hf[0, 0]]
forma de matriz
```

Out[•]//MatrixForm=
$$\begin{pmatrix} 0 & 0 \\ 0 & -4 \end{pmatrix}$$

In[•]:=
```
Eigenvalues[Hf[0, 0]]
autovalores
```

Out[•]=
$$\{-4, 0\}$$

En este caso la matriz es semidefinida (negativa) por tanto la CSO no decide.

- (-1,1/2)

In[•]:=
```
MatrixForm[Hf[-1, 1/2]]
```
forma de matriz

Out[•]//MatrixForm=

$$\begin{pmatrix} -3 & -6 \\ -6 & -10 \end{pmatrix}$$

In[•]:=
```
Eigenvalues[Hf[-1, 1/2]] // N
```
autovalores valor numérico

Out[•]=
```
{-13.4462, 0.446222}
```

En este caso se trata de un punto de silla

- (-2,1)

In[•]:=
```
MatrixForm[Hf[-2, 1]]
```
forma de matriz

Out[•]//MatrixForm=

$$\begin{pmatrix} -6 & -12 \\ -12 & -28 \end{pmatrix}$$

In[•]:=
```
Eigenvalues[Hf[-2, 1]] // N
```
autovalores valor numérico

Out[•]=
```
{-33.2788, -0.721179}
```

Concluimos que el punto (-2,1) es un máximo local.

6.2 Extremos globales

En los casos en los que la función objetivo es diferenciable y cóncava (respectivamente convexa) el Teorema local-global asegura que todo punto crítico es máximo global (respectivamente mínimo global). Si además la función objetivo es de clase C^2, se puede determinar el carácter cóncavo (respectivamente convexo) estudiando si el hessiano es semidefinido negativo (respectivamente semidefinido positivo).
Para ilustrar el uso del teorema local-global resolveremos el siguiente ejercicio:

Ejercicio. Justificar que la función $f(x, y) = x^2 + y(y^3 - 4)$ tiene un extremo global en $\mathbb{R}^2$, indicando de qué tipo es.

Procedemos como en la sección anterior:

In[•]:=
```
f[x_, y_] := x^2 + y (y^3 - 4)
```

In[•]:=
```
Gf[x_, y_] = Grad[f[x, y], {x, y}]
```
gradiente

Out[•]=
```
{2 x, -4 + 4 y^3}
```

In[•]:=
```
Solve[Gf[x, y] == {0, 0}, {x, y}, Reals]
resuelve                                  números r
```

Out[•]=
```
{{x → 0, y → 1}}
```

Veamos si la función f(x,y) es convexa o cóncava en $\mathbb{R}^2$, lo cual, vía el teorema local-global, garantizará que estamos ante un extremo global. Para ello calculamos su hessiano:

In[•]:=
```
Hf[x_, y_] = HessianH[f[x, y], {x, y}]
```

Out[•]=
```
{{2, 0}, {0, 12 y^2}}
```

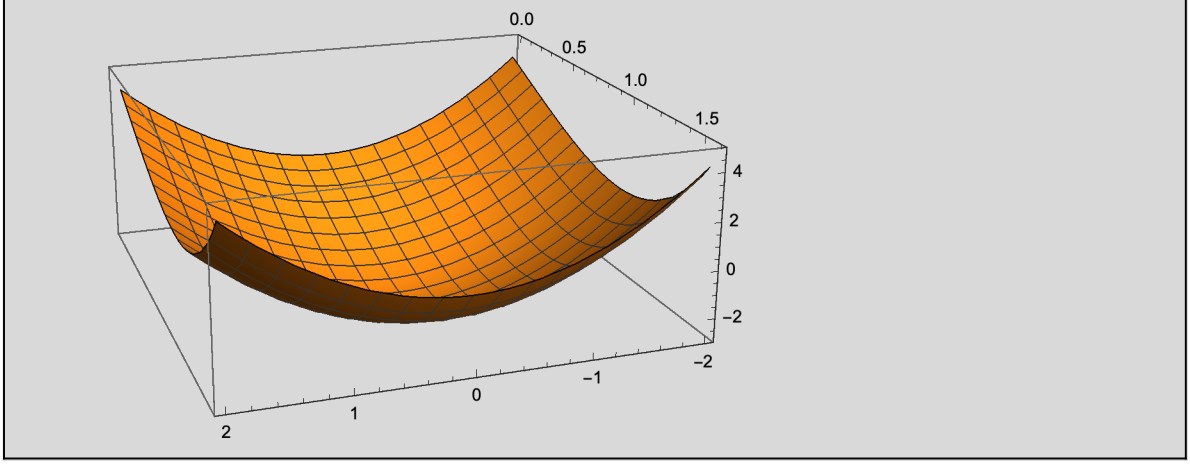
In[•]:=
```
MatrixForm[Hf[x, y]]
forma de matriz
```

Out[•]//MatrixForm=
```
( 2    0
  0  12 y^2 )
```

En cada punto la matriz es diagonal, por tanto los valores propios son 2 y 12 y^2 (los valores de la diagonal). Evidentemente ambos son mayores o iguales que cero en $\mathbb{R}^2$, por tanto concluimos que f(x,y) es convexa, y el punto crítico (0,1) es un mínimo global:

In[•]:=
```
Plot3D[f[x, y], {x, -2, 2}, {y, 0, 1.6}, PlotRange → {-3, 5}]
representación gráfica 3D                  rango de representación
```

Out[•]=



Nota: Todo lo que se ha descrito en esta sección es válido para problemas del tipo

Optimizar $f(x_1,x_2,\ldots,x_n)$
s.a. $\varphi(x_1,x_2,\ldots,x_n) < 0$

La única diferencia con lo descrito es que solo deben considerarse los puntos críticos que pertenecen al abierto, es decir, los que verifican la restricción.

7. Optimización con restricciones de igualdad

Dada una función objetivo $f(x_1,x_2,\ldots,x_n)$ y una restricción $\varphi(x_1,x_2,\ldots,x_n)$ en este caso nos planteamos el cálculo de los extremos relativos de f condicionados (o sujetos) a $\varphi(x_1,x_2,\ldots,x_n) = 0$,

es decir:

Optimizar $f(x_1,x_2, ..., x_n)$
s.a. $\varphi(x_1,x_2, ..., x_n) = 0$

Nota: La restricción φ puede ser una función con valores vectoriales, es decir, $\varphi = (\varphi_1 \ldots ,\varphi_m)$. En los ejercicios que resolveremos en esta práctica trabajaremos con 2 o 3 variables y 1 o 2 restricciones.

Existen dos procedimientos para resolver el problema anterior. El primero consiste en utilizar la restricción para eliminar (tras despejar) alguna o algunas de las variables, y de ese modo reducir el problema a uno de las características del que fue analizado en la sección anterior. La segunda alternativa está basada en el uso de los llamados multiplicadores de Lagrange.

7.1 Reducción a un extremo ordinario

El procedimiento es básicamente el mismo que el de la sección anterior con dos diferencias: es necesario redefinir la función objetivo de acuerdo a la restricción, y en este caso no existen puntos de ensilladura:

1) Definimos la función objetivo: F.

Usando la restricción $\varphi = \varphi(x_1,x_2, ..., x_n)$ se despejan m variables (supongamos que son las últimas) de $(x_1,x_2, ..., x_n)$ en función de las n –m restantes.

$f(x_1,x_2, ..., x_n)$ ----------> $F(x_1,x_2, ... , x_{n-m})$

Nota: Para poder realizar el proceso anterior con garantías de no perder ningún punto crítico, es necesario que se verifique una condición de regularidad. Revisar el material teórico para una exposición rigurosa.

2) Calculamos el Jacobiano de $F(x_1,x_2, ... , x_{n-m})$.
3) Calculamos los puntos críticos (CPO).
4) Clasificamos los puntos críticos. En este caso:
- Si el hessiano de F en un punto crítico es definido positivo, el punto crítico es un mínimo local condicionado.
- Si el hessiano de F en un punto crítico es definido negativo, el punto crítico es un máximo local condicionado.
- En el resto de casos no decide.

Ejemplo. Utilizaremos el esquema anterior para calcular los extremos relativos de $f (x, y, z) = x^2 + y^2 + z^2$ a lo largo de la curva $g(x,y,z) = 2x + 2y – z + 3 = 0$.

1) Definimos de la función objetivo. Mirando la restricción, observamos que es sencillo despejar la variable z en función de las restantes: $z = 2x + 2y + 3$

Definimos la función F(x,y) como:

In[•]:=
```
F[x_, y_] := x^2 +  y^2 + (2 x + 2 y + 3)^2
```

2) Calculamos el Jacobiano:

In[•]:=
```
JF[x_, y_] = Grad[F[x, y], {x, y}]
             gradiente
```

Out[•]=
$$\{2 x + 4 (3 + 2 x + 2 y), 2 y + 4 (3 + 2 x + 2 y)\}$$

3) Calculamos los puntos críticos de F:

In[•]:=
```
Solve[JF[x, y] == {0, 0}, {x, y}]
resuelve
```

Out[•]=
$$\left\{\left\{x \to -\frac{2}{3}, y \to -\frac{2}{3}\right\}\right\}$$

4) Clasificación de puntos críticos:

In[•]:=
```
HF[x_, y_] = HessianH[F[x, y], {x, y}]
```

Out[•]=
```
{{10, 8}, {8, 10}}
```

Dado que es es constante, no es necesario evaluar en el punto.

In[•]:=
```
Eigenvalues[HF[x, y]]
autovalores
```

Out[•]=
```
{18, 2}
```

De donde deducimos que el único punto crítico de F se trata de un mínimo de F. Recuerda que para recuperar el punto del problema original, debemos calcular primero el valor de su variable z:

In[•]:=
```
2 (-2/3) + 2 (-2/3) + 3
```

Out[•]=
$$\frac{1}{3}$$

Es decir, el mínimo de f sujeto a la restricción impuesta está localizado en el punto (-2/3, -2/3, 1/3).

7.2 Multiplicadores de Lagrange

En ocasiones no es posible utilizar la restricciones para eliminar algunas variables y es preciso utilizar el método de los multiplicadores de Lagrange. Para poder hacer uso de este procedimiento se requiere que la restricción cumpla la denominada condición de regularidad: Rang d φ (a) =m para todo punto crítico, donde m es el número de restricciones.

Exponemos ahora este procedimiento:

1) Definición del Lagrangiano asociado al problema. A partir de la restricción $\varphi = (\varphi_1 \ldots, \varphi_m)$ se

define la función:

$L(x_1,x_2, ..., x_n, \lambda_1, \ ..., \lambda_m) = f(x_1,x_2, ..., x_n) - \lambda_1 \varphi_1 (x_1,x_2, ..., x_n) - ... - \lambda_m \varphi_m (x_1,x_2, ..., x_n)$

2) Cálculo del Jacobiano del Lagrangiano $L(x_1,x_2, ..., x_n, \lambda_1, \ ..., \lambda_m)$

3) Cálculo de los puntos críticos mediante la condición de primer orden, CPO ($\nabla_x L = 0, \varphi = 0$)

4) Clasificación de los puntos críticos. Se calcula $H_x L$, en cada punto crítico y entonces:
- Si es una matriz definida positiva, el punto crítico es mínimo local condicionado.
- Si es una matriz definida negativa, el punto crítico es máximo local condicionado.
- En otro caso:
    i) Calculamos el jacobiano de la restricción φ en el punto crítico.
    ii) Calculamos una base del núcleo del jacobiano, que almacenamos en forma matricial y denotamos BJ.
    iii) Restringimos el hessiano HL al núcleo del jacobiano. Es decir, calculamos
    $$HR = BJ^t \ HL \cdot BJ$$
    iv) Clasificamos HR:
        - Si HR es definido positivo entonces el punto crítico es un mínimo local condicionado
        - Si HR es definido negativo entonces el punto crítico es un máximo local condicionado
        - Si HR es indefinido el punto crítico no es extremo local.
        - En otro caso, no decide.

Ejemplo. Calcular los extremos relativos de la función $f(x,y,z) = x^2 + y^2 + z^2$ condicionada por $2x + y + z = 2$ e $x - y - 3z = 4$.

1) Definición del Lagrangiano: Definimos la función f, y las restricciones g1, g2 (observa que todos los términos se han pasado al primer miembro). Denotaremos por L al lagrangiano y usaremos λ y μ como multiplicadores:

In[•]:=
```
f[x_, y_, z_] := x^2 + y^2 + z^2
```

In[•]:=
```
g1[x_, y_, z_] := 2 x + y + z - 2
```

In[•]:=
```
g2[x_, y_, z_] := x - y - 3 z - 4
```

In[•]:=
```
L[x_, y_, z_, λ_, μ_] := f[x, y, z] - λ g1[x, y, z] - μ g2[x, y, z]
```

In[•]:=
```
L[x, y, z, λ, μ]
```

Out[•]=
$$x^2 + y^2 + z^2 - (-2 + 2x + y + z)\,\lambda - (-4 + x - y - 3z)\,\mu$$

2) Cálculo del Jacobiano:

In[•]:=
```
GL[x_, y_, z_, λ_, μ_] = Grad[L[x, y, z, λ, μ], {x, y, z, λ, μ}]
                         |gradiente
```

Out[•]=
$$\{2x - 2\lambda - \mu,\ 2y - \lambda + \mu,\ 2z - \lambda + 3\mu,\ 2 - 2x - y - z,\ 4 - x + y + 3z\}$$

3) Cálculo de los puntos críticos:

```
In[•]:= Solve[GL[x, y, z, λ, μ] == {0, 0, 0, 0, 0}, {x, y, z, λ, μ}, Reals]
        resuelve                                                    números r
```

$$\text{Out[•]=}\quad \left\{\left\{x \to \frac{44}{31},\ y \to \frac{1}{31},\ z \to -\frac{27}{31},\ \lambda \to \frac{30}{31},\ \mu \to \frac{28}{31}\right\}\right\}$$

4) Clasificación de los puntos críticos:

```
In[•]:= HL[x_, y_, z_, λ_, μ_] = HessianH[L[x, y, z, λ, μ], {x, y, z}]
```

$$\text{Out[•]=}\quad \{\{2, 0, 0\}, \{0, 2, 0\}, \{0, 0, 2\}\}$$

```
In[•]:= MatrixForm[HL[x, y, z, λ, μ]]
        forma de matriz
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Observa que el hessiano es constante y definido positivo (tiene valor propio 2 con multiplicidad 3) y por tanto concluimos que el punto $\left(x \to \frac{44}{31},\ y \to \frac{1}{31},\ z \to -\frac{27}{31}\right)$ es un mínimo local condicionado.

Ejemplo (restricción al núcleo del Jacobiano). Calcular los extremos relativos de la función $f(x,y,z) = xy + yz + xz$ condicionada por $x + y + z = 6$.

1) Definición del Lagrangiano: Definimos la función f y la restricción g (nota que todos los términos se han pasado al primer miembro). Denotaremos por L al lagrangiano y usaremos λ como multiplicador:

```
In[•]:= f[x_, y_, z_] := x y + y z + x z
```

```
In[•]:= g[x_, y_, z_] := x + y + z - 6
```

```
In[•]:= L[x_, y_, z_, λ_] := f[x, y, z] - λ g[x, y, z]
```

```
In[•]:= L[x, y, z, λ]
```

$$\text{Out[•]=}\quad x\,y + x\,z + y\,z - (-6 + x + y + z)\,\lambda$$

2) Cálculo de Jacobiano:

```
In[•]:= GL[x_, y_, z_, λ_] = Grad[L[x, y, z, λ], {x, y, z, λ}]
                             gradiente
```

$$\text{Out[•]=}\quad \{y + z - \lambda,\ x + z - \lambda,\ x + y - \lambda,\ 6 - x - y - z\}$$

3) Cálculo de los puntos críticos:

In[•]:=
```
Solve[GL[x, y, z, λ] == {0, 0, 0, 0}, {x, y, z, λ}, Reals]
```
resuelve números r

Out[•]=
```
{{x → 2, y → 2, z → 2, λ → 4}}
```

4) Clasificación de los puntos críticos:

In[•]:=
```
HL[x_, y_, z_, λ_] = HessianH[L[x, y, z, λ], {x, y, z}]
```

Out[•]=
```
{{0, 1, 1}, {1, 0, 1}, {1, 1, 0}}
```

In[•]:=
```
MatrixForm[HL[x, y, z, λ]]
```
forma de matriz

Out[•]//MatrixForm=
$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

In[•]:=
```
Eigenvalues[HL[2, 2, 2, 4]]
```
autovalores

Out[•]=
```
{2, -1, -1}
```

El hessiano es indefinido, y por tanto, para continuar es necesario restringirlo al núcleo del jacobiano. En primer lugar calculamos el jacobiano de la restricción y su núcleo en el punto crítico:

In[•]:=
```
Jg[x_, y_, z_] = Grad[g[x, y, z], {x, y, z}]
```
gradiente

Out[•]=
```
{1, 1, 1}
```

In[•]:=
```
BJ = NullSpace[{Jg[2, -1, -1]}]
```
espacio nulo

Out[•]=
```
{{-1, 0, 1}, {-1, 1, 0}}
```

In[•]:=
```
MatrixForm[NullSpace[{Jg[2, -1, -1]}]]
```
forma de matriz espacio nulo

Out[•]//MatrixForm=
$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

Por último restringimos el hessiano:

In[•]:=
```
HR = BJ. HL[2, 2, 2, 4].Transpose[BJ]
```
transposición

Out[•]=
```
{{-2, -1}, {-1, -2}}
```

In[ ]:=
```
MatrixForm[HR]
```
forma de matriz

Out[ ]//MatrixForm=

$$\begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix}$$

In[ ]:=
```
Eigenvalues[HR]
```
autovalores

Out[ ]=
```
{-3, -1}
```

Por tanto, el hessiano restringido es definido negativo, y podemos concluir que el punto (2,2,2) es un máximo local condicionado.

7.3 Extremos globales

El problema modelo en este caso sería: dada una función objetivo $f(x_1,x_2, ..., x_n)$ y una restricción $\varphi(x_1,x_2, ..., x_n)$ nos planteamos el cálculo de los extremos globales de f condicionados (o sujetos) a $\varphi(x_1,x_2, ..., x_n) = 0$. Si la restricción φ determina un conjunto compacto, el problema anterior se simplifica gracias al teorema de Weierstrass y no requiere el uso de las condiciones de segundo orden.

Utilizando el teorema de Weierstrass basta con determinar los punto críticos y evaluar la función en dichos puntos para determinar el máximo y el mínimo. La dificultad en este tipo de problemas radica en asegurar si la restricción delimita o no un compacto. Sin embargo, en *Mathematica* y si la dimensión es 2 o 3 siempre es posible dibujar la curva o superficie asociada para asegurarnos de que es acotada y por tanto delimita un compacto.

El esquema, **cuando estemos seguros de que las restricciones delimitan un compacto**, sería el siguiente:

1) Definición del lagrangiano.

$$L(x_1,x_2, ..., x_n, \lambda_1, ..., \lambda_m) = f(x_1,x_2, ..., x_n) - \lambda_1 \varphi_1 (x_1,x_2, ..., x_n) - ... - \lambda_m \varphi_m (x_1,x_2, ..., x_n)$$

2) Cálculo del Jacobiano del Lagrangiano.
3) Cálculo de los puntos críticos mediante la condición de primer orden (CPO).
4) Para cada punto crítico se evalúa la función f en dicho punto crítico. El punto (o los puntos) crítico(s) que proporcione(n) el mayor/menor valor será(n) máximo(s)/mínimo(s) absoluto(s) respectivamente.

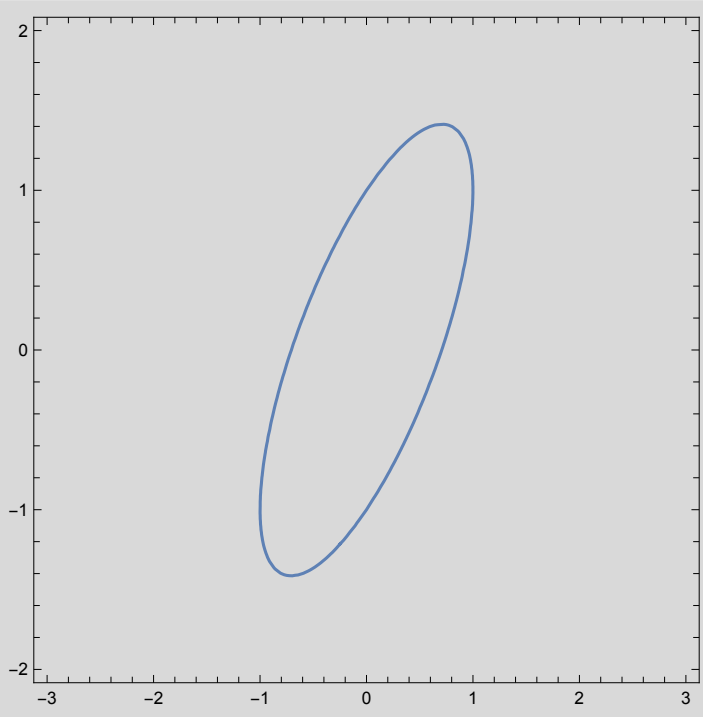
Ejemplo. Calcular los extremos absolutos de la función $f(x, y) = x^2 + y^2 - 2 x y$ condicionados a $g(x, y) = 2 x^2 + y^2 - 2 x y - 1 = 0$.

1) Comprobar que la restricción g(x,y) = 0 delimita un compacto. Para ello vamos a dibujar la curva g(x, y) = 0 comprobar que es un elipse:

In[ ]:=
```
g[x_, y_] := 2 x^2 + y^2 - 2 x y - 1
```

In[ ]:=
```
ContourPlot[g[x, y] == 0, {x, -3, 3}, {y, -2, 2}]
```
representación de contornos

Out[ ]=



2) Definición del Lagrangiano.

In[ ]:=
```
f[x_, y_] := x^2 + y^2 - 2 x y
```

In[ ]:=
```
L[x_, y_, λ_] := f[x, y] - λ g[x, y]
```

3) Cálculo del Jacobiano del Lagrangiano :

In[ ]:=
```
GL[x_, y_, λ_] = Grad[L[x, y, λ], {x, y, λ}]
```
gradiente

Out[ ]=
$$\{2 x - 2 y - (4 x - 2 y)\ \lambda,\ -2 x + 2 y - (-2 x + 2 y)\ \lambda,\ 1 - 2 x^2 + 2 x y - y^2\}$$

4) Cálculo de los puntos críticos mediante la condición de primer orden (CPO)

In[ ]:=
```
Solve[GL[x, y, λ] == {0, 0, 0}, {x, y, λ}]
```
resuelve

Out[ ]=
```
{{x → -1, y → -1, λ → 0}, {x → 0, y → -1, λ → 1},
 {x → 0, y → 1, λ → 1}, {x → 1, y → 1, λ → 0}}
```

5) Para cada punto crítico se evalúa la función

In[ ]:=
```
f[-1, -1]
```

Out[ ]=
```
0
```

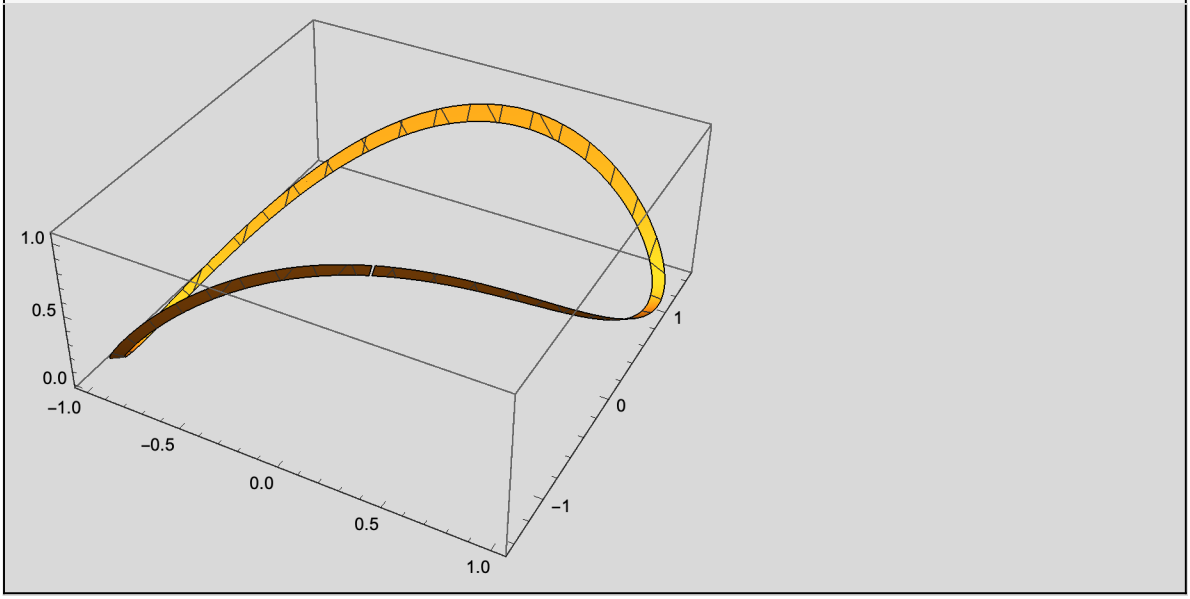
In[•]:=
```
f[0, -1]
```
Out[•]=
```
1
```

In[•]:=
```
f[0, 1]
```
Out[•]=
```
1
```

In[•]:=
```
f[1, 1]
```
Out[•]=
```
0
```

De donde deducimos que (-1,-1) y (1,1) son mínimos globales y (0,-1) y (0,1) son máximos globales. Podemos corroborar este resultado, dibujando la función en la restricción:

In[•]:=
```
Plot3D[f[x, y], {x, -2, 2}, {y, -2, 2}, RegionFunction →
 representación gráfica 3D                 función de región
  Function[{x, y, z}, 0.95 ≤ 2 x^2 + y^2 - 2 x y  ≤ 1.05], PlotPoints → 60 ]
  función                                                   número de puntos en la r
```
Out[•]=



Observación. En la instrucción anterior en lugar de escribir la restricción directamente (a través de la opción **RegionFunction**) 2 x^2 + y^2 - 2 x y = 1, hemos indicado el recinto 0.95≤ 2 x^2 + y^2 - 2 x y ≤ 1.05. Esto es debido a que Plot3D, dibuja superficies y no curvas. Este "tanteo" ser realiza de forma manual, hasta conseguir un gráfico aceptable. La opción **PlotPoints** controla la resolución en el dibujo. A mayor número de puntos, mayor resolución y por tanto mayor tiempo de procesado.

8. Optimización con restricciones de desigualdad

8.1 Extremos locales

Buscamos en este caso determinar los extremos de una función $f(x_1,x_2, ..., x_n)$ sujeta a la restricción $\varphi(x_1,x_2, ..., x_n)\leq 0$, cuya formulación es:

Optimizar $f(x_1,x_2, ..., x_n)$
s.a. $\varphi(x_1,x_2, ..., x_n) \leq 0$

En principio la función $\varphi(x_1,x_2, ..., x_n)$ podría ser de naturaleza vectorial, lo que correspondería a varias restricciones. Con el objetivo de simplificar la práctica y la exposición, en lo que sigue $\varphi(x_1,x_2, ..., x_n)$ denotará una función real, es decir, que en la formulación del problema aparecerá una única restricción.

El problema anterior se puede descomponer en dos subproblemas:

1. Optimización en un abierto:

Optimizar $f(x_1,x_2, ..., x_n)$
s.a. $\varphi(x_1,x_2, ..., x_n) < 0$

que se resuelve como fue descrito en la sección 6, con el matiz de que sólo se admiten los extremos que cumplan la condición $\varphi(x_1,x_2, ..., x_n) < 0$

2. Optimización con restricción de igualdad.

Optimizar $f(x_1,x_2, ..., x_n)$
s.a. $\varphi(x_1,x_2, ..., x_n) = 0$

que se resuelve como fue indicado en la sección 7.

Nota. Si la función $\varphi(x_1,x_2, ..., x_n)$ es de naturaleza vectorial, es decir existen varias restricciones, el problema original se divide en dos subproblemas: uno en el interior y otro en la frontera. La resolución sobre la frontera, puede dar lugar a su vez, a diferentes subproblemas que deben ser analizados en cada caso. Como adelantábamos al inicio de esa sección, este tipo de problemas no serán objeto de este curso.

Ejemplo. Determinar los extremos relativos de la función f(x,y) = $x^3+y^3+x^2+2\,y^2$ sujeta a la restricción $(x-1)^2+y^2 \leq 1$.

1) Optimización sin restricciones:

In[ ]:=
```
f[x_, y_] := x^3 + y^3 + x^2 + 2 y^2
```

In[ ]:=
```
g[x_, y_] := (x - 1)^2 + y^2 - 1
```

In[ ]:=
```
Jf[x_, y_] = Grad[f[x, y], {x, y}]
             gradiente
```

Out[ ]=
```
{2 x + 3 x^2, 4 y + 3 y^2}
```

In[•]:=
```
Solve[Jf[x, y] == {0, 0}, {x, y}, Reals]
```
resuelve números r

Out[•]=
$$\left\{\left\{x \to -\frac{2}{3}, y \to -\frac{4}{3}\right\}, \left\{x \to 0, y \to -\frac{4}{3}\right\}, \left\{x \to -\frac{2}{3}, y \to 0\right\}, \{x \to 0, y \to 0\}\right\}$$

Hemos obtenido, 4 soluciones, veamos cuáles verifican la restricción g < 0:

In[•]:=
```
g[-2/3, -4/3]
```

Out[•]=
$$\frac{32}{9}$$

In[•]:=
```
g[-2/3, 0]
```

Out[•]=
$$\frac{16}{9}$$

In[•]:=
```
g[0, -4/3]
```

Out[•]=
$$\frac{16}{9}$$

In[•]:=
```
g[0, 0]
```

Out[•]=
```
0
```

Resulta que ninguna de ellas verifica la restricción, por tanto no hay que analizar ningún punto crítico.

2) Optimización con restricciones de igualdad.

In[•]:=
```
L[x_, y_, λ_] := f[x, y] - λ g[x, y]
```

In[•]:=
```
GL[x_, y_, λ_] = Grad[L[x, y, λ], {x, y, λ}]
```
gradiente

Out[•]=
$$\left\{2x + 3x^2 - 2(-1 + x)\lambda, 4y + 3y^2 - 2y\lambda, 1 - (-1 + x)^2 - y^2\right\}$$

In[•]:=
```
Solve[GL[x, y, λ] == {0, 0, 0}, {x, y, λ}, Reals]
```
resuelve números r

Out[•]=
$$\{\{x \to 0, y \to 0, \lambda \to 0\}, \{x \to 2, y \to 0, \lambda \to 8\}\}$$

Clasificamos los puntos críticos:

In[•]:=
```
HL[x_, y_, λ_] = HessianH[L[x, y, λ], {x, y}]
```

Out[•]=
$$\{\{2 + 6x - 2\lambda, 0\}, \{0, 4 + 6y - 2\lambda\}\}$$

In[ ]:=
```
Eigenvalues[HL[0, 0, 0]]
```
autovalores

Out[ ]=
```
{4, 2}
```

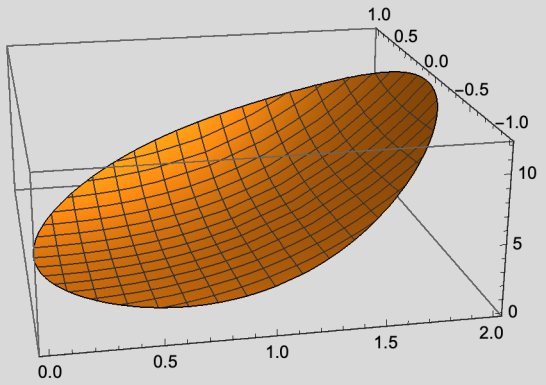
In[ ]:=
```
Eigenvalues[HL[2, 0, 8]]
```
autovalores

Out[ ]=
```
{-12, -2}
```

De donde deducimos que (0,0) es mínimo local condicionado y (-12,-2) es máximo local condicionado. Gráficamente:

In[ ]:=
```
Plot3D[f[x, y], {x, -1, 3}, {y, -1, 1},
 RegionFunction → Function[{x, y, z}, g[x, y] ≤ 0]]
```
representación gráfica 3D; función de región; función

Out[ ]=



Nota. Como veremos en la última sección, el cálculo realizado en la última parte del ejemplo anterior es innecesario. Dado que la restricción delimita un compacto (se trata de un círculo) el teorema de Weierstrass asegura la existencia de máximo y mínimo. Puesto que sólo se han obtenido dos puntos críticos forzosamente deben ser los dos cuya existencia asegura el teorema de Weierstrass. Por tanto, una simple evaluación habría determinado su naturaleza sin ser preciso el cálculo del hessiano.

Observa que si se hubieran obtenido más de dos puntos críticos es preciso utilizar las CSO para determinar el tipo de extremo.

8.2 Extremos absolutos

Por último, estudiamos el cálculo de los extremos absolutos de una función $f(x_1,x_2,\ldots,x_n)$ sujeta a la restricción $\varphi(x_1,x_2,\ldots,x_n)\leq 0$. En este caso, si la restricción $\varphi(x_1,x_2,\ldots,x_n)\leq 0$ delimita un compacto se procede igual que se hizo en la sección 7.3. Es decir, se calculan los puntos críticos del problema en el interior:

$$\begin{aligned} &\text{Optimizar } f(x_1,x_2,\ldots,x_n)\\ &\text{s.a.} \qquad \varphi(x_1,x_2,\ldots,x_n) < 0 \end{aligned}$$

y los puntos críticos del problema en la frontera:

$$\begin{aligned} &\text{Optimizar } f(x_1,x_2,\ldots,x_n)\\ &\text{s.a.} \qquad \varphi(x_1,x_2,\ldots,x_n)=0 \end{aligned}$$

y después se evalúa en ellos la función $f(x_1,x_2,\ldots,x_n)$ para decidir qué puntos críticos tienen carácter de máximo y mínimo absoluto.

Ejemplo. Nos interesa determinar los extremos absolutos de la función $f(x, y) = x^3 + y^2 - 2xy$ en el recinto $(x-1)^2+(y-1)^2 \leq 1$.

Dado que la restricción delimita un compacto (es una circunferencia de centro (1,1) y radio 1), usando el teorema de Weierstrass basta con calcular los puntos críticos que están en el recinto y evaluar en la función f(x,y) para determinar los extremos.

Procedemos por tanto al cálculo de los puntos críticos:

1) Interior

In[•]:=
```
f[x_, y_] := x^3 + y^2 - 2 x y
```

In[•]:=
```
g[x_, y_] := (x - 1)^2 + (y - 1)^2 - 1
```

In[•]:=
```
Jf[x_, y_] = Grad[f[x, y], {x, y}]
             gradiente
```

Out[•]=
$$\{3x^2 - 2y,\ -2x + 2y\}$$

In[•]:=
```
Solve[Jf[x, y] == {0, 0}, {x, y}]
resuelve
```

Out[•]=
$$\left\{\{x \to 0,\ y \to 0\},\ \left\{x \to \frac{2}{3},\ y \to \frac{2}{3}\right\}\right\}$$

Hemos obtenido, 2 soluciones, veamos cuáles verifican la restricción $g < 0$

In[•]:=
```
g[0, 0]
```

Out[•]=
```
1
```

In[•]:=
```
g[2/3, 2/3]
```

Out[•]=
$$-\frac{7}{9}$$

Resulta que el punto (2/3, 2/3) está situado dentro del dominio de estudio y debe ser considerado.

2) Frontera

```
In[•]:= L[x_, y_, λ_] := f[x, y] - λ g[x, y]
```

```
In[•]:= JL[x_, y_, λ_] = Grad[L[x, y, λ], {x, y, λ}]
                         gradiente
Out[•]= {3 x^2 - 2 y - 2 (-1 + x) λ, -2 x + 2 y - 2 (-1 + y) λ, 1 - (-1 + x)^2 - (-1 + y)^2}
```

```
In[•]:= NSolve[JL[x, y, λ] == {0, 0, 0}, {x, y, λ}, Reals]
         resolvedor numérico                        números r
Out[•]= {{x → 1.97294, y → 0.768942, λ → 5.2108},
         {x → 0.256515, y → 1.66875, λ → 2.11175},
         {x → 1.19343, y → 1.98111, λ → 0.802848},
         {x → 0.33855, y → 0.250011, λ → 0.118054}}
```

Para concluir, basta evaluar la función f en la colección de puntos críticos:

```
In[•]:= f[x, y] /. {x → 2/3, y → 2/3}
Out[•]= -4/27
```

```
In[•]:= f[x, y] /. {x → 1.972939936537698`, y → 0.7689418257450746`}
Out[•]= 5.23677
```

```
In[•]:= f[x, y] /. {x → 1.1934287244061181`, y → 1.9811143300220537`}
Out[•]= 0.895944
```

```
In[•]:= f[x, y] /. {x → 0.25651519439703585`, y → 1.6687528271622682`}
Out[•]= 1.94549
```

```
In[•]:= f[x, y] /. {x → 0.33854992071410883`, y → 0.25001080500270556`}
Out[•]= -0.0679736
```

De donde deducimos que el mínimo absoluto esta en el punto (2/3, 2/3) y el máximo absoluto en el punto: (1,972939944134078, 0,7689417658109).

9. Optimización con *Mathematica*

Para concluir esta práctica, comentamos que *Mathematica* dispone de funciones que permiten el cálculo de máximos y mínimos globales, que no precisan que el usuario conozca o maneje el cálculo diferencial en varias variables. Estas funciones son **Maximize** / **Minimize** y sus análogas numéricas **NMaximize**/ **NMinize**:

In[ ]:=
```
? Maximize
```

Out[ ]=

Symbol

Maximize[f, x] maximizes f with respect to x.
Maximize[f, {x, y, …}] maximizes f with respect to x, y, ….
Maximize[{f, $cons$}, {x, y, …}] maximizes f subject to the constraints $cons$.
Maximize[…, $x \in reg$] constrains x to be in the region reg.
Maximize[…, …, dom] constrains variables to the domain dom, typically Reals or Integers.

In[ ]:=
```
? Minimize
```

Out[ ]=

Symbol

Minimize[f, x] minimizes f with respect to x.
Minimize[f, {x, y, …}] minimizes f with respect to x, y, ….
Minimize[{f, $cons$}, {x, y, …}] minimizes f subject to the constraints $cons$.
Minimize[…, $x \in reg$] constrains x to be in the region reg.
Minimize[…, …, dom] constrains variables to the domain dom, typically Reals or Integers.

In[ ]:=
```
? NMaximize
```

Out[ ]=

Symbol

NMaximize[f, x] maximizes f numerically with respect to x.
NMaximize[f, {x, y, …}] maximizes f numerically with respect to x, y, ….
NMaximize[{f, $cons$}, {x, y, …}] maximizes f numerically subject to the constraints $cons$.
NMaximize[…, $x \in reg$] constrains x to be in the region reg.

In[ ]:=
```
? NMinimize
```

Out[ ]=

Symbol

NMinimize[f, x] minimizes f numerically with respect to x.
NMinimize[f, {x, y, …}] minimizes f numerically with respect to x, y, ….
NMinimize[{f, $cons$}, {x, y, …}] minimizes f numerically subject to the constraints $cons$.
NMinimize[…, $x \in reg$] constrains x to be in the region reg.

Por ejemplo, para resolver el ejercicio de la sección anterior bastaría escribir:

Para el máximo

In[•]:=
```
NMaximize[{x^3 + y^2 - 2 x y,  (x - 1) ^2 + (y - 1) ^2 <= 1}, {x, y}]
```
maximiza aproximadamente

Out[•]=
```
{5.23677, {x → 1.97294, y → 0.768927}}
```

Para el mínimo:

In[•]:=
```
Minimize[{x^3 + y^2 - 2 x y,  (x - 1) ^2 + (y - 1) ^2 <= 1}, {x, y}]
```
minimiza

Out[•]=
$$\left\{-\frac{4}{27}, \left\{x \to \frac{2}{3}, y \to \frac{2}{3}\right\}\right\}$$

Es posible incluir restricciones. En el caso del primer ejemplo de la sección 6.2 escribiríamos:

In[•]:=
```
f[x_, y_, z_] := x^2 + y ^2 + z^2
```

In[•]:=
```
g1[x_, y_, z_] := 2 x + y + z - 2
```

In[•]:=
```
g2[x_, y_, z_] := x - y - 3 z - 4
```

In[•]:=
```
Minimize[{f[x, y, z], g1[x, y, z] == 0 , g2[x, y, z] == 0}, {x, y, z}]
```
minimiza

Out[•]=
$$\left\{\frac{86}{31}, \left\{x \to \frac{44}{31}, y \to \frac{1}{31}, z \to -\frac{27}{31}\right\}\right\}$$

Dado que *Mathematica* calcula extremos globales, podemos deducir que este extremo que fue catalogado en 6.2 como local, es también global (de hecho la optimización convexa diferenciable - que queda fuera de los contenidos de este curso - analiza bajo qué condiciones los extremos locales son globales).

No obstante estas funciones no permiten el cálculo de extremos locales. Obsreva que si tratamos de maximizar la función anterior (no acotada superiormente) obtenemos:

In[•]:=
```
Maximize[{f[x, y, z], g1[x, y, z] == 0 , g2[x, y, z] == 0}, {x, y, z}]
```
maximiza

Maximize: The maximum is not attained at any point satisfying the given constraints.

Out[•]=
```
{∞, {x → Indeterminate, y → Indeterminate, z → Indeterminate}}
```

Otro ejemplo de lo anterior lo obtenemos cuando tratamos de optimizar la función del ejemplo de la sección 6.1 (no acotada, sin extremos globales):

In[•]:=
```
Maximize[x^3 + 3 x^2 y - 2 y^2 (1 + y^2), {x, y}]
```
maximiza

Maximize: The maximum is not attained at any point satisfying the given constraints.

Out[•]=
$$\left\{\infty, \left\{x \to \infty, y \to -\frac{12}{5}\right\}\right\}$$

In[ ]:=
```
Minimize[x^3 + 3 x^2 y - 2 y^2 (1 + y^2), {x, y}]
minimiza
```

Minimize: The minimum is not attained at any point satisfying the given constraints.

Out[ ]=
$$\left\{-\infty, \left\{x \to -\infty, y \to -\frac{12}{5}\right\}\right\}$$

10. Ejemplos.

10.1 Elasticidad de la demanda

En 1696 el estadístico inglés Gregory King proporcionó el primer antecedente de la noción de elasticidad. La *ley de King* determina de forma empírica la elasticidad-precio de la demanda. King estudia el efecto de las cosechas de trigo sobre el precio del trigo. A través de sus observaciones deduce que una mala cosecha de trigo produce un aumento en el precio. Sus cálculos suponen la primera idea de elasticidad, aunque no estableció una definición de elasticidad.

Aunque la idea de elasticidad como cambio proporcional de una variable respecto al cambio de otra variable relacionada ya era conocida, la noción actual de elasticidad fue establecida por primera vez en 1890 por el economista inglés Alfred Marshall en sus *Principios de Economía.*

La **elasticidad** es un concepto fundamental en la teoría económica. Se utiliza en numerosos campos de la Economía: teoría del consumidor (en el estudio de la demanda y los distintos tipos de bienes), teoría de la empresa (en la incidencia de la fiscalidad indirecta y los conceptos marginales), economía del bienestar (en el análisis de la distribución del bienestar teniendo en cuenta los excedentes del consumidor y del productor), etc.

Uno de los conceptos de elasticidad más utilizados es la *elasticidad precio de la demanda* (*elasticidad de la demanda)* que mide la variación porcentual de la demanda con respecto a la variación porcentual del precio. Pero tenemos otros conceptos similares como la *elasticidad renta,* que mide también la variación porcentual de la cantidad demandada, pero ante un cambio porcentual en la renta; la *elasticidad de la oferta,* que mide la variación porcentual de la oferta con respecto a la variación porcentual del precio, etc.

La elasticidad de la demanda mide la sensibilidad de los consumidores ante alteraciones en los precios. La expresión matemática de la elasticidad de la demanda es:

$$E_p^d = \frac{\Delta\,\%\,q}{\Delta\,\%\,p} = \frac{\frac{q_f - q_0}{q_0}}{\frac{p_f - p_0}{p_0}}$$

donde q representa la demanda y p el precio. En el caso general de una función $f(x)$ tendríamos que

$$E\,f(x) = \frac{\Delta\,f(x)}{\Delta\,x}\,\frac{x}{f(x)}$$

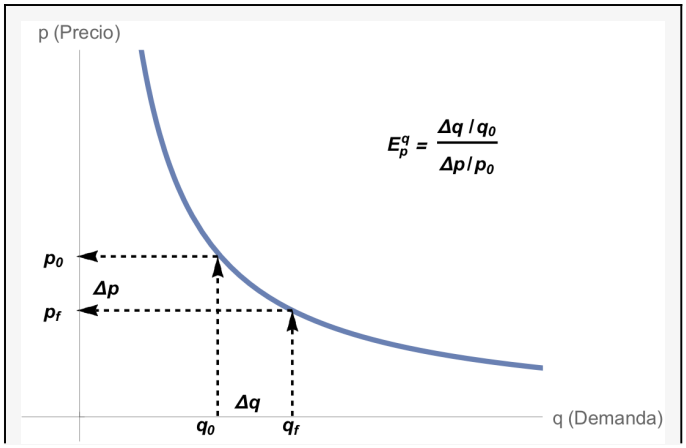
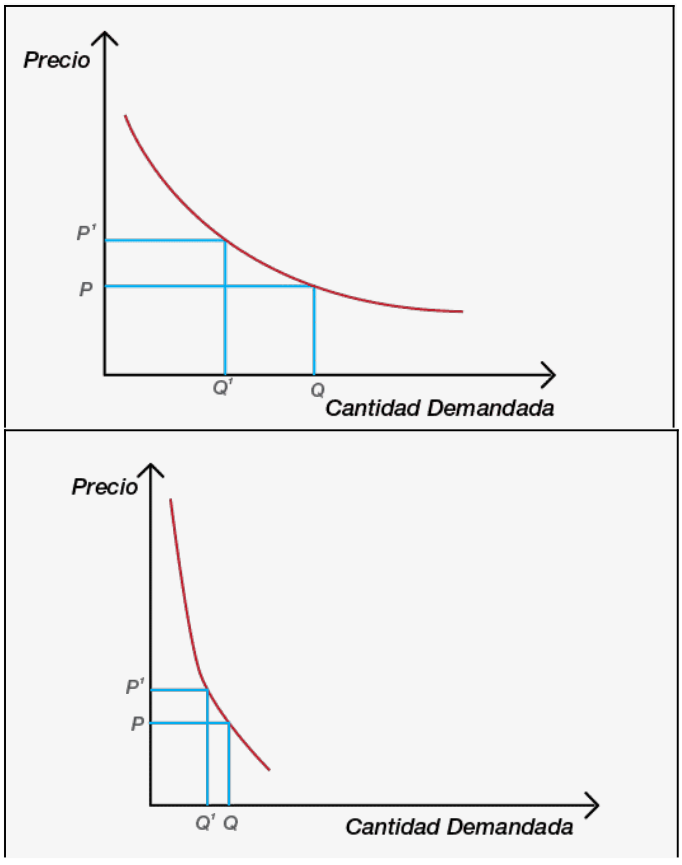


Figura 1: Definición de elasticidad-precio de la demanda.

En una economía de mercado cuando aumenta el precio la demanda disminuye, por tanto, el resultado numérico de la elasticidad precio de la demanda es negativo. Sin embargo, para el análisis se toma habitualmente el valor absoluto. Si la reacción ante una variación pequeña del precio es grande, esto es, si cuando el precio aumenta poco la demanda disminuye en mayor proporción, la elasticidad es grande (en valor absoluto) y se dice que la ***demanda*** es ***elástica***. Si por el contrario la demanda aumenta en proporción menor al aumento de precio hablamos de ***demanda inelástica***. Si ambas, precio y demanda, se incrementan en igual proporción, el valor de la elasticidad es 1.



(a) (b)

Figura 2: (a) *Demanda elástica*: la demanda de un bien es elástica cuando la respuesta de los

consumidores ante el cambio de precio no es significativa ($0 < E < 1$). (b) *Demanda inelástica*: la demanda de un bien es inelástica cuando la respuesta de los consumidores ante el cambio de precio es significativa ($E > 1$).

En la toma de decisiones sobre la fijación de precios es importante conocer el comportamiento de la elasticidad. Cuando la demanda es inelástica se tiene un amplio margen de subida de precios, y que una bajada de precios no serviría de nada, ya que no se tendrían variaciones significativas en la demanda.

En una misma función de demanda la elasticidad, por lo general, va variando a lo largo de la curva, por lo que tiene sentido considerar incrementos infinitesimales del precio. Esto nos lleva a calcular el límite de la expresión de la elasticidad cuando el incremento del precio tiende a 0, obteniendo así la expresión de la elasticidad en cada punto (siempre y cuando ese límite exista): $E = q'(p) \cdot p/q$, donde p y q son, respectivamente, precio y demanda en un determinado momento.

Se define también la *elasticidad arco* para considerar la elasticidad-precio de la demanda entre dos puntos de la curva. Se emplea como referencia el precio y la cantidad media entre esos dos puntos:

$$E^d_{xy} = \frac{\Delta \text{ Variación porcentual en la variable } x}{\Delta \text{ Variación porcentual en la variable } y} = \frac{\frac{x_2 - x_1}{(x_2 - x_1)/2}}{\frac{y_2 - y_1}{(y_2 - y_1)/2}}$$

Aplicación con *Mathematica*

En el siguiente *App* se calculan la **elasticidad** y la **elasticidad arco**. La elasticidad arco mide la variación porcentual de una variable calculada entre dos puntos de una función, de forma que en su cálculo se toma como referencia un punto medio entre los precios inicial y final.

Se ha utilizado la **función de producción de Cobb-Douglas**. Es una función clásica que se utiliza para representar las relaciones entre un producto y las variaciones de los insumos tecnología, trabajo y capital. Su expresión matemática es $\mathbf{Q = A\, T^{\alpha}\, K^{\beta}}$, donde Q es la producción total (es decir, el valor monetario anual de todos los bienes producidos), T es el trabajo insumo, K es el capital insumo, A es el factor total de productividad y las constantes α y β, que están asociadas a la tecnología disponible, son las elasticidades producto del trabajo y el capital.

Este cuaderno permite variar los parámetros de entrada para dos funciones de Cobb-Douglas y ofrece las gráficas de las dos funciones. Además, *Mathematica* calcula y muestra dinámica y visualmente los resultados de la elasticidad. Por otra parte, se puede elegir con la pestaña superior el cálculo de la elasticidad para el caso de elasticidad punto o elasticidad arco.

Estos son los datos iniciales y las funciones auxiliares:

In[2]:=
```
cobbDouglas[Q_, A_] := (Q / (A x^α))^(1/(1-α));

cobbDouglasInv[Q_, A_, y_] := (Q / (A y^(1-α)))^(1/α);

α := 0.5;
elasticidad[C1_, C2_, q1_, a1_, elasticidadpunto_] :=
  If[elasticidadpunto == 2,
   si
   N[((C1 + C2) / (q1-C1/a1 + q1-C2/a1)) ((q1-C1/a1 - q1-C2/a1) / (C2 - C1)), 3],
   valor numérico
   N[(C1 / q1-C1/a1) (1 / -a1), 3]];
   valor numérico

elasticidadpunto = 2;
```

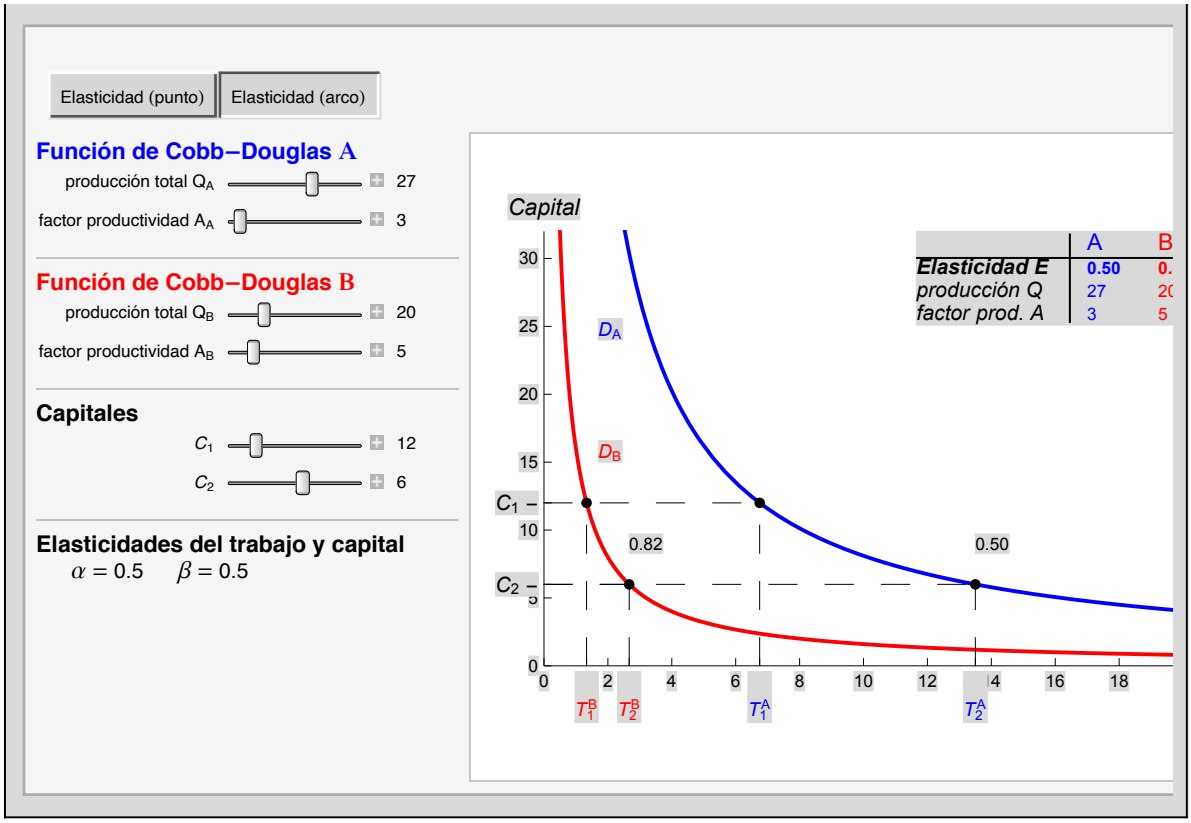
Esta es la función principal con la instrucción **`Manipulate`**:

In[7]:=
```
Manipulate[Show[Plot[{cobbDouglas[q1, a1], cobbDouglas[q2, a2]},
manipula  mues…  representación gráfica
    {x, 0, 20}, PlotRange → {{0, 22}, {0, 32}},
                rango de representación
    AxesLabel → {Style["Trabajo", 12, Italic], Style["Capital", 12, Italic ]},
    etiqueta de ejes  estilo  itálica  estilo  itálica
    AxesOrigin → {0, 0},
    origen de ejes
    Ticks → {Union@Flatten[Map[Append[Table[i, {i, 0, 20, 2}], #] &,
    marcas  unión  aplana  apl…  añade  tabla
          {{cobbDouglasInv[q1, a1, C1], Style["\nT1B", 10, Red]},
                                        estilo  rojo
           {cobbDouglasInv[q2, a2, C2], Style["\nT2A", 10, Blue]},
                                        estilo  azul
           {cobbDouglasInv[q2, a2, C1], Style["\nT1A", 10, Blue]},
                                        estilo  azul
           {cobbDouglasInv[q1, a1, C2], Style["\nT2B", 10, Red]}}], 1],
                                        estilo  rojo
      Append[Append[Table[i, {i, 0, 30, 5}], {C1, Style["C1 -", 11]}],
      añade  añade  tabla                    estilo
       {C2, Style["C2 -", 11]}]},
            estilo
    PlotStyle → {Directive[Thick, Red], Directive[Thick, Blue]},
                directiva  grueso  rojo  directiva  grueso  azul
    Epilog → {{Inset[TableForm[{
                encarte  forma de tabla
          {Style[NumberForm[elasticidad[C1, C2, q2, a2, elasticidadpunto], 2],
           estilo  forma de número
            Blue, Bold], Style[NumberForm[
                 negrita  estilo  forma de número
             elasticidad[C1, C2, q1, a1, elasticidadpunto], 2], Red, Bold]},
                                                    rojo  negrita
          {Style[Style[NumberForm[q2, 2], Blue], Red],
           estilo  estilo  forma de número  azul  rojo
           Style[NumberForm[q1, 2], Red]},
                 forma de número  rojo
```

```
          {Style[Style[NumberForm[a2, 2], Blue], Red],
           Style[NumberForm[a1, 2], Red]}}, TableHeadings →
          {{Style["Elasticidad E", 11, Italic, Bold], Style["producción Q",
            11, Italic], Style["factor prod. A", 11, Italic]},
           {Style["A", 12, Blue], Style["B", 12, Red]}}], {16, 32}, {0, 1}]}},
  ImageSize → 1.1 {400, 280}], Graphics[{{Dashing[.04],
    Line[{{0, C1},
      {Max[cobbDouglasInv[q1, a1, C1], cobbDouglasInv[q2, a2, C1]], C1}}],
    Line[{{cobbDouglasInv[q1, a1, C1], 0},
      {cobbDouglasInv[q1, a1, C1], C1}}],
    Line[{{0, C2}, {cobbDouglasInv[q2, a2, C2], C2}}], Line[
     {{cobbDouglasInv[q2, a2, C2], 0}, {cobbDouglasInv[q2, a2, C2], C2}}],
    Line[{{cobbDouglasInv[q1, a1, C2], 0},
      {cobbDouglasInv[q1, a1, C2], C2}}], Line[
     {{cobbDouglasInv[q2, a2, C1], 0}, {cobbDouglasInv[q2, a2, C1], C1}}],
    Line[{{0, C2}, {cobbDouglasInv[q1, a1, C2], C2}}]},
   {PointSize[.015], Point[{cobbDouglasInv[q1, a1, C1], C1}],
    Point[{cobbDouglasInv[q2, a2, C2], C2}],
    Point[{cobbDouglasInv[q1, a1, C2], C2}],
    Point[{cobbDouglasInv[q2, a2, C1], C1}]},
   {Inset[NumberForm[elasticidad[C1, C2, q1, a1, elasticidadpunto], 2],
     {Max[cobbDouglasInv[q1, a1, C1], cobbDouglasInv[q1, a1, C2]],
      .5 (C1 + C2)}, {-1, 0}],
    Inset[NumberForm[elasticidad[C1, C2, q2, a2, elasticidadpunto], 2],
     {Max[cobbDouglasInv[q2, a2, C1], cobbDouglasInv[q2, a2, C2]],
      .5 (C1 + C2)}, {-1, 0}]},
   {Text[Style["D_A", Blue, 10], {1.7, Min[31, q2 - a2]}, {-1, -1}],
    Text[Style["D_B", Red, 10], {1.7, Min[25, q1 - a1]}, {-1, -1}]}]],
 {{elasticidadpunto, 1, ""}, {1 → "Elasticidad (punto)",
```

```
  2 → "Elasticidad (arco)"},
 ControlPlacement → Top},
                    arriba

Style["Función de Cobb-Douglas A", Blue, Bold, 12],
estilo                              azul  negrita

{{q2, 27, "producción total QA"}, 3,
 40, .01, Appearance -> "Labeled", ImageSize → Tiny},
          apariencia          etiquetado  tamaño de ima··· tamaño minúsculo

{{a2, 3, "factor productividad AA"}, 3, 20, .01,
 Appearance -> "Labeled", ImageSize → Tiny},
                 etiquetado  tamaño de ima··· tamaño minúsculo

Delimiter,
delimitador

Style["Función de Cobb-Douglas B", Red, Bold, 12],
estilo                              rojo  negrita

{{q1, 20, "producción total QB"}, 3, 80, .01, Appearance -> "Labeled",
                                                apariencia          etiquetado

 ImageSize → Tiny}, {{a1, 5, "factor productividad AB"},
             tamaño minúsculo

 3, 20, .01, Appearance -> "Labeled", ImageSize → Tiny},
             apariencia          etiquetado  tamaño de ima··· tamaño minúsculo

Delimiter,
delimitador

Style["Capitales", 12, Bold],
estilo                  negrita

{{C1, 12, Subscript[Style["C", Italic], 1]},
          subíndice  estilo  c···  itálica

 10, 25, .01, Appearance -> "Labeled", ImageSize → Tiny},
              apariencia          etiquetado  tamaño de ima··· tamaño minúsculo

{{C2, 6, Subscript[Style["C", Italic], 2]}, 2, 9, .01,
         subíndice  estilo  c···  itálica

 Appearance -> "Labeled", ImageSize → Tiny},
 apariencia          etiquetado  tamaño de ima··· tamaño minúsculo


Delimiter,
delimitador


Style[ "Elasticidades del trabajo y capital", 12, Bold],
estilo                                                negrita

Style[ "      α = " <> ToString[α] <> "      β = " <> ToString[1 - α], 11],
estilo                  convierte a cadena de caracteres          convierte a cadena de caracteres


SaveDefinitions → True,
guarda definiciones  verdadero

ControlPlacement → Left]
posicionamiento de control  izquierda
```

Out[7]=



Bibliografía

10.2. Modelo de cartera de Markowitz

Harry Markowitz ha sido uno de los pioneros en el enfoque cuantitativo de las finanzas. Recibió el Premio Nobel de Ciencias Económicas en 1990 conjuntamente con Merton Miller y William Sharpe. Markovitz propuso en 1952 considerar el riesgo junto con la rentabilidad en la gestión de carteras de inversión. En su trabajo también resalta la importancia de la diversificación del riesgo en un portafolio de inversión.

Antes del planteamiento de **Markowitz**, los inversores establecían sus estrategias de formación de carteras a través de la maximización de la rentabilidad de las inversiones. El modelo de Markowitz pretende elegir de forma óptima para cada inversor los activos de la cartera de inversión en términos de rentabilidad y riesgo.

En la teoría moderna de formación de carteras se pretende determinar el conjunto de carteras eficientes, la actitud del inversor frente al riesgo y, finalmente, la cartera de inversiones óptima. La hipótesis inicial fundamental es que el inversor siempre prefiere la cartera con mayor rentabilidad y menor riesgo. La rentabilidad de una cartera se mide con la media o esperanza matemática. La volatilidad se utiliza para cuantificar el riesgo de la cartera mediante la varianza o la desviación típica.

Carteras eficientes

Una **cartera eficiente** es una cartera con el riesgo mínimo para un valor de rentabilidad esperado. La frontera eficiente es el conjunto formado por todas las carteras eficientes. Como vemos en la

Figura 1, para aumentar la rentabilidad se debe aumentar el riesgo. Desde el punto de vista matemático, el cálculo de la frontera eficiente se convierte en el problema de maximizar $\overline{R}_p = \sum_{i=1}^{n} X_i \overline{R}_i$ sujeto a la a la condición de no negatividad y a las restricciones presupuestaria y paramétrica:

- *Condición de no negatividad*: Los pesos de la cartera no deben ser negativos, es decir, no se puede realizar venta en corto. Matemáticamente, $X_i, X_2, \ldots X_n \geq 0$.
- *Restricción presupuestaria*: La suma de todos los pesos de la cartera no es superior a uno (se consideran el tanto por uno, en lugar del %), es decir, la cantidad invertida en la cartera no debe ser mayor al presupuesto de la cartera. Matemáticamente, $\sum_{i=1}^{n} X_i \leq 1$.
- *Restricción paramétrica*: La suma de todos los pesos de cada elemento de la cartera multiplicados por las correspondientes covarianzas es igual a la varianza estimada de la cartera. El riesgo se mide mediante la varianza de su rendimiento esperado. Matemáticamente, $\sigma_p = \sum_{i=1}^{n} \sum_{j=1}^{n} X_i X_j \sigma_{i,j}$, donde $\sigma_{i,j}$ es la covarianza que representa como el activo *i* y el activo *j* se mueven de forma conjunta.

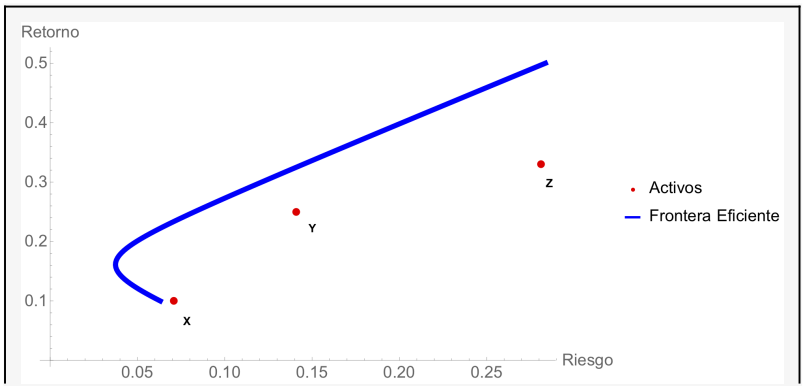


Figura 1: Frontera eficiente de la cartera formada por tres activos. Las cartera localizadas sobre la curva ofrecen el mínimo riesgo para el retorno fijado.

Riesgo y cartera óptima

Las curvas de indiferencia muestran las preferencias del inversor de un portafolio, es decir, su actitud frente al riesgo. Cada inversor establece un umbral de riesgo particular y una rentabilidad mínima para cada nivel de riesgo. De esta forma, la cartera óptima de un inversor concreto viene dada por el punto tangente entre una de sus curvas de indiferencia y la frontera eficiente. No serán posibles las curvas por encima de ese punto y la satisfacción será menor para las que están por debajo.

Aplicación con *Mathematica*

En esta sección aplicaremos el modelo de Markowitz para determinar la **cartera óptima** y la **frontera eficiente** en el caso de que no exista restricción de positividad, es decir, que admite operaciones en corto. Al final de la práctica se muestran algunas funciones que proporciona *Mathematica* para la obtención de dividendos en distintas bolsas y su manipulación para conseguir los valores necesarios para la aplicación del modelo de Markowitz.

- **Modelo de Markowitz sin restricciones de positividad**

En este apartado se construyen en primer lugar las funciones que permiten el cálculo de la cartera de mínima varianza para un determinado retorno cuando se admiten operaciones en corto. Después se calcula la frontera eficiente y la cartera de inversiones óptima con las herramientas gráficas de *Mathematica*.

Cartera de mínima varianza para un determinado retorno

Se define una función **Optimportfolio** de *Mathematica* que calcula la cartera de varianza mínima para un determinado retorno cuando se admiten operaciones en corto. Las entradas de esta función son los retornos de los activos, la matriz de varianzas y el retorno esperado. Las salidas son el vector de pesos de la cartera y su riesgo. A continuación, se muestra el uso de la función con un ejercicio concreto.

Estos son los datos iniciales (retornos de los activos, matriz de varianzas y retorno esperado):

In[8]:=
```
returns = {0.1, 0.25, 0.33};
variance = {{0.005, -0.006, 0.002},
   {-0.006, 0.0199, -0.0158}, {0.002, -0.0158, 0.0790}};
expectedReturn = 0.2;
```

Esta es la función **OptimportfolioI**:

In[11]:=
```
OptimPorfolio[returns_, variance_, expectedReturn_] :=
 Module[{n, i, invV, ones, A, B, C, M, Lambda1, Lambda2, x, sigma2, sigma},
  n = Length[returns];
  ones = Table[1, {i, 1, n}];
  invV = Inverse[variance];
  A = ones.invV.ones;
  B = ones.invV.returns;
  C = returns.invV.returns;
  M = {{A, B}, {B, C}};
  Lambda1 = 1 / Det[M] (C invV.ones - B invV.returns);
  Lambda2 = 1 / Det[M] (A invV.returns - B invV.ones);
  x = Lambda1 + Lambda2 expectedReturn;
  sigma2 = 1 / Det[M] (A expectedReturn^2 - 2 B expectedReturn + C);
  sigma = Sqrt[sigma2];
  Return[{x, sigma}];
 ]
```

Y estos son los resultados de aplicar la función (vector de pesos de la cartera y riesgo):

In[12]:=
```
{px, psigma} = OptimPorfolio[returns, variance, expectedReturn]
```

Out[12]=
```
{{0.414031, 0.434662, 0.151307}, 0.0493784}
```

Frontera eficiente

La frontera eficiente asociada a una cartera recibe como entradas los retornos de los activos, la matriz de varianzas y el rango de retornos. Esta función devuelve los pesos y riesgos como salida y también el gráfico correspondiente. A continuación, se muestra el uso de la función con un ejercicio concreto con la representación gráfica de la frontera eficiente.

Estos son los datos iniciales (retornos de los activos, matriz de varianzas y rango de retornos):

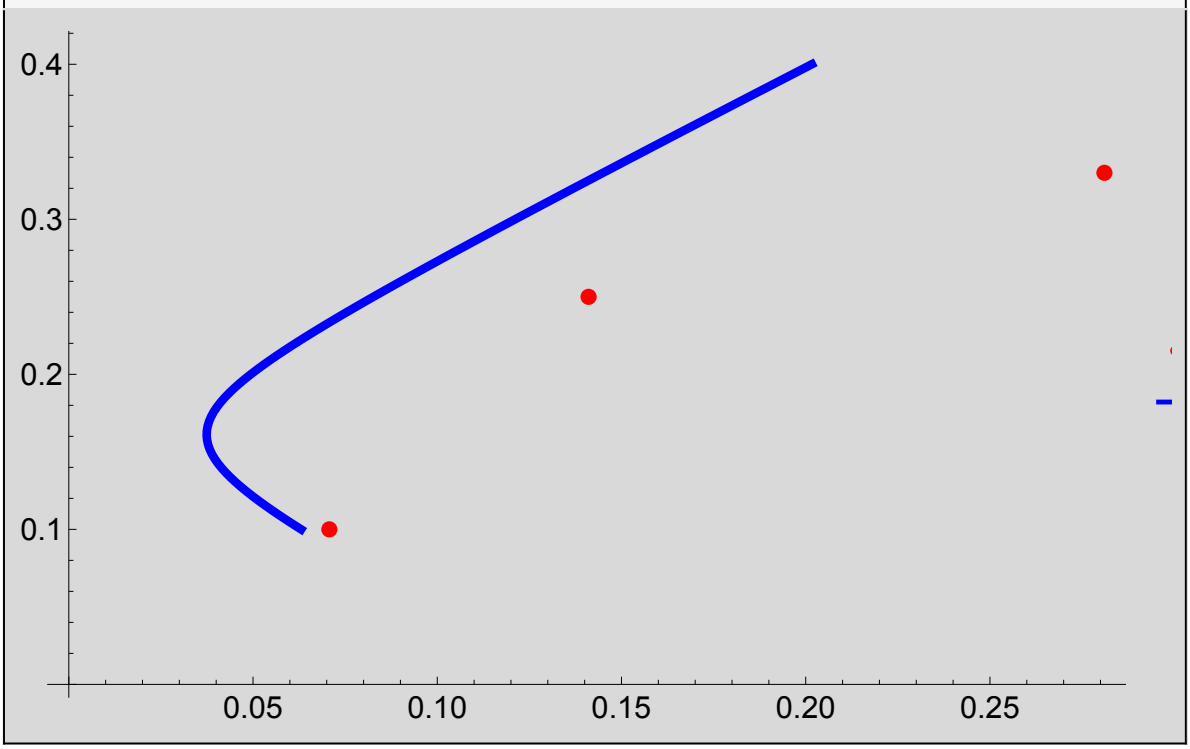
In[13]:=
```
returns = {0.1, 0.25, 0.33};
variance = {{0.005, -0.006, 0.002},
   {-0.006, 0.0199, -0.0158}, {0.002, -0.0158, 0.0790}};
range = {0.1, 0.40};
```

Esta es la función **effPorfolioFrontier**:

In[16]:=
```
effPorfolioFrontier[returns_, variance_, range_] :=
 Module[{np, n, i, expectedReturn, pX, pSigma, effG},
 módulo
  np = 100;
  expectedReturn =
   Table[(1. - i) range[[1]] + i range[[2]], {i, 0, 1, 1. / (np - 1)}];
   tabla
  n = Length[returns];
      longitud
  pX = Table[0, {i, 1, np}];
       tabla
  pSigma = Table[0, {i, 1, np}];
           tabla
  For[i = 1, i ≤ np, i = i + 1,
  para cada
   {pX[[i]], pSigma[[i]]} =
     OptimPorfolio[returns, variance, expectedReturn[[i]]];
  ];
  ListPlot[{Transpose[{Sqrt[Diagonal[variance]], returns}],
  representaci··· transposición raíz ··· diagonal
    Transpose[{pSigma, expectedReturn}]}, Joined → {False, True},
    transposición unido falso verdadero
   PlotStyle → {{Red, PointSize[0.015]}, {Blue, Thickness[0.008]}},
   estilo de represen··· rojo tamaño de punto azul grosor
   LabelStyle → Directive[Black, FontSize → 16],
   estilo de etiqueta directiva negro tamaño de tipo de letra
   PlotLegends → {"Activos", "Frontera Eficiente"}, ImageSize → Large]
   leyendas de representación tamaño de im··· grande
  (*Return[{pX, pSigma,effG}]*)
    retorna
 ]
```

In[17]:=
```
effPorfolioFrontier[returns, variance, range]
```

Out[17]=



■ Funciones auxiliares con Mathematica para la obtención de datos financieros

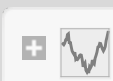
En este apartado se muestran unos ejercicios de cálculo con *Mathematica* del retorno y las volatilidades de los activos con datos financieros reales. Se utilizan las facilidades que proporciona *Mathematica*. En este sentido, la función **FinancialData** es especialmente importante para obtener en distintas bolsas los valores (e.g., precio, dividendo, volatilidad, retorno, etc.) al cierre de cotización para un rango de fechas.

Por ejemplo, recuperamos ahora los datos al cierre de cotización en el NASDAQ del año 2019:

In[59]:=
```
FinancialData["NASDAQ:GOOGL", "Name"]
```
datos financieros

Out[59]=
```
Alphabet Class A Shares
```

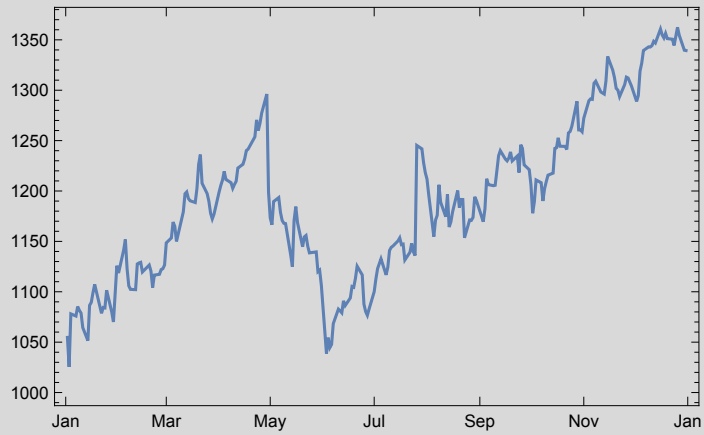
In[62]:=
```
dataGoogle =
 FinancialData["NASDAQ:GOOGL", "Close", {{2019, 1, 1}, {2019, 12, 31}}]
```
datos financieros — cierra

Out[62]=
```
TimeSeries[  ]
```
Time: 02 Jan 2019 to 31 Dec 2019
Data points: 251

Los datos proporcionados incorporan la fecha y el valor y se pueden representar con la función **DateListPlot**:

In[63]:=
```
DateListPlot[dataGoogle, Joined → True]
```
representación de lista respecto a f··· unido verdade

Out[63]=



Para terminar esta sección, comentamos que la función **FinancialData**, también permite recuperar valores como precio, dividendo, volatilidad, retorno para un rango de fechas, etc. Por ejemplo, se obtiene el dividendo para una acción de Apple en NASDAQ, la bolsa de valores electrónica automatizada más grande de Estados Unidos, con la instrucción: **FinancialData["NASDAQ:AAPL", "DividendYield"]** y la matriz de retornos para un rango de fechas también se calcula fácilmente:

In[45]:=
```
FinancialData["NASDAQ:AAPL", "Return", {{2018, 2, 1}, {2018, 2, 14}}][
  "DatePath"] // TableForm
```
datos financieros retorna
forma de tabla

Out[45]//TableForm=

Fri 2 Feb 2018 00:00:00 GMT+2.	-4.33901%
Mon 5 Feb 2018 00:00:00 GMT+2.	-2.49844%
Tue 6 Feb 2018 00:00:00 GMT+2.	4.17918%
Wed 7 Feb 2018 00:00:00 GMT+2.	-2.14071%
Thu 8 Feb 2018 00:00:00 GMT+2.	-2.75166%
Fri 9 Feb 2018 00:00:00 GMT+2.	1.21818%
Mon 12 Feb 2018 00:00:00 GMT+2.	4.02788%
Tue 13 Feb 2018 00:00:00 GMT+2.	1.00178%
Wed 14 Feb 2018 00:00:00 GMT+2.	1.84374%

■ Cálculo de retorno y riesgo para una cartera

La siguiente función **GetReturn** permite recuperar los retornos y matriz de varianzas para una serie de activos utilizando la función **FinancialData** de *Mathematica*. La función toma como argumentos los nombres de compañías y las fechas inicial y final:

In[50]:=
```
GetReturn[companies_, initialDate_, finalDate_] :=
 Module[{i, nc, ExpReturn, Variance, Data},
  nc = Length[companies];
  Data = Table[0, {i, 1, nc}];
  ExpReturn = Table[0, {i, 1, nc}];
  For[i = 1, i ≤ nc, i = i + 1,
   Print[i, ".Importing data...   ", companies[[i]]];
   Data[[i]] = 20 * Normal[ FinancialData[companies[[i]],
         "Return", {initialDate, finalDate}]["Values"]];
   ExpReturn[[i]] = Mean[Data[[i]]];
  ];
  Variance = Covariance[Transpose[Data]];
  Return[{ExpReturn, Variance}];
 ]
```

Y se muestra aquí un ejemplo de ejecución:

In[51]:=
```
companies = {"NASDAQ:AAPL", "NYSE:KO", "NYSE:BBVA"};
initialDate = {2019, 1, 1};
finalDate =  {2019, 3, 31};
```

In[54]:=
```
GetReturn[companies, initialDate, finalDate]
```

```
1.Importing data...   NASDAQ:AAPL
2.Importing data...   NYSE:KO
3.Importing data...   NYSE:BBVA
```

Out[54]=
```
{{0.068559, 0.00454161, 0.0283045}, {{0.176201, 0.0163544, 0.0431085},
  {0.0163544, 0.0823669, 0.00945443}, {0.0431085, 0.00945443, 0.100771}}}
```

Bibliografía

- Hal Varian. *A portfolio of Nobel laureates: Markowitz, Miller and Sharpe*. Journal of Economic Perspectives, **7**(1):159–169, 1993.
- Mark Rubinstein. *Markowitz's "portfolio selection": A fifty-year retrospective.* The Journal of Finance, **57**(3):1041-1045, 2002.
- Alaitz Mendizábal Zubeldia, Luis María Miera Zabalza, Marian Zubia Zubiaurre. *El modelo de Markowitz en la gestión de carteras.* Cuadernos de gestión, Instituto de Economía Aplicada a la Empresa de la Universidad del País Vasco, **2**(1):33-48, 2002.

- José Manuel Cascón Barbero. *Informática para la Optimización*. Gredos, Universidad de Salamanca, 2018. Accesible en: http://hdl.handle.net/10366/136303

11. Ejercicios

1. Prueba que los límites reiterados de la función $f(x, y) = \frac{x^2-y^2}{x^2+y^2}$ en el punto (0,0) son distintos.

2. Calcula los límites reiterados en (0,0) de las siguientes funciones. Si estos son iguales, calcula sus límites direccionales en (0,0) y, si estos son iguales, los límites según las curvas $y = mx^2$:

- $f(x, y) = \frac{x-y}{x+y}$
- $3yf(x, y) = \frac{x}{\sqrt{x^2+y^2}}$
- $f(x, y) = \frac{x^2 y}{(y-x^2)^2}$
- $f(x, y) = \frac{2xy^2-x^3}{x^2+y^4}$

3. Dada la función $f(x, y) = x^2 - 3y + \log(xy)$:

- Calcula el gradiente de f en el punto (1,2)
- Calcula la derivada de f según el vector (1,1) en el punto (1,2)

4. Sea $f : \mathbb{R}^2 \to \mathbb{R}^3$ la función definida como *f(x,y,z)=(x cos y, x sen y, x cos y sen y)*

- Calcula la diferencial de f en $(\pi, \frac{\pi}{2})$: $df_{(\pi,\frac{\pi}{2})}$
- Calcula $df_{(\pi,\frac{\pi}{2})}(3,2)$
- Calcula la matriz jacobiana de f en el punto (x_0, y_0, z_0): $J_f(x_0, y_0, z_0)$

5. Halla la matriz hessiana de las siguientes funciones en el punto (x,y):

- $f(x, y) = \cos(xy) + e^y$
- $f(x, y) = 3xy + 2y^2$
- $f(x, y) = 3xy^2 - 2y + 5x^2 y^2$

6. Sea $f(x, y) = x^3 y + x^2 y^2 - xy^2 + 3$. Halla el desarrollo de Taylor de grado 2 de f en el punto (1,2).

7. Desarrolla f(x; y; z) = xy – xz + 2 yz – z^2 + xy^2 + xyz en potencias de *x-1, y,* y *z + 2*.

8. Calcular y clasificar los extremos relativos de las siguientes funciones:

- $f(x, y) = x^3 + 3x^2 y - 2y^2(1 + y^2)$
- $f(x, y) = (x^2 + 3y^2) e^{1-x^2-y^2}$
- $f(x, y, z) = x^3 + xz + y^2 + xyz$
- $f(x, y, z) = \log(y^2 + 1) - \cos(xz) - \text{sen}(y^2 z)$

9. Justifica que la función $f(x, y) = 3x - 2y - x^3 - y^2$ tiene un extremo global en *C={(x,y,z):x>0}*, indicando de qué tipo es.

10. Calcula los extremos relativos de las siguientes funciones sujetas a las correspondientes restricciones reduciendo los problemas al cálculo de extremos sin restricciones:

- xy sujeta a la restricción $y + x^3 - x^2 = 1$

- $x^2 + y^2$ sujeta a la restricción $y + x^2 = 1$
- $x+y+z$ sujeta a la restricción $x + y + z = 6$
- xyz sujeta a la restricción $x - +y + z - 1 = 0$

11. Determina los extremos relativos de las siguientes funciones utilizando la técnica de los multiplicadores de Lagrange:

- $x^2 + y^2 - xy$ sujeta a la restricción $x^2 + y^2 = 1$
- $6x - y^2 + xz + 60$ sujeta a la restricción $x^2 + y^2 + z^2 = 6$
- $\frac{1}{x} + \frac{1}{y}$ sujeta a la restricción $x + 4y = 1$
- $x + 4y + 3z$ sujeta a las restricciones $g_1(x, y, z) = xy - 1 = 0,\ g_2(x, y, z) = yz - 1 = 0$
- $x^2 + 2\ y^2 + 6\ z^2 - 4\ xy - 12\ z$ sujeta a las restricciones $g_1(x, y, z) = x - y = 0,\ g_2(x, y, z) = x - z + 3 = 0$

12. Calcula los extremos absolutos de los siguientes problemas:

- $x^3 + y^3 + x^2 + 2y^2$ sujeta a la restricción $z^2 + y^2 \leqslant 1$
- $x^2 - xy$ sujeta a la restricción $x^2 + y^2 - xy \leqslant 1$