**Teoría de dualidad**

**1.** **Descripción matemática**

¿Qué es la teoría de la dualidad?

Todo problema de programación lineal tiene asociado con el otro problema de programación lineal llamado DUAL. El problema inicial es llamado PRIMAL. Los dos conjuntos son llamados problemas duales ya que ambos están formados por el mismo conjunto de datos.

Para obtener el problema dual de un problema primal:

* Se cambia el sentido de optimidad, si en el primal se busca un máximo entonces en el dual se buscará un mínimo y viceversa
* Se transpone el vector de términos independientes del primal y pasa a ser el vector de coeficientes de la función objetivo dual.
* Se transpone el vector de coeficientes del problema primal y pasa a ser el vector de términos independientes de la función objetivo dual.
* Se transpone también la matriz de coeficientes de las restricciones

Ejemplo:

Primal

Max

S.a.:

Dual

Min

S.a.:

En resumen, las relaciones entre problemas duales y primales son las siguientes:

* Si el problema primal es de máximo, el problema dual es de mínimo y viceversa
* Cada restricción del problema primal se relaciona con una variable del problema dual
* Cada variable del problema primal se relaciona con una restricción del problema dual.

Por lo tanto, el problema dual tendrá tantas variables principales como restricciones tenga el primal y tantas restricciones como variables principales tenga el primal.

Para finalizar y obtener la solución se puede utilizar el método simplex.

Vamos a realizar el paso a paso del ejemplo del punto uno para pasar del problema primal al problema dual.

Primal

Max

S.a.:

Comenzamos cambiando el sentido de optimidad, como tenemos un máximo lo pasaremos a minimo

Min G =

Transponemos el vector de términos independientes del primal para formar la función objetivo dual

Como cada variable primal está asociada a una restricción dual, el problema dual estará sujeto a dos restricciones duales.

Transponemos la matriz de coeficientes de las restricciones del primal

La primera columna de la matriz de coeficientes del primal, pasa a ser la primera fila del dual.

La segunda columna de la matriz de coeficientes del primal, pasa a ser la segunda fila del dual.

Ahora los términos independientes del dual van a resultar de transponer los coeficientes de la función objetivo primal, con respecto al signo que tendrán las desigualdades, si la variable tiene un sentido canónico, es decir, es no negativa (en este caso tenemos X>=0) entonces el signo de la desigualdad seguirá en sentido canónico.

Para finalizar, las variables duales, tenemos que cada variable dual se asocia con una restricción primal.

**2. Pseudocódigo**

Trabajaremos con el siguiente ejemplo:

Min

S.a.:

cantI=2

cantE=3

c=[20,30]

A\_ub=[[2,2],[3,1],[1,2]]

b\_ub=[5,3,4]

para x hasta cantE:

b\_ub[x]= b\_ub[x]\*-1

para y hasta cantI

A\_ub[x][y]=A\_ub[x][y]\*-1

res=lingprog(c,A\_ub,b\_ub,bounds=(0,None))

**3. Prueba de escritorio**

Establecemos los valores para cantI (cantidad de términos independientes) y para cantE(cantidad de inecuaciones)

cantI=2

cantE=3

Establecemos los valores de c: función objetivo, A\_ub: matriz de coeficientes, b\_ub: vector solución.

c=[20,30]

A\_ub=[[2,2],[3,1],[1,2]]

b\_ub=[5,3,4]

Como separamos cada parte del ejercicio, no es necesario transponer las matrices, simplemente las filas se trabajarán como columnas y viceversa,

para mantener el sentido canónico del ejercicio multiplicamos a ambos lados de las desigualdad por -1 para cambiar el sentido de >= a <=

Inicio iteraciones

|  |  |
| --- | --- |
| x | 0 |
| b\_ub | [-5,3,4] |
| A\_ub | [[2,2],[3,1],[1,2]] |
| y |  |

|  |  |
| --- | --- |
| x | 0 |
| b\_ub | [-5,3,4] |
| A\_ub | [[-2,2],[3,1],[1,2]] |
| y | 0 |

|  |  |
| --- | --- |
| x | 0 |
| b\_ub | [-5,3,4] |
| A\_ub | [[-2,-2],[3,1],[1,2]] |
| y | 1 |

|  |  |
| --- | --- |
| x | 1 |
| b\_ub | [-5,-3,4] |
| A\_ub | [[-2,-2],[3,1],[1,2]] |
| y |  |

|  |  |
| --- | --- |
| x | 1 |
| b\_ub | [-5,-3,4] |
| A\_ub | [[-2,-2],[-3,1],[1,2]] |
| y | 0 |

|  |  |
| --- | --- |
| x | 1 |
| b\_ub | [-5,-3,4] |
| A\_ub | [[-2,2],[-3,-1],[1,2]] |
| y | 1 |

|  |  |
| --- | --- |
| x | 2 |
| b\_ub | [-5,-3,-4] |
| A\_ub | [[-2,-2],[-3,-1],[1,2]] |
| y |  |

|  |  |
| --- | --- |
| x | 2 |
| b\_ub | [-5,-3,-4] |
| A\_ub | [[-2,2],[-3,-1],[-1,2]] |
| y | 1 |

|  |  |
| --- | --- |
| x | 2 |
| b\_ub | [-5,3,4] |
| A\_ub | [[2,2],[3,1],[-1,-2]] |
| y | 2 |

Se pasan los valores nuevos de b\_ub y A\_ub a la función linprog para que optimice el problema

Nos arroja los valores para las x

x: array([1 , 1.5])

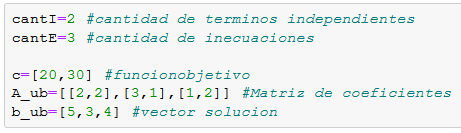
Valor optimo: 64.999

**4. Programación**

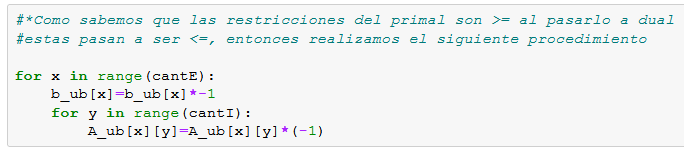
Importación de librerías necesarias

****

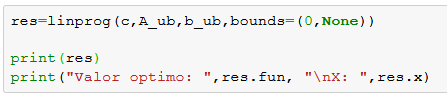
Definición de variables



Cambio de la desigualdad



Solución del problema dual



**Relaciones primal-dual**

Asociado a cada problema lineal existe otro problema de programación lineal denominado problema dual (PD) , que posee importantes propiedades y relaciones notables con respecto al problema lineal original, problema que para diferencia del dual se denomina entonces como problema primal (PP).

a). El problema dual tiene tantas restricciones como variables tiene el programa primal.

b). El problema dual tiene tantas variables como restricciones tiene el programa primal.

c). La matriz de coeficientes técnicos del problema dual es la traspuesta de la matriz técnica del problema primal.



Ejemplo:

Maximizar:

Primero se identifica lo el problema a desarrollar en este caso un PRIMAL, como en este se quiere encontrar un Máximo, en el problema a resolver será un DUAL el cual es un Mínimo.

Luego se toman los valores independiente los cuales acompañan los valores de las restricciones, ya una vez ubicado los valores independientes estos se llevaran a la nueva función a minimizar

Minimizar:

Para el problema dual se hará la transpuesta de los valores de las restricciones y quedara de la siguiente forma

Y ya para hacer las condiciones de las variables se tomaran como lo muestra en la tabla

**Pseudocodigo**

**metodo newZ(Restriction):**

W=[]

para i in range(0,(Restriction).longitud):

W.adicionar(Restriction[i][2])

string="W = "

c=1

paraCada i in W:

string +=str(i)

string += "y"

string += str(c)

string += " "

c=c+1

imprimir(string)

return W

**metodo transform(Z, Restriction):**

opt=introducir("Que desea hacer maximizar(max) o minimizar(min) : \n")

while(opt.lower() != "max" and opt.lower() != "min"):

opt=introducir("Que es un problema de maximizar(max) o minimizar(min) : \n")

sign = "<="

si (opt == “max”) entonces

sign = ">="

F = []

paraCada i en restriccion

F.adiccionar([len(i)-1])

Result = []

para i hasta range(0, Z.longitud)

a = []

para j hasta range(0, Restriction.longitud)

a.adiccionar(restriccion[j][i])

a.adiccionar (sign)

a.adiccionar(Z[i])

Result.adiccionar(aux)

return Result

**metodo show(Result):**

newZ(Restriction)

s=""

para i hasta result.longitud

string = ""

para j hasta Result[i].longitud

string += Result[i][j].ToString

if(j< (result[0]).longitud-2):

string += "y"

string += (j+1).ToString

string += " "

imprimir("Result equation " + i+1 + " is: " + line)

para i = 0 hasta result[0].longitud-2

s+= ("y").ToString

s+=(i+1). ToString

if(i<result[0].longitud-3):

s+= (", ") .ToString

imprimir (s, ">= 0")

Z = [2, 12]

Restriction= [ [1, 2, 10],

[5, 6, 20],

[2, 1, 12] ]

result = solution(Z, Restriction)

show(result)

**Prueba de escritorio**

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [] |
| a | [1] |
| sign | >= |
| opt | max |
| i | 0 |
| j | 0 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [] |
| a | [1,5] |
| sign | >= |
| opt | max |
| i | 0 |
| j | 1 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [] |
| a | [1,5,2, “>=”, 2] |
| sign | >= |
| opt | max |
| i | 0 |
| j | 2 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [[1,5,2, “>=”, 2],] |
| a | [1,5,2, “>=”, 2] |
| sign | >= |
| opt | max |
| i | 0 |
| j | 2 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [[1,5,2, “>=”, 2],] |
| a | [2] |
| sign | >= |
| opt | max |
| i | 1 |
| j | 0 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [[1,5,2, “>=”, 2],] |
| a | [2,6] |
| sign | >= |
| opt | max |
| i | 1 |
| j | 1 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| Result | [ [1,5,2, “>=”, 2],[ 2,6,1, “>=”, 12] ] |
| a | [2,6,1, “>=”, 12] |
| sign | >= |
| opt | max |
| i | 1 |
| j | 2 |

**newZ**

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10] |
| i | 0 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20] |
| i | 1 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “W = ” |
| c | 1 |
| i | 2 |

|  |  |
| --- | --- |
| Restriction | [[1,5,2, “>=”, 2],[ 2,6,1, “>=”, 12]] |
| W | [10,20,12] |
| string | “W = 10y1 ” |
| c | 1 |
| i | 10 |

|  |  |
| --- | --- |
| Result | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “W = 10y1 20y2” |
| c | 2 |
| i | 20 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “W = 10y1 20y2 12y3” |
| c | 3 |
| i | 12 |

**Show**

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “1y1” |
| s | “” |
| i | 0 |
| j | 0 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “1y1 5y2” |
| s | “” |
| i | 0 |
| j | 1 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “1y1 5y2 2y3” |
| s | “” |
| i | 0 |
| j | 2 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “1y1 5y2 2y3 >=” |
| s | “” |
| i | 0 |
| j | 3 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “1y1 5y2 2y3 >= 2” |
| s | “” |
| i | 0 |
| j | 4 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 ” |
| s | “” |
| i | 1 |
| j | 0 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 ” |
| s | “” |
| i | 1 |
| j | 1 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3” |
| s | “” |
| i | 1 |
| j | 2 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3 >=” |
| s | “” |
| i | 1 |
| j | 3 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3 >= 12” |
| s | “” |
| i | 1 |
| j | 4 |

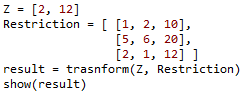
|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3 >= 12” |
| s | “y1, ” |
| i | 0 |
| j | 4 |

|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3 >= 12” |
| s | “y1, y2, ” |
| i | 1 |
| j | 4 |

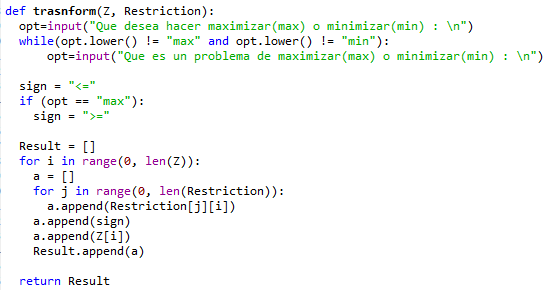
|  |  |
| --- | --- |
| Restriction | [ [1, 2, 10], [5, 6, 20], [2, 1, 12] ] |
| W | [10,20,12] |
| string | “2y1 6y2 1y3 >= 12” |
| s | “y1, y2, y3” |
| i | 2 |
| j | 4 |

**Programación**

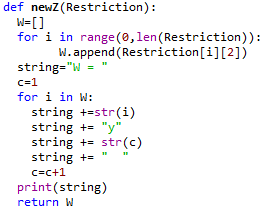
Creación de variables y llamado a los métodos



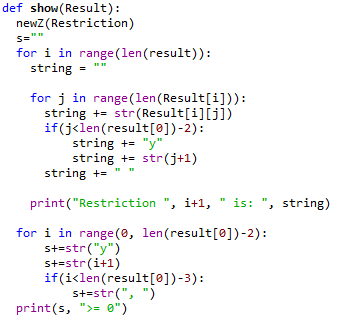
Método Para transformación de la matriz



Método para la nueva “Z”



Método para mostrar los resultados en pantalla



**Adaptación a otras formas del primal**

Es la conversión de programación lineal a la forma estándar. Construcción del dual de un problema dual. Método CER. Formas primal-dual correspondientes.

Estas conversiones se resumen en la siguiente tabla. Con esto, siempre se tiene la opción de convertir cualquier modelo a nuestra forma estándar y después construir su problema dual en la forma usual. Como ejemplo, se hace esto para el problema dual estándar (que también debe tener un dual). Nótese que el resultado de todo esto es justo el problema primal estándar Como cualquier par de problemas primal y dual se pueden cambiar a estas formas, este hecho demuestra una propiedad importante de las relaciones primal-dual:

**Propiedades de simetría**

Para cualquier problema primal y su problema dual, todas las relaciones entre ellos deben ser simétricas ya que el dual de este problema dual es este problema primal.

|  |  |
| --- | --- |
| Problema primal | Problema dual |
| Superóptima | Subóptima |

Conversiones de los modelos de programación lineal a la forma estándar

|  |  |
| --- | --- |
| Forma no estándar | Forma estándar equivalente |
| Minimizar Z | Maximizar (-Z)  y    (), |

Como resultado, todas las afirmaciones que se hicieron antes sobre las relaciones del problema dual con el primal también se cumplen en el sentido inverso.

Otra consecuencia de la propiedad de simetría es que no tiene ninguna trascendencia a cuál de los problemas se le dé el nombre de primal y a cuál el de dual. En la práctica, se puede encontrar un problema de programación lineal que se ajuste a nuestra forma estándar y al que se le dé de problema dual. La convención es que el modelo formulado para representar el problema real recibe el nombre de problema primal sin importar qué forma tiene.

**Construcción del dual de un problema dual**

|  |
| --- |
| **Problema dual** |
| Minimizar  Sujeta a  Y |

|  |
| --- |
| **Convertido en la forma estándar** |
| Maximizar (  Sujeta a  Y |

|  |
| --- |
| **Su problema dual** |
| Minimizar (  Sujeta a  Y |

|  |
| --- |
| **Convertido en la forma estándar** |
| Maximizar (  Sujeta a  Y |

**Solución del ejemplo de terapia de radiación**

El ejemplo está casi listo para que se le aplique el método simplex. Si se usa la forma de maximización que se acaba de obtener, el sistema de ecuaciones completo es

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **-Z** | **+** | **+** |  | **+ M** |  | **+ M** | = 0 |
| **1** |  | **+** | **+** | **+** |  |  |  | **=** 2.7 |
| **2** |  | **+** | **+** |  | **+** |  |  | **=** 6 |
| **3** |  | **+** | **+** |  |  | -x5 | **+** | **=** 6 |

**Algoritmo**

1. Formule el problema primal de cualquier forma, maximización o minimización, y el problema dual automáticamente quedará en la forma contraria.

|  |  |  |  |
| --- | --- | --- | --- |
| **-Z** | **+** | **+** | = 0 |
|  | **+** | **+** | **=** 2.7 |
|  | **+** | **+** | **=** 6 |
|  | **+** | **+** | **=** 6 |

1. Determine si cada forma de las restricciones funcionales y de las restricciones sobre las variables es común. La condición de las restricciones funcionales depende de que el problema sea un problema de maximización o un problema de minimización.

**Maximizar -Z = - -**

|  |  |  |  |
| --- | --- | --- | --- |
| **C** | **+** | **+** | 2.7 |
| **E** | **+** | **+** | **=** 6 |
| **R** | **+** | **+** | 6 |

1. Por cada restricción sobre una variable individual en el problema dual, use la forma que tiene la misma condición que la restricción funcional en el problema primal que corresponde a esta variable dual.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C** | **+** | **+** | 2.7 |  |
| **E** | **+** | **+** | **=** 6 |  |
| **R** | **+** | **+** | 6 |  |

1. Por cada restricción funcional en el problema dual, use la forma que tiene la misma condición que la restricción sobre la variable individual correspondiente del problema primal.

|  |  |
| --- | --- |
|  |  |
|  |  |

**Programación**

**#formular el problema primal**

**Z=[0.4,0.5,0]**

**C=[0.3,0.1,2.7]**

**E=[0.5,0.5,6]**

**R=[0.6,0.4,6]**

**newC = C[0] + C[1] =< C[2]**

**newE = E[0])+E[1]=E[2]**

**newR = R[0]+ R[1]) =>R[2]**

**cantI = 2**

**for x in range(cantI):**

**Z[x] = Z[x]\*-1**

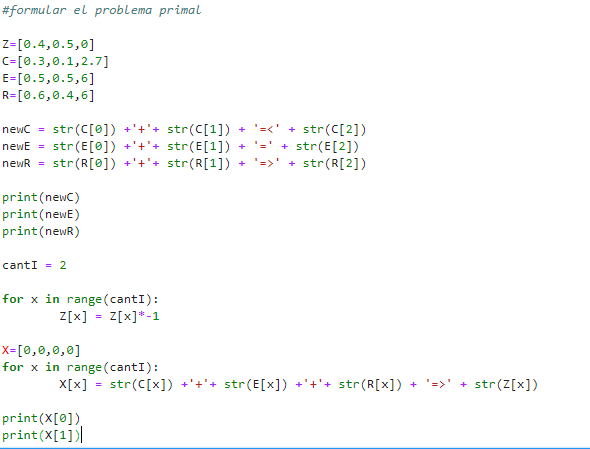
**X=[0,0,0,0]**

**for x in range(cantI):**

**X[x] = str(C[x]) +'+'+ str(E[x]) +'+'+ str(R[x]) + '=>' + str(Z[x])**

**print(X[0])**

**print(X[1])**



**Repositorio GitHub**

<https://github.com/OscarDogar/Tarea2>

**Bibliografía**

 *CHAPRA, Steven C y Raymond P. Canale. Métodos Numéricos para Ingenieros. México:*

 *McGraw-Hill. Séptima edición. 2015 BRONSON, RICHARD. Investigación de operaciones. Primera edición. México:McGraw-Hill. 1996.*

 *WINSTON, Wayne. Investigacion de operaciones, aplicaciones y algoritmos. Cuarta edición. Cengage Learning. 2008.*