Reporte de Investigación
2007-09

Optimización de Funciones Lineales con Restricciones en MATLAB™: Teoría y Ejemplos

Responsables: William Marchena
Carlos Ornelas

Supervisor: Francisco M. González-Longatt

Línea de Investigación:
Fuentes Alternas de Energía
Y
Generación Distribuida

27-Feb-2007

Optimización en MATLAB™

La caja de herramientas de optimización o denominado *Toolbox* extiende la capacidad de MATLAB®, acercándolo a un ambiente numérico computacional. *Toolbox* incluye rutinas de muchos tipos, incluyendo la optimización [1]:

- Minimización no lineal sin restricciones.
- Minimización no lineal con restricciones, problemas de minimizar y maximizar, y problemas de minimización semi-infinitos.
- Programación cuadrática y lineal.
- No lineal, mínimos cuadrados y curvas adecuadas.
- Solución de ecuaciones de sistemas no lineales.
- Restricciones lineales de mínimos cuadrados.
- Problemas de larga escala.

Funciones de Optimización

Todas las funciones de la caja de herramientas son (*m-files*) de MATLAB, haciendo de MATLAB un instrumento especializado en algoritmos de optimización. Se puede ver de MATLAB los códigos de las funciones usando la presentación [1]:

```
type function_name
```

Se puede extender la capacidad de optimización de la caja de herramientas escribiendo sus propios *m-files*, o usando la caja de herramienta con otras cajas de herramientas, o con MATLAB o Simulink®.

Caja de Herramienta de Optimización GUI

La herramienta de optimización (*optimtool*), es un GUI (*Grafics Unit Interface*), para seleccionar la solución, especificando las opciones de optimización y los problemas corrientes. Se puede definir y modificar los problemas rápidamente con GUI [1].

Usando Funciones de Optimización

Aquí se describirá como se deberá realizar la utilización de cada una de estas funciones de optimización:

Definición de la Función Objetivo

Muchas de las funciones de optimización requieren que se cree una función de MATLAB que calcule la *función objetivo*. La función, en la entrada debe aceptar vectores, y retornar a la salida un escalar de tipo doble.

Existen dos maneras de crear la función objetivo:

1) Se crea una función anónima en la línea de comando. Por ejemplo, si se crea una función anónima para $\mathbf{x}^2$, se coloca:

```
square = @ [(x)]* x.^2;
```

Y se llama a la función de optimización con el cuadrado del primer argumento de la entrada. Se puede usar este método si la función objetivo es relativamente sencilla, y no requiera ser utilizada en una sesión futura de MATLAB.

2) Si se escribe un M-file para la función, por ejemplo, para escribir la función $\mathbf{x}^2$ como un *m-file*, se debe abrir un nuevo archivo en el editor MATLAB y se deberá colocar el siguiente código:

```
function y = square(x)
y = x.^2;
```

Se puede llamar la función de optimización con @ al cuadrado como el primer argumento de entrada. El signo @ crea una función manejable al cuadrado. Este método se emplea si la función objetiva es complicada o si se sospecha que tal función será utilizada en una próxima sesión de MATLAB.

Maximización

Las funciones de optimización *fminbnd, fminsearch, fminunc, fmincon, fgoalattain, fminimax, lsqcurvefit*, y *lsqnonlin* todas realizan la minimización de la función objetivo $f(x)$. La maximización es alcanzada sustituyendo en las rutinas con $-f(x)$. Asimismo para alcanzar la maximización para *quadprog* se sustituye $-H$ y $-f$, y para *linprog* $-f$ [2].

Restricciones Mayores que Cero

La Caja de herramientas de Optimización (Toolbox) asume que las restricciones de desigualdad no lineales son de la forma $C_i(\mathbf{x})\leq 0$. Las restricciones mayores que cero son expresadas como menores que cero multiplicándolas por -1. Por ejemplo, una restricción de la forma $C_i(\mathbf{x})\geq 0$ es equivalente a la restricción $(-C_i(\mathbf{x}))\leq 0$); una restricción de la forma $C_i(\mathbf{x})\geq b$ es equivalente a la restricción $(-C_i(\mathbf{x}) + b)\leq 0$ [2].

Maximización vs Minimización

La optimización funciona en la caja de herramienta minimizando la función objetivo. Para maximizar la función f se aplica una optimización para minimizar la función $-f$. El punto resultante donde el máximo f ocurre también es el punto donde el mínimo de $-f$ ocurre.

Problemas Cubiertos por la Caja de Herramientas

Las siguientes tablas muestran las funciones disponibles para la minimización y maximización de las funciones a utilizar en este trabajo especial de grado, donde la función objetivo es una ecuación lineal con restricciones lineales o no lineales de desigualdad.

Tabla 1. Tabla representativa de las funciones disponibles para la minimización y maximización de las funciones a utilizar en este trabajo especial de grado

Tipo	*Notación*	*Función*
Programación lineal	$\min_{\mathbf{x}} \mathbf{f}^{\mathbf{T}}\mathbf{x}$ tal que $\mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{A_{eq}x} = \mathbf{b_{eq}}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$	Linprog
Minimización de restricciones	$\min_{\mathbf{x}} f(\mathbf{x})$ tal que $c(\mathbf{x}) \leq 0, \quad ceq(\mathbf{x}) = 0$ $\mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{Aeq.x} = \mathbf{beq} \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$	Fmincon

Linprog [1]

Propósito

Resolver problemas de programación lineal.

Ecuación

Encuentra el mínimo de un problema especificado por:

$$\min \mathbf{f}^T\mathbf{x} \quad \text{tal que} \quad \begin{aligned} &\mathbf{A}\mathbf{x} \le \mathbf{b} \\ &\mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq} \\ &\mathbf{lb} \le \mathbf{x} \le \mathbf{ub} \end{aligned} \qquad (1)$$

Donde: **f, x, b, $\mathbf{b}_{eq}$, lb,** y **ub** son vectores y **A** y $\mathbf{A}_{eq}$ son matrices

Sintaxis

```
x = linprog(f,A,b)
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
[x,fval] = linprog(...)
[x,lambda,exitflag] = linprog(...)
[x,lambda,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)
```

Descripción

Linprog soluciona problemas de programación lineal

`x = linprog(f,A,b)`: soluciona min f'***x** tales que **A*x≤b**

`x = linprog(f,A,b,Aeq,beq)`: soluciona el problema descrito inicialmente (1), mientras que además satisface la igualdad de las restricciones **Aeq*x = beq**. Se coloca **A**=[ ] y **b**=[ ] si no existen desigualdades

`x = linprog(f,A,b,Aeq,beq,lb,ub)`: Se definen los limites inferior y superior sobre la variable de diseño en **x**, de modo que la solución este siempre entre el rango **lb ≤ x ≤ ub.** Se coloara **Aeq** = [ ] y **beq** = [ ] si no existen igualdad.

`x = linprog(f,A,b,Aeq,beq,lb,ub,x0)`: fija el punto de partida x0. Esta opción esta solamente disponible con el algoritmo a media escala (la opción de Larga Escala, se fija a *"off"* usando optimización). El defecto del algoritmo a larga escala y del algoritmo simplex es que ignoran los puntos de partida.

`x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)`: Reduce al mínimo con las opciones de optimización especificadas en la estructura *options*. Se utiliza *optimset* para fijar estas opciones.

`[x,fval] = linprog(...)`: Retorna el valor de la función objetivo *fun* en la solución x: fval = f'***x**.

`[x,lambda,exitflag] = linprog(...)`: devuelve un valor *exitflag* que describe la condición de salida.

`[x,lambda,exitflag,output] = linprog(...)`: Devuelve una estructura *output* que contiene información acerca de la optimización.

`[x,fval,exitflag,output,lambda] = linprog(...)`: Devuelve una estructura *lambda* donde contiene los campos de los multiplicadores de Lagrange en la solución **x**. [1]

Entrada de Argumentos

A continuación se muestra una descripción general de los argumentos pasados en linprog [1]:

Tabla 2. Representativa de los argumentos de entrada para las funciones linprog y fmincon.

Nombre de la entrada	Descripción	Usada por las funciones:
Aeq, beq	La matriz **Aeq** y el vector **beq** son, respectivamente, los coeficiente de las restricciones de la inecuaciones lineales y al correspondiente al lado derecho del vector: **Aeq*x = beq**	*fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog*
f	El vector de los coeficientes para el termino lineal en la ecuación lineal *f'****x** o la ecuación cuadrática **x****H****x** + *f'****x**.	*linprog, quadprog*
fun	La función para optimizar. *fun* es una función manipulada por una función *m-file* o por una función anónima.	*fgoalattain, fmincon, fminimax, fseminf, fminbnd, fminsearch, fminunc, fsolve, fzero, lsqcurvefit, lsqnonlin*

Tabla 3. Representativa de los argumentos de entrada para las funciones linprog y fmincon (continuación)

Nombre de la entrada	Descripción	Usada por las funciones:
lb, ub	Limites inferior y superior de vectores (o matrices). Los argumentos son normalmente del mismo tamaño que **x**. Sin embargo, si **lb** tiene menos elementos que **x,** entonces solo el primer elemento de *m* es el límite inferior. Los límites superiores en **ub** se pueden definir de la misma manera. También se pueden especificar variables infinitas usando *–inf* (para los limites superiores). Por ejemplo si **lb**(i) = -*inf*, la variable **x**(i) es el limite inferior.	*fgoalattain, fmincon, fminimax, fseminf, linprog, lsqcurvefit, lsqlin, lsqnonlin, quadraprog*
Nonlcon	La función que calcula las inecuaciones no lineales, las restricciones de ecuaciones e inecuaciones no lineales "evitando variables globales vía funciones anónimas y jerarquizadas"	*fgoalattain, fmincon, fminimax*
A, b	La matriz **A** y el vector **b** son, respectivamente, los coeficientes de las restricciones lineales de desigualdad y el vector correspondiente del lado derecho: **Ax ≤ b**	*fgoalattain, fmincon, fminimax, fseminf, linprog, lsqlin, quadprog*
Options	Una estructura que define las opciones, se utiliza por las funciones de optimización.	Todas las funciones
x0	Punto inicial (es un escalar, vector o matriz) (para *fzero*, x0 también puede ser dos elementos vectoriales representando un intervalo que es conocido para restricciones un cero)	Todas las funciones excepto *fminbnd*

Argumentos de salida

A continuación se muestra una descripción general de los argumentos de salida [1]:

Tabla 4. Representativa de los argumentos de salida de las funciones linprog y fmincon

Nombre de la salida	Descripción	Usada por las funciones:
exitflag	Un número entero que identifica la razón en la que el algoritmo de optimización finalizo. Se puede utilizar el *exitflag* como una herramienta de programación al escribir *m-fi*les como calculo de optimización. A su vez puede mostrar un mensaje que indica porque una optimización finalizo, llamando la función de optimización con el argumento de salida *output* y mostrando en pantalla *output.message*	Todas las funciones
fval	La evaluación de la función objetivo *fun* con la solución **x**	*fgoalattaing, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, linprog, quadprog*
output	Una estructura de salida que contiene información acerca de los resultados de la optimización.	Todas las funciones
x	La solución encontrada por la optimización de la función. Si *exitflag* > 0, entonces **x** es una solución, si no, **x** es el valor de la optimización rutinaria cuando esta es terminada prematuramente.	Todas las funciones

Función exiflag, lambda y output

A continuación se trataran detalles de las funciones específicas de *exitflag Lambda* y *output* [1]:

exitflag

La función converge a una solución **x**

0 Numero de iteraciones excedidas options. MaxIter.
-2 No se encontró ningún punto factible.
-3 El problema es infinito.
-4 El valor NaN fue encontrado durante la ejecución del algoritmo.
-5 Los problemas principales y duales son infactibles.
-6 La dirección de la búsqueda llego a ser demasiado pequeña. Ningún otro progreso puede ser hecho.

lambda

La estructura contiene los multiplicadores de Lagrange en la solución **x** (separado por el tipo de restricción). Los campos de la estructura son:

lower: Limites inferiores **lb**
upper: Limites superiores **ub**
ineqlin: desigualdades lineales
eqlin: igualdades lineales

output

Estructura que contiene la información sobre la optimización. Los campos de la estructura son:

Algorithm: Algoritmo usado
cgiterations: Número de iteraciones conjugadas del gradiente (solamente logaritmos de larga escala)

iterations: Numero de iteraciones
message: Mensaje de salida

Opciones

Son las opciones de optimización usadas por *linprog* Algunas opciones se aplican a todos los algoritmos, y otras son solamente relevantes cuando se usa el algoritmo de larga escala. Se puede utilizar *optimset* para fijar o para cambiar los valores de estos campos en la estructura de opciones, *options*. A continuación se muestran las opciones de optimización utilizadas por las funciones *linprog* y *fmincon* [1]:

Tabla 4. Representativa de las opciones de los programas linprog y fmincon

Nombre de la opción	Descripción	L, M, B	Usado por las funciones:
Diagnostics	Display muestra información acerca de la función a minimizar	B	Todas, excepto *fmindbnd, fminsearch, fzero* y *lsqnonneg*
DiffMaxChange	Máximo cambio en variables para diferencias infinitas	M	*fgoalattain, Fmincon, fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin*
DiffMinChange	Mínimo cambio en variables para diferenciación finita	M	*Fgoalattaing, fmincon,fminimax, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin*
Display	Nivel del display en *'off'* el display no muestra la salida; *'iter'*. Muestra la salida en cada iteración; *'final'* muestra el final de la salida *'notify'* muestra si la función no converge	B	Todas las funciones
FunValCheck	Comprueba si la función objetivo y los valores de restricciones son validos. *'on'* muestra un error cuando la función objetivo o las restricciones devuelven un valor que es *complex*, *NaN* o *Inf*.	B	*fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit, lsqnonlin*
LargeScale	Usa el algoritmo de larga escala si es posible	B	*fmincon, fminunc, fsolve, linprog, lsqcurvefit, lsqlin, lsqnonlin,quadraprog*
MaxFunEvals	Máximo número de evaluaciones realizadas a la función	B	*fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, lsqcurvefit, lsqnonlin*
MaxIter	Máximo numero de iteraciones realizadas	B	Todas las funciones menos *fzero* y *lsqnonneg*
MaxSQPIter	Máximo numero de iteraciones de programación cuadrática secuencial	M	*Fmincon*
OutputFcn	Especifica una o mas usos definidos de las funciones que la función a optimizar visita cada iteración	B	*fgoalattain, fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, fzero, lsqcurvefit y lsqnonlin*
PrecondBandWidth	Amplitud de banda superior, preacondicionado para *PCG*, el ajuste de *'inf'* usa una factorización directa en lugar de *CG*	L	f*mincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog*

Tabla 4. Representativa de las opciones de los programas linprog y fmincon (continuación)

Nombre de la opción	Descripción	L, M, B	Usado por las funciones:
RelLineSrchBnd	Condición de borde	M	*fgoalattain, fmincon, fminimax, fseminf*
RelLineSrchBndDuration	Numero de iteraciones para el cual la condición de borde especificada en *RelLineSrchBnd* debe ser activa	M	*Fgoalattain, fmincon, fminimax, fseminf*
Simplex	Si la función esta en, *'on'* la función utiliza el algoritmo simplex.	M	*Linprog*
TolFun	Terminación de la tolerancia en la función evaluada	B	*bintprog, fgoalattain, fmincon, fminimax, fminsearch, fminunc, fseminf, fsolve, linprog (L siempre), lsqcurvefit, lsqlin (L siempre) lsqnonlin, quadprog (Lsiempre)*
TolPCG	Terminación de la tolerancia en la iteración *PCG*	L	*fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog*
Tolx	Terminación de la tolerancia en **x**	B	Todas las funciones ecepto algoritmos de Media Escala para *linprog, lsqlin* y *quadprog*
TypicalX	Evaluación **x** típica. La longitud del vector es igual al numero de elementos de x0	B	*fgoalattain, fmincon, fminunc, fsolve, lsqcurvefit, lsqlin, lsqnonlin, quadprog*

Algoritmos de Media y Larga Escala

Estas opciones son utilizadas por algoritmos de media y larga escala [1]:

Diagnostics: Imprime la información de diagnostico sobre la función que se reducirá.
Display: Nivel de exhibición, *"off"*: no se muestra ninguna salida. *'iter'* muestra la salida de cada iteración. *'final'*: se muestra la salida final. En este tiempo los trabajos con *'iter'* se realizan con algoritmos de larga escala y algoritmos simplex.
MaxIter: Máximo número de iteraciones permitidas.

Únicamente Algoritmos de Media Escala

Las siguientes opciones son utilizadas por el algoritmo a media escala:

Simplex: Si la función esta en *'on'*, linprog utiliza el algoritmo simplex. El algoritmo simplex utiliza un punto de partida incorporado, sin considerar el punto de partida x0.

Únicamente algoritmos de larga escala

Estas opciones son utilizadas únicamente por el algoritmo de larga escala

TolFun: Tolerancia del valor final de la función

Algoritmo

Optimización a larga escala

El método de larga escala, se basa en *LIPSOL (Solución Lineal del Punto Interior)* la cual es una variable del algoritmo *Mehrotra's predictor-corrector*, un método primal-dual del punto interior. Un numero de pasos previos del proceso ocurren antes de que el algoritmo comience a iterar [1].

Optimización a media escala

Linprog: es un método activo del sistema y es así una variación del método simplex, bien conocido para programación lineal. El algoritmo encuentra una solución factible inicial por la solución de otro problema de programación lineal:

```
options = optimset('LargeScale', 'off', 'Simplex', 'on')
```

Y pasando *options* como un argumento de entrada de *linprog*. El algoritmo simplex retorna una solución óptima del vértice. Es importante tener presente que no se podrá proveer un punto inicial x0 para *linprog*, del método a gran escala o del método a media escala usando el algoritmo del método simplex. En cualquier caso si se coloca x0 como un argumento de entrada, el *linprog* no toma en cuenta x0 y calcula su propio punto inicial para el algoritmo.

Diagnóstico

Optimización a Larga Escala

La primera etapa del algoritmo puede implicar un proceso previo de las restricciones. Varias condiciones posibles pueden hacer que ocurra esto, por ejemplo *linprog* a la salida con un mensaje de infactibilidad. En cada caso, el valor retornado de *exitflag*. Por *linprog* fija a un valor negativo para indicar falla [1].

Si una fila de ceros se detecta en **Aeq**, pero el elemento correspondiente al **beq** no es cero, el mensaje de salida es [1]:

Salida debido a infactibilidad: Toda la fila de ceros de la matriz de las restricciones no tiene un cero en el tamaño correspondiente a la entrada derecha.

Si uno de los elementos de **x** se encuentra para no ser limitado, el mensaje de salida es:

*Salida debido a infactibilidad: La función objetivo f'*x es ilimitada debajo.*

Si una de las filas de **Aeq** tiene solamente un elemento distinto a cero, el valor asociado en **x** se llama una variable *singleton (semifallo)*. En este caso, el valor de esa componente de **x** se puede calcular de **Aeq** y de **beq**. Si el valor calculado viola otra restricción, el mensaje de la salida es:

Salida debido a infactibilidad: Las variables de semifallo en restricciones de igualdad no son factibles.

Si la variable *singleton (semifallo)* puede ser solucionada para resolver la solución viola los límites superiores o inferiores, el mensaje de la salida es:

Salida debido a infactibilidad: Las variables de semifallo en las restricciones de igualdad no están dentro de los límites.

Una vez que el proceso previo haya finalizado la parte iterativa del algoritmo comienza a resolver los criterios hasta que se detiene. Si el residuo crece en lugar de disminuir, o el residuo ni crece ni disminuye, uno de los dos mensajes de culminación se muestra en pantalla respectivamente:

Uno o más de los residuos, banda dual o error total ha crecido más de 100.000 veces que su valor mínimo:

ó

Uno o más residuos, banda dual o error relativo total ha caído.

Después que uno de estos mensajes se muestren, es seguido por uno de los siguientes seis mensajes que indican el dual, el primal, o ambos, aparecen ser infactibles, los mensajes se diferencian deacuerdo a la infactibilidad:

El dual parece ser infactible (y el primal ilimitado). (El primal residual < TolFun.)

El primal parece ser infactible (y el dual ilimitado). (El dual residual < TolFun.)

*El dual parece ser infactible (y el primal ilimitado) desde entonces el dual residual> sqrt (TolFun). (El primal residual <10*TolFun).*

*El primal parece ser infactible (y el dual ilimitado) desde entonces el primal residual> sqrt (TolFun). (El dual residual <10*TolFun).*

El dual parece ser infactible y el primal ilimitado desde entonces el primal objetivo <-1e+10 y el dual objetivo<1e+6.

El primal parece ser infactible y el dual ilimitado desde entonces el dual objetivo> 1e+10 y el primal objetivo>-1e+6. Tanto el primal como el dual parecen ser infactibles.

Observe que, por ejemplo el primal objetivo puede ser ilimitado, y el primal residual que es una medida satisfactoria de la restricción primal, puede ser pequeño.

Optimización a Media Escala

El *linprog*, da una advertencia cuando el problema es infactible. Advirtiendo: las restricciones son demasiado severas; no hay ninguna solución factible [1].

En este caso, el *linprog* produce un resultado que minimiza el peor caso, que es la violación de restricciones. Cuando las restricciones de igualdad son incoherentes, el *linprog* da una advertencia: *Las restricciones de igualdad son demasiado severas; no hay ninguna solución factible* [1].

Las soluciones ilimitadas producen la advertencia: *La solución es ilimitada y es infinita; las restricciones no son tan restrictivas*. En este caso, el *linprog* devuelve un valor **x** que satisface las restricciones [1].

Limitaciones de la optimización a media escala

En este momento, los únicos niveles de despliegue, usando la opción del Despliegue en las opciones, es *'fuera de'* y *'último'*; rendimiento reiterativo que usa *'el iter'* no es disponible [1].

Fmincon

Propósito

Encontrar el mínimo o máximo de una función multivariable con restricciones no lineales (para el calculo de restricciones lineales, de igual manera ve aplicación) [1].

Ecuación

Un modelo de la función que ésta realiza, lo podemos ver en [1]:

$$\min_x f(x):$$

Sujeta a:

$$\begin{aligned} &c(x) \le 0 \\ &ceq(x) = 0 \\ &Ax \le b \\ &Aeqx = beq \\ &lb \le x \le ub \end{aligned} \qquad (2)$$

Donde **x**, **b**, **beq**, **lb**, y **ub** son vectores, **A** y **Aeq** son matrices, *c* y *ceq(x)* son funciones que retornan vectores, y *f(x)* es una función que retorna un escalar. *f(x)*, *c(x)*, y *ceq(x)* pueden ser funciones no lineales (para éste caso trataremos funciones estrictamente lineales) [1].

Sintaxis

```
x = fmincon(fun,x0,A,b)
x = fmincon(fun,x0,A,b,Aeq,beq)
x = fmincon(fun, x0,A,b,Aeq,beq,lb,ub)
x = fmincon(fun, x0,A,b,Aeq,beq,lb,ub,nonlcon)
x = fmincon(fun, x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
[x,fval] = fmincon(...)
[x,fval,exitflag] = fmincon(...)
[x,fval,exitflag,output] = fmincon(...)
[x,fval,exitflag,output,lambda] = fmincon(…)
[x,fval,exitflag,output,lambda,grad] = fmincon(…)
[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(…)
```
[1]

Descripción

fmincon intenta encontrar un mínimo de una función escalar de varias variables que comienzan en una estimación inicial. Esto es generalmente denominado optimización de restricciones no lineales o programación no lineal [1].

`x = fmincon(fun,x0,A,b)`: Evalúa el valor inicial x0 ,y encuentra un mínimo de **x** para la función descrita en *fun* sujeta a las desigualdades lineales **Ax ≤ b**. x0 puede ser un escalar, un vector o una matriz.

`x = fmincon(fun,x0,A,b,Aeq,beq)`: Minimiza *fun* sujeta a las igualdades lineales **Aeq· x = beq**, así como también la de la forma **Ax ≤ b**. se colocara **A** = [ ] y **b** = [ ] si no existen desigualdades.

`x = fmincon(fun, x0,A,b,Aeq,beq,lb,ub`: Aquí se define y se colocan los limites inferior y superior sobre la variable de diseño en **x**, por esta razón la solución siempre estará entre los rangos **lb ≤ x ≤ ub.** Se colocara **Aeq** = [ ] y **beq** = [ ] si no existen igualdades, **lb(i)** = *-Inf* si **x(i)** es el limite inferior, y **ub(i)** = *Inf* si **x(i)** es el limite superior.

`x = fmincon(fun, x0,A,b,Aeq,beq,lb,ub,nonlcon)`: Realiza la minimización sujeta a las desigualdades no lineales *c(x)* o a las igualdades *ceq*(*x*) definida en *nonlcon*. *fmincon* optimiza sujeta a $c(x) \leq 0$ y $ceq(x)=0$. Se colocara **lb** = [ ] y⁄o ***ub*** = [ ] si no existen limites.

`x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)`: Minimiza con las opciones de optimización especificadas en la estructura de opciones. Se utilizara *optimset* para colocar estas opciones. Se coloca *nonlcon* = [ ] si no hay restricciones de igualdad o desigualdad no lineales.
`[x,fval] = fmincon(...)`: Retorna el valor de la función objetivo *fun* en la solución **x**.

`[x,fval,exitflag] = fmincon(...)`: Retorna un valor *exitflag* que describe la condición de salida de *fmincon*.

`[x,fval,exitflag,output] = fmincon(...)`: Retorna una estructura *output* con información acerca de la optimización.

`[x,fval,exitflag,output,lambda] = fmincon(…)`: Retorna una estructura *lamda* cuyos campos contienen los multiplicadores de lagrange en la solución de **x.**

`[x,fval,exitflag,output,lambda,grad] = fmincon(...)`: Retorna el valor del gradiente de *fun* en la solución de **x**.

Función fun, nonlcon, y options

A continuación se proporcionara detalles de la función específica para: *fun, nonlcon y options* [1]:

fun

Es la función a ser minimizada o maximizada. *fun* es una función que acepta un vector **x** y retorna un escalar *f*, que es la función objetivo evaluada en **x**. La función *fun* puede ser especificada como una función *handle* para los archivos *m-file.*

```
x = fmincon(@myfun,x0,A,b)
```

Donde *myfun* es una función de MATLAB como:

```
function f = myfun(x)
f = ... % Compute function value at x
```

fun puede ser también una función *handle* para una función anónima.

```
x = fmincon(@(x)norm(x)^2,x0,A,b);
```

Si el gradiente de *fun* también requiere ser calculado la opción *GradObj* estará en *"on"*, como se puede ver en lo siguiente:

```
options = optimset('GradObj','on')
```

Entonces la función *fun* debe volver, en la segunda salida el argumento, el gradiente evaluado en *g*, un vector en **x**. Debe tenerse presente que por medio de comprobar el valor de *nargout*, la función *g* puede evitar ser calculada cuando la opción *fun* es llamada con solo un argumento de salida (En el caso donde la optimización solo necesite el valor de *f* pero no de *g*).

```
function [f,g] = myfun(x)
f = ... % Compute the function value at x
if nargout > 1 % fun called with two output arguments
g = ... % Compute the gradient evaluated at x
end
```

El gradiente consiste en las derivadas parciales de *f* en el punto **x**. Esto es, la *i*-ésima componente de *g* es la derivada parcial de *f* con respecto a la *i*-ésima componente de **x**.

Nonlcon
Es la función que calcula las restricciones de desigualdad no lineales c(**x**)≤ 0 y las restricciones de igualdad no lineal **ceq**(**x**)=0. La función *nonlcon* acepta un vector **x** y retorna dos vectores **c** y **ceq**. El vector **c** contiene las desigualdades no lineales evaluadas en **x**, y **ceq** contiene las igualdades no lineales evaluadas en **x**. La función *nonlcon* puede ser especificada como una función *handle*.

```
x = fmincon(@myfun,x0,A,b,Aeq,beq,lb,ub,@mycon)
```

Donde *mycon* es una función de MATLAB como:

```
function [c,ceq] = mycon(x)
c = ... % Compute nonlinear inequalities at x
ceq = ... % Compute nonlinear equalities at x
```

Si el gradiente de las restricciones necesita ser calculado la opción *GradConstr* debe estar en *'on'*, como puede observarse en lo siguiente:

```
options = optimset('GradConstr','on')
```

Cuando la opción *nonlcon* deba volver, en los terceros y cuartos argumentos de salida, GC, el gradiente de **c(x)**, y GCeq, el gradiente de **ceq(x)**. Debe tenerse presente que por medio de comprobar el valor de la opción *nargout* la función GC y GCeq puede evitar ser calculada cuando la opción *nonlcon* es llamada con solo dos argumentos de salida (en el caso donde el algoritmo de optimización solo necesite los valores de **c** y **ceq** pero no los de GC y GCeq).

```
c = ... % Nonlinear inequalities at x
ceq = ... % Nonlinear equalities at x
if nargout > 2 % nonlcon called with 4 outputs
GC = ... % Gradients of the inequalities
GCeq = ... % Gradients of the equalities
end
```

Si *nonlcon* retorna un vector **c** de *m* componentes y **x** tiene longitud *n*, donde *n* es la longitud de x_0, cuando el gradiente GC de **c(x)** es una matriz *n* x *m*, donde GC(*i*,*j*) es la derivada parcial de *c*(*j*) con respecto a *x*(*i*), (la *j*-ésima columna de GC es el gradiente de la *j*-ésima restricción de desigualdad *c*(*j*)). De la misma manera, si **ceq**, tiene *p* componentes, el gradiente GCeq de **ceq(x)** es una matriz *n* x *p*, donde GCeq(*i*,*j*) es la derivada parcial de *ceq*(*j*) con respecto a *x*(*i*) (la *j*-ésima columna de GCeq es el gradiente de la *j*-ésima restricción de igualdad *ceq*(*j*)).

Solución de Problemas de Optimización empleando MATLAB™

Ejemplo #1

Minimizar la función $f(x_1, x_2) = 2x_1 + 8x_2$ sometida a las restricciones [3]:

$x_1 \geq 0;$

$x_2 \geq 0;$

$2x_1 + 4x_2 \geq 8;$

$2x_1 - 5x_2 \leq 0;$

$-x_1 + 5x_2 \leq 5;$

Llamando, respectivamente *r, s* y *t* a las rectas expresadas en las tres últimas restricciones, la zona de soluciones factibles sería:

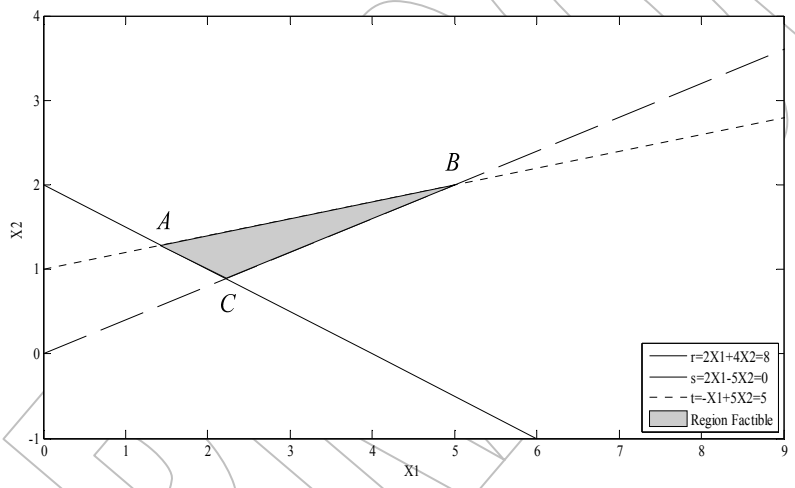

Figura 1. Grafica para obtener los puntos de intersección de las restricciones del ejemplo 1

Siendo los vértices:

A intersección de *r* y *t*:

$$\left.\begin{aligned} 2x_1 + 4x_2 &= 8 \\ -x_1 + 5x_2 &= 5 \end{aligned}\right\} \rightarrow A\left(\frac{10}{7}, \frac{9}{7}\right)$$

B intersección de *s* y *t*:

$$\left.\begin{aligned} 2x_1 - 5x_2 &= 0 \\ -x_1 + 5x_2 &= 5 \end{aligned}\right\} \rightarrow B(5,2)$$

C intersección de *r* y *s*:

$$\left.\begin{aligned} 2x_1 + 4x_2 &= 8 \\ 3x_1 - 5x_2 &= 0 \end{aligned}\right\} \rightarrow C\left(\frac{20}{9}, \frac{8}{9}\right)$$

Siendo los valores de la función objetivo en ellos:

$$f(A) = 2\frac{10}{7} + 8\frac{9}{7} = \frac{92}{7} \approx 13{,}1$$

$$f(B) = 2 * 5 + 8 * 2 = 26$$

$$f(C) = 2\frac{20}{9} + 8\frac{8}{9} = \frac{104}{9} \approx 11{,}5 \text{ minimo}$$

Alcanzándose el mínimo en el punto *C*.

Resolución en MATLAB

Minimizar la función $f(x_1, x_2) = 2x_1 + 8x_2$ sometida a las restricciones:

$x_1 \geq 0;$

$x_2 \geq 0;$

$2x_1 + 4x_2 \geq 8;$

$2x_1 - 5x_2 \leq 0;$

$-x_1 + 5x_2 \leq 5;$

Para resolver el siguiente problema en MATLAB es necesario que todas las restricciones estén de la forma: ≤ 0 para poder crear el archivo *M-file*, esto se logra multiplicando cada restricción por -1 quedando:

$x_1 \leq 0;$

$x_2 \leq 0;$

$-2x_1 - 4x_2 + 8 \leq 0;$

$2x_1 - 5x_2 \leq 0;$

$-x_1 + 5x_2 \leq 5;$

- Luego se crea un archivo *M-file* en MATLAB definiendo la función objetivo; quedando de la siguiente manera:

```
%Creación de un archivo M-file para definir la función objetivo
function f = objfun(x)
f = 2*x(1) +8*x(2);
```

- Otro archivo M-file es creado para definir las restricciones del problema, las cuales todas deben ser ≤ 0, quedando:

```
function [c, ceq] = confun(x)
% restricciones de desigualdades lineales o no lineales
c = [-x(1); -x(2);  -2*x(1)-4*x(2)+8; 2*x(1)-5*x(2); -x(1)+5*x(2)-5];
% restricciones de igualdades lineales o no lineales
ceq = [];
```

Se observa que en **ceq** no se define ninguna inecuación, esto se debe a que todas las restricciones dadas en el problema son de la forma $\mathbf{G}(\mathbf{x}) \leq 0$ o $\mathbf{G}(\mathbf{x}) \geq 0$ pero no de la forma $\mathbf{G}(\mathbf{x}) = 0$.

Luego se crea otro archivo M-file que se encargue de llamar a los otros dos archivos M-file donde se definen la función objetivo y las restricciones del problema, de la siguiente manera:

```
%Programa de ejemplo de utilización de funciones del toolbox de
%optimización en MATLAB
%Realizado por:- Marchena Williams
%               - Ornelas Carlos
%Supervizado por: Gonzalez-Longatt, F.
%Fecha: 12-02-2006
%UNEFA - Núcleo Maracay
%-----------------------------------------------------------------
clc   % Borra la pantalla
clear % Borra todas las variables
disp('  ') % DISP: muestra una cadena de caracteres
disp(' Ejemplo 1')
disp('-----------------------------------------------------------')
disp(' Condiciones Iniciales')
x0=[-1,1]
options=optimset('LargeScale','off');
[x,fval]=fmincon(@objfun,x0,[],[],[],[],[],[],@confun,options)

%-----------------------------------------------------------------
```

Donde se indican los valores iniciales de **x**, se indica que es una función de media escala ya que la de larga escala esta en modo *'off'*, y por ultimo se indican los valores que se desean de la solución, en este caso: *[x,fval]*, y se llama la función objetivo con *objfun* y sus restricciones con *confun*. Dando el siguiente resultado:

```
Optimization terminated: first-order optimality measure less
 than options.TolFun and maximum constraint violation is less
 than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                      3
                                      4


x =

    2.2222    0.8889


fval =

   11.5556
```

Al obtener este resultado con la aplicación de MATLAB y compararlo con el obtenido de forma gráfica, se comprueba de forma consistente que la solución del problema es efectivamente en el punto *C*, que es el punto donde se alcanza el mínimo para la función $f(x_1, x_2) = 2x_1 + 8x_2$.

Ejemplo #2

Determine los valores de x que minimicen la siguiente función*: f(x)=-$x_1x_2x_3$,* Que comienza en el punto: **x**= [10; 10; 10], y sujeta a las siguientes restricciones [1]:

$$0 \le x_1 + 2x_2 + 2x_3 \le 72$$

Este ejemplo es tomado de la optimización toolbox para uso con MATLAB, el cual es un ejemplo resuelto de la siguiente manera:

Primero, escriba un M-file que devuelva un valor escalar *f* de la función evaluada en x

```
Function f = myfun(x)
F = -x(1)*x(2)*x(3);
```

Entonces, reescriba las restricciones como dos ecuaciones, menor que o igual a una constante:

$$-x_1 - 2x_2 - 2x_3 \le 0$$
$$x_1 + 2x_2 + 2x_3 \le 72$$

Puesto que ambas restricciones son lineales, formulándolo como una matriz de desigualdad *A*x = b*, donde:

$$A = \begin{bmatrix} -1 & -2 & -2 \\ 1 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 72 \end{bmatrix}$$

Luego coloque un punto de partida e invoque la optimización:

```
x0 = [10; 10; 10];      % comenzar conjetura de la solución

[x,fval] = fmincon(@myfun,x0,A,b)
```

Después de 66 evaluaciones de la función, la solución es:

```
x =
    24.0000
    12.0000
    12.0000
```

Donde la función evaluada es:

```
fval =
    -3.4560e+03
```

Y las restricciones lineales de desigualdad, se evalúan para ser menor o igual que cero:

```
A*x-b=
          -72
            0
```

Resolución en MATLAB

Determine los valores de *x* que minimicen la siguiente función*: f(x)=-x₁x₂x₃,* Que comienza en el punto: x = [10; 10; 10], y sujeta a las siguientes restricciones:

$0 \le x_1 + 2x_2 + 2x_3 \le 72$

Creación del archivo *M-file* para definir la función objetivo:

```
%Creación de un archivo M-file para definir
%la función objetivo
función f = objfun(x)
f = -x(1)*x(2)*x(3);
```

Luego, creando otro archivo *M-file* se definen las restricciones de la función objetivo

```
%Restricciones son lineales de desigualdad
función [c, ceq] = confun(x)
% restricciones no lineales de desigualdad
c = [-x(1)-2*x(2)-2*x(3);
    x(1)+2*x(2)+2*x(3)-72];
% restricciones no lineales de igualdad
ceq = [];
```

Por último es creado otro archivo *M-file* para definir las variables de entrada, a través de las cuales obtenemos los resultados de la optimización

```
%Programa de ejemplo de utilización de funciones del toolbox de
%optimización en MATLAB
%realizado por:- Marchena Williams
%               - Ornelas Carlos
%Supervizado por: Gonzalez-Longatt, F.
%Fecha: 12-02-2006
%UNEFA - Núcleo Maracay
%-----------------------------------------------------------------
disp('  ')
disp(' Ejemplo 2')
disp('-----------------------------------------------------------')

disp(' Condiciones Iniciales')
x0=[10; 10; 10]
disp('Matrices de Coeficientes de Ax=b')
A=[-1 -2 -2; 1 2 2]
b=[0; 72]
options=optimset('LargeScale','off');
[x,fval]=fmincon(@objfun1,x0,A,b,[],[],[],[],@confun1,options)
```

En este caso, como las restricciones son de igualdad del tipo: **A*x = b**, definiendo las restricciones en forma de **A*x = b**, donde **A** es una matriz y **b** un vector, como se menciona anteriormente, una vez definida A y b, se indica a *fmincon* para poder obtener la optimización, como sigue:

```
Condiciones Iniciales

x0 =

    10
    10
```

```
    10

Matrices de Coeficientes de Ax=b

A =

    -1    -2    -2
     1     2     2


b =

     0
    72


Optimization terminated: magnitude of directional derivative in search
 direction less than 2*options.TolFun and maximum constraint violation
  is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                           2         2
x =

   24.0000
   12.0000
   12.0000


fval =

 -3.4560e+003
```

Al realizar el ejercicio por medio de las aplicaciones de MATLAB se puede observar que la solución alcanzada para el valor de x que reduce al mínimo la función $f(x)=-x_1x_2x_3$, es el punto x= (24,12,12), y donde el valor de la función evaluada en este punto x, es de -3.4560e+003. De esta manera se corrobora el resultado ya arrojado en el cálculo mostrado anteriormente.

Ejemplo #3

Encuentre el valor de x que reduce al mínimo [1]:

$$f(x) = -5x_1 - 4x_2 - 6x_3$$

Sujeto a:

$x_1 - x_2 + x_3 \le 20$

$3x_1 + 2x_2 + 4x_3 \le 42$

$3x_1 + 2x_2 \le 30$

$0 \le x_1, 0 \le x_2, 0 \le x_3$

Primero, introduzca los coeficientes:

```
f = [-5; -4; -6]

A = [1 -1 1
     3 2 4
     3 2 0];

b = [20; 42; 30];

lb = zeros(3,1);
```

Luego, se llama la rutina de programación lineal `[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);`

Introduciendo *x, lambda.ineqlin*, y *lambda.lower*, se coloca:

```
x =
      0.0000
      15.0000
      3.0000
lambda.ineqlin =
      0
      1.5000
      0.5000
lambda.lower =
      1.0000
      0
      0
```

Resolución en MATLAB

Encuentre el valor de *x* que reduce al mínimo:

$$f(x) = -5x_1 - 4x_2 - 6x_3$$

Sujeto a:

$x_1 - x_2 + x_3 \le 20$

$3x_1 + 2x_2 + 4x_3 \le 42$

$3x_1 + 2x_2 \le 30$

$0 \le x_1, 0 \le x_2, 0 \le x_3$

Creación del archivo M-file para definir la función objetivo de la siguiente manera:

```
function f = objfun(x)
f = -5*x(1)-4*x(2)-6*x(3);
```

Luego, creando otro archivo M-file se definen las restricciones de la función objetivo:

```
%Definición de las restricciones, las cuales son ecuaciones
%lineales de desigualdad.
función [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = [x(1)-x(2)+2*x(3)-20; 3*x(1)+2*x(2)+4*x(3)-42;
    3*x(1)+2*x(2)-30; -x(1); -x(2); -x(3)];
% Nonlinear equality constraints
ceq = [];
```

Por último se crea otro archivo *M-file* para definir las variables de entrada, a través de las cuales obtenemos los resultados de la optimización:

```
%Programa de ejemplo de utilización de funciones del toolbox de
%optimización en MATLAB
%realizado por:- Marchena Williams
%               - Ornelas Carlos
%Supervizado por: Gonzalez-Longatt, F.
%Fecha: 12-02-2006
%UNEFA - Núcleo Maracay
%-----------------------------------------------------------------
clc %Borrar la pantalla
clear %Borrar todas las variables
disp(' ') %DISP: muestra una cadena de caracteres
disp(' ejemplo 3')
disp('-----------------------------------------------------------')
disp('Condiciones iniciales')
x0 = [-1,1,1]
options = optimset('LargeScale','off');
f = [-5; -4; -6]
A = [1 -1 1
     3 2 4
     3 2 0];
b = [20; 42; 30];
[x, fval] = fmincon(@objfun2,x0,A,b,[],[],[],[],@confun2,options)
```

Al igual que en el ejercicio anterior, como las restricciones son de igualdad del tipo: **A*x = b**, definiendo las restricciones en forma de **A*x = b**, donde **A** es una matriz y b un vector, como se menciona anteriormente, una vez definida **A** y **b**, se indica a *fmincon* para poder obtener la optimización, como sigue:

```
Ejemplo 3
-----------------------------------------------------------
Condiciones iniciales

x0 =

    -1     1     1


f =

    -5
```

```
    -4
    -6


Optimization terminated: first-order optimality measure less
 than options.TolFun and maximum constraint violation is less
 than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                           2          2
                           3          3
                                      4

x =

         0   15.0000    3.0000
fval =

  -78.0000
```

Al realizar el ejercicio por medio de las aplicaciones de MATLAB se puede observar que la solución alcanzada para el valor de x que reduce al mínimo la función $f(x)= -5x_1-4x_2-6x_3$, es el punto x =(0,15,3), y donde el valor de la función evaluada en este punto x, es de -78. De esta manera se corrobora el resultado ya arrojado en el cálculo mostrado anteriormente.

Ejemplo 4

A una persona le tocan 10 millones de bolívares en una lotería y le aconsejan que las invierta en dos tipos de acciones, *A* y *B*. Las de tipo A tienen más riesgo pero producen un beneficio del 10 %. Las de tipo B son más seguras, pero producen sólo el 7% anual. Después de varias deliberaciones decide invertir como máximo 6 millones en la compra de acciones *A* y por lo menos, 2 millones en la compra de acciones B. Además, decide que lo invertido en *A* sea, por lo menos, igual a lo invertido en *B*. ¿Cómo deberá invertir 10 millones para que le beneficio anual sea máximo? [3]

Sean las variables de decisión:
x_1= cantidad invertida en acciones *A*
x_2= cantidad invertida en acciones *B*

La función objetivo es:

$$f(x) = \frac{10x_1}{100} + \frac{7x_2}{100}$$
$$f(x) = 0.1x_1 + 0.07x_2$$

Y las restricciones son:

$$\left.\begin{array}{l} x_1 \geq 0; x_2 \geq 0 \\ x_1 + x_2 \leq 10 \\ x_1 \leq 6 \\ x_2 \leq 2 \\ x_1 \geq x_2 \end{array}\right\}$$

La zona de soluciones factibles es:

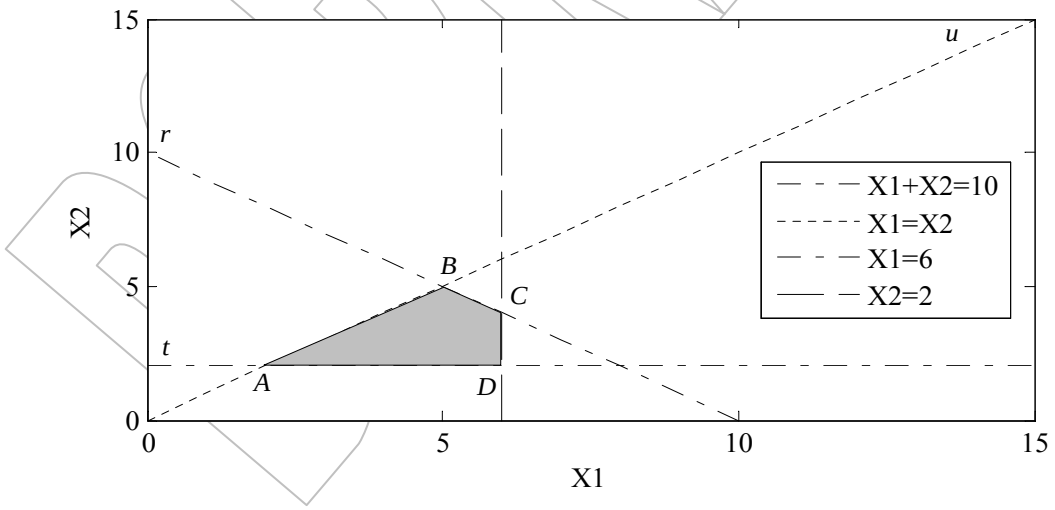

Figura 2. Grafica para obtener los puntos de intersección de las restricciones de la función objetivo del Ejemplo 4

Siendo los vértices del recinto:

A intersección de *u*,*t*:

$$\left.\begin{array}{l} x_1 = x_2 \\ x_2 = 2 \end{array}\right\} \rightarrow A(2,2)$$

B intersección de *r,u*:

$$\left.\begin{array}{l} x_1 + x_2 = 10 \\ x_1 = x_2 \end{array}\right\} \rightarrow B(5,5)$$

C intersección de *r,s*:

$$\left.\begin{array}{l} x_1 + x_2 = 10 \\ x_1 = 6 \end{array}\right\} \rightarrow C(6,4)$$

D intersección se *s,t:*

$$\left.\begin{array}{l} x_1 = 6 \\ x_2 = 2 \end{array}\right\} \rightarrow D(6,2)$$

La función objetivo toma en ellos los valores:

$$f(A) = \frac{20}{100} + \frac{14}{100} = \frac{34}{100} = 0{,}34\text{millones}$$
$$f(B) = \frac{50}{100} + \frac{35}{100} = \frac{85}{100} = 0.85\text{millones}$$
$$f(C) = \frac{60}{100} + \frac{28}{100} = \frac{88}{100} = 0.88\text{millones MAXIMO}$$
$$f(D) = \frac{60}{100} + \frac{14}{100} = \frac{74}{100} = 0{,}74\text{millones}$$

Siendo la solución óptima invertir 6 millones de bolívares en acciones tipo *A* y 4 millones en acciones tipo *B*.

Resolución en MATLAB

La función objetivo es:

$$f(x) = 0.1x_1 + 0.07x_2$$

Y las restricciones son:

$$\left.\begin{array}{l} x_1 \geq 0; x_2 \geq 0 \\ x_1 + x_2 \leq 10 \\ x_1 \leq 6 \\ x_2 \geq 2 \\ x_1 \geq x_2 \end{array}\right\} \rightarrow \left\{\begin{array}{l} -x_1 \leq 0; -x_2 \leq 0 \\ x_1 + x_2 - 10 \leq 0 \\ x_1 - 6 \leq 0 \\ -x_2 + 2 \leq 0 \\ -x_1 + x_2 \leq 0 \end{array}\right.$$

Creación del archivo M-file para definir la función objetivo:

```
%creación de un archivo M-file para definir la función objetivo
function f = objfun(x)
f =-(0.1*x(1) + 0.07*x(2));
```

- Luego, creando otro archivo M-file se definen las restricciones de la función objetivo:

```
%Restricciones de inecuaciones lineales
function [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = [-x(1); -x(2); x(1)+x(2)-10; x(1)-6; -x(2)+2;
    -x(1)+x(2)];
% Nonlinear equality constraints
ceq = [];
```

Por último se crea otro archivo M-file para definir las variables de entrada, a través de las cuales obtenemos los resultados de la optimización:

```
%Programa de ejemplo de utilización de funciones del toolbox de
%optimización en MATLAB
%realizado por:- Marchena Williams
%               - Ornelas Carlos
%supervizor: Francisco M. Gonzalez-Longatt
%Fecha: 12-02-2006
%UNEFA - Núcleo Maracay
%-----------------------------------------------------------------
disp (' ')
disp('Ejemplo 4')
disp('-----------------------------------------------------------------')
disp('condiciones iniciales')
x0=[-1,1]
options=optimset('LargeScale','off');
[x,fval]=fmincon(@objfun3,x0,[],[],[],[],[],[],@confun3,options)
```

Como las restricciones son de igualdad del tipo: **A*x = b**, definiendo las restricciones en forma de **Ax = b**, donde **A** es una matriz y **b** un vector, como se menciona anteriormente, una vez definida **A** y **b**, se indica a `fmincon` para poder obtener la optimización, como sigue:

```
Ejemplo 4
-------------------------------------------------------------
Condiciones iniciales

x0 =

    -1     1

Optimization terminated: first-order optimality measure less
 than options.TolFun and maximum constraint violation is less
 than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
                                      3
                                      4

x =

     6     4


fval =

   -0.8800
```

Al obtener este resultado con la aplicación de MATLAB y compararlo con el obtenido de forma gráfica, se comprueba de forma consistente que la solución del problema es efectivamente invertir 6 millones de bolívares en acciones tipo *A* y 4 millones en acciones tipo *B*. Es importante señalar que el hecho en que la evaluación de la función en el punto arrojado x= (6,4), de un resultado negativo (-), se debe a que MATLAB establece que para maximizar la función *f* se aplica una optimización para minimizar la función *-f*. El punto resultante donde el máximo *f* ocurre también es el punto donde el mínimo de *-f* ocurre.

Referencias Bibliográficas

[1] User`s Guide for Mathworks "*Optimization Toolbox For use with MATLAB*". 1990 – 2006.

[2] MatlabTM R2006a. (Online) disponible en: http://www.mathworks.com.

[3] M.A. Juan Ricardo Salinas Ascencio, Mg. Ovidio Zubieta Bejar, Doc. Juan Morales Romero. "*Problemas resueltos de Programación Lineal*".