

# Chapitre 1

## Utiliser `optimtool` pour résoudre les problèmes PL

**D**ans ce chapitre, nous allons montrer à travers des illustrations guidées comment utiliser cette boîte à outils pour résoudre les problèmes de la programmation linéaire.

## 1.1. Introduction

Nous montrons avec des illustrations comment utiliser cette outil pour résoudre quelques types de problèmes de la programmation linéaire.

## 1.2. Illustration de la programmation linéaire

Dans cette section, nous présentons quelques illustrations de programmes linéaires avec égalités et inégalités.

Le problème est :

$$\min_x f^T x,$$

Sachant que :

$$\begin{cases} A \cdot x \leq b \\ Aeq \cdot x = 0 \\ x \geq 0 \end{cases}$$

Comme une première approche, vous pouvez utiliser directement l'interface graphique de l'outil d'optimisation.

- Le choix du solveur linprog.
- Le choix de l'algorithme Simplex.
- Faire entrer la fonction objectif et les contraintes (A, Aeq, b, beq, f).
- Régler les options ...
- Exécuter le solveur et visualiser les résultats.

### Illustration 1.1

Cette illustration montre comment utiliser l'outil d'optimisation avec le solveur linprog pour minimiser une fonction linéaire soumise à des contraintes et des limites linéaires.

Considérez le problème d'optimisation suivant :

$$\min Z(x_1, x_2): 2x_1 + 6x_2$$

S.C :

$$x_1 + x_2 \leq 40$$

$$x_1 - x_2 = 30$$

$$-x_1 + 4x_2 \geq -160$$

$$x_1 \text{ illimitée}; x_2 \geq 8$$

Étape 1: Écrivez le problème sous la forme matricielle.

- Vous remarquez que le problème est un problème de minimisation, donc on saisit les coefficients de la fonction objectif tels qu'ils sont. On a deux inégalités, une égalité et les contraintes de borne.
- Ramenez les inégalités sous la forme  $A \cdot x \leq b$ . Donc, on doit multiplier la deuxième inégalité par (-1).
- Écrivez le problème sous la forme matricielle, et extraire les vecteurs et les matrices des coefficients.

Étape 2: configurez et exécutez le problème à l'aide de l'outil d'optimisation.

- Saisir `optimtool` dans « **Command Window** » (la fenêtre de commande) pour ouvrir l'outil d'optimisation (**Optimization Tool**).
- Sélectionnez « **linprog** » dans la liste des solveurs et changez le champ « **Algorithm** » en Medium scale - simplex.

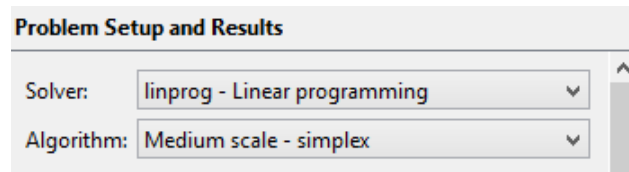


Figure 4.1. Choix du solveur et d'algorithme.

Définissez les contraintes.

- Pour créer des variables pour la fonction objectif, faire entrer les coefficients [2 6] dans le champ f.
- Pour créer des variables pour les contraintes d'inégalités, faire entrer [1 1;1 -4] dans le champ A et entrez [40;160] dans le champ b.
- Pour créer des variables pour les contraintes d'égalité, faire entrer [1 -1] dans le champ Aeq et entrez 30 dans le champ beq.
- Définissez les limites des variables, faire entrer [-inf 8] dans le champ Lower pour les limites inférieures et laissez les bornes supérieures non définies.

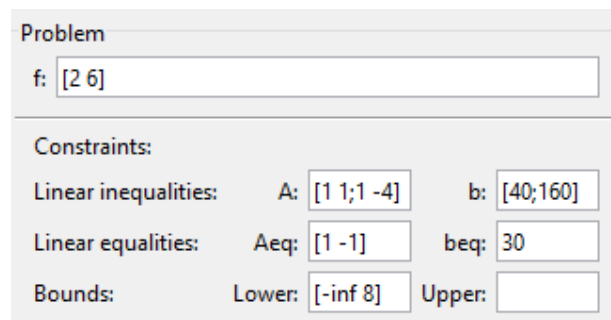


Figure 4.2. Saisie des entrées.

- Pour un premier lancement, l'algorithme choisit le point de départ (par défaut [0;0]). Après la première exécution du solveur, vous pouvez spécifier un point de départ quelconque dans le champ « **Start point** » (Point de départ).

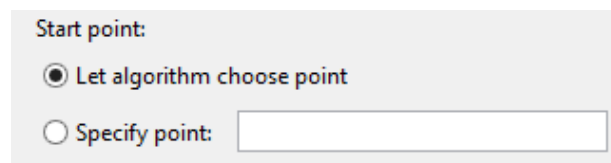


Figure 4.3. Choix de la solution initiale.

- Dans le volet Options, développez l'option « **Display to command window** » (Afficher dans la fenêtre de commande) si nécessaire, et sélectionnez « **Iterative** » (itératif) pour afficher les informations d'algorithme à « **Command Window** » (la Fenêtre de commande) pour chaque itération.
- De plus, pour afficher un rapport de diagnostic détaillé dans la fenêtre de commande, cochez « **Show diagnostics** ».

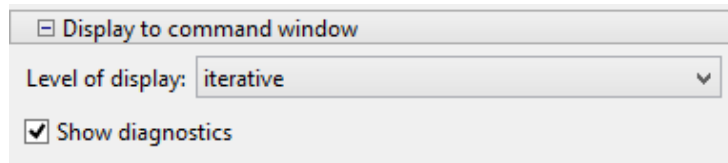


Figure 4.4. Affichage dans la fenêtre de commande.

- Vous pouvez aussi spécifier des conditions d'arrêts dans l'option « **Stopping criteria** ». Comme vous pouvez les laissez par défaut.

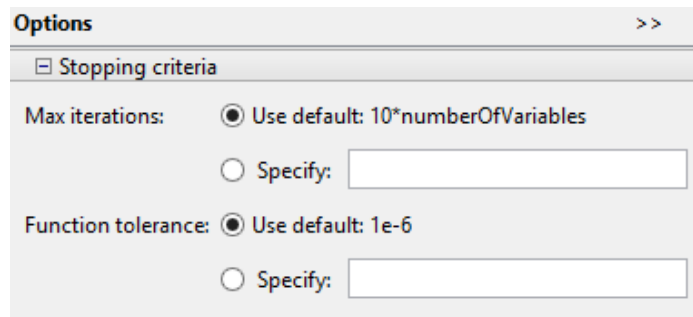


Figure 4.5. Critères d'arrêts.

- Cliquez sur le bouton « **Start** » (Démarrer), comme illustré dans la figure suivante.

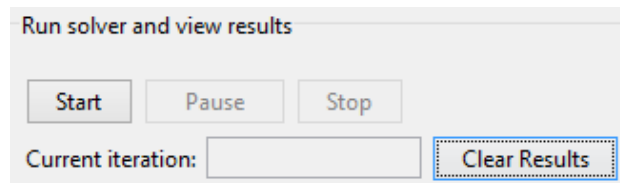


Figure 4.6. Lancement du solveur.

- Lorsque l'algorithme se termine, sous « **Run solver and view results** », les informations suivantes s'affichent:

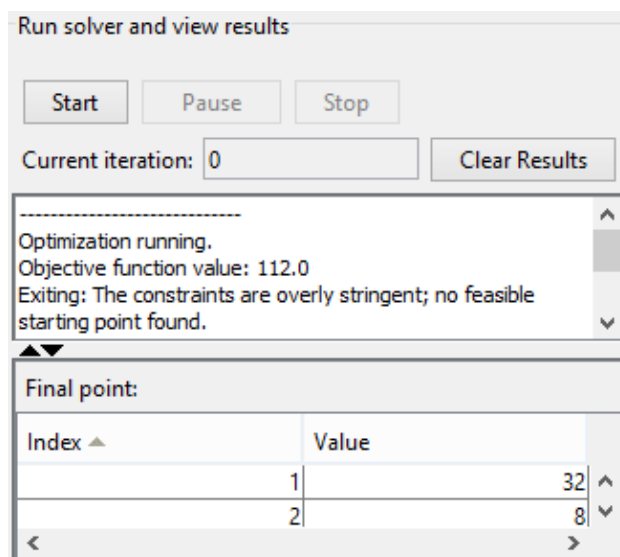


Figure 4.7. Visualisation des résultats.

- o La valeur d'itération actuelle (**Current iteration**) à la fin de l'algorithme, qui est 0 pour cette illustration.
- o La valeur finale de la fonction objectif à la fin de l'algorithme est :  
Valeur de la fonction objectif: 112.0
- o Le message de fin d'algorithme est :

```
Optimization running.
Objective function value: 112.0
Exiting: The constraints are overly stringent; no feasible
starting point found.
```

- o Le point final, qui est pour cette illustration :

Final point:	
Index ▲	Value
1	32
2	8

Figure 4.8. Point final.

- o Dans **Command Window** (la fenêtre de commande), les informations de l'algorithme sont affichées pour chaque itération:

```
-----
Diagnostic Information

Number of variables: 2

Number of linear inequality constraints:    2
Number of linear equality constraints:     1
Number of lower bound constraints:        1
Number of upper bound constraints:        0

Algorithm selected
medium-scale: simplex

-----
End diagnostic information

Phase 1: Compute initial basic feasible point.
      Iter      Infeasibility
      0          38
      1           6
Exiting: The constraints are overly stringent; no feasible starting point found.
>> |
```

Si vous travaillez sur la fenêtre de commande :

Vous pouvez charger les matrices et les vecteurs A, Aeq, b, beq, f et les limites inférieures lb dans MATLAB workspace (l'espace de travail MATLAB) s'ils sont déjà créés (file.mat) avec la commande :

```
>> load nomfichier
```

Ou le créé directement dans l'espace de travail MATLAB.

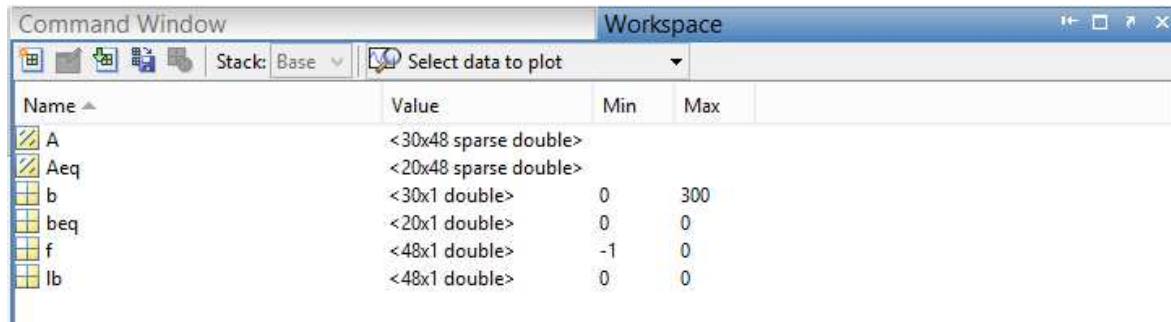
## Illustration 1.2

Prenez le problème qui existe déjà dans Matlab sous le nom `sc50b.mat` a 48 variables, 30 inégalités et 20 égalités.

Vous chargez les matrices et les vecteurs `A`, `Aeq`, `b`, `beq`, `f` et les limites inférieures `lb` dans l'espace de travail MATLAB avec la commande :

```
>> load sc50b
>>
```

La sous fenêtre de l'espace de travail MATLAB s'affiche comme suit :



Name	Value	Min	Max
A	<30x48 sparse double>		
Aeq	<20x48 sparse double>		
b	<30x1 double>	0	300
beq	<20x1 double>	0	0
f	<48x1 double>	-1	0
lb	<48x1 double>	0	0

Figure 4.10. La sous fenêtre de l'espace de travail MATLAB.

Puis, vous pouvez utiliser `linprog` dans la fenêtre de commande pour résoudre le problème:

```
[x,fval,exitflag,output]=linprog(f,A,b,Aeq,beq,lb,[],[],optimset('Display',  
'iter')) ;
```

Sachant que la syntaxe générale est la suivante :

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

### N.B :

- Un argument = [] s'il n'existe pas.
- Pour le point de départ `x0`. Cette option n'est disponible qu'avec l'algorithme de moyenne échelle. L'option `LargeScale` est désactivée avec `optimset`. Par défaut l'algorithme à grande échelle et l'algorithme simplexe ignorent tout point de départ.

Étant donné que l'affichage itératif a été défini à l'aide de `optimset`, les résultats affichés sont :

```
>> load sc50b
>> [x,fval,exitflag,output]=linprog(f,A,b,Aeq,beq,lb,[],[],optimset('Display','iter'));
```

Residuals:	Primal	Dual	Duality	Total
	Infeas	Infeas	Gap	Rel
	A*x-b	A'*y+z-f	x'*z	Error
Iter 0:	1.50e+03	2.19e+01	1.91e+04	1.00e+02
Iter 1:	1.15e+02	2.37e-15	3.62e+03	9.90e-01
Iter 2:	1.61e-12	2.62e-15	4.32e+02	9.48e-01
Iter 3:	3.11e-12	4.26e-15	7.78e+01	6.88e-01
Iter 4:	6.00e-11	6.43e-16	2.38e+01	2.69e-01
Iter 5:	3.56e-11	9.43e-16	5.05e+00	6.89e-02
Iter 6:	7.35e-11	1.21e-16	1.64e-01	2.34e-03
Iter 7:	3.35e-12	1.20e-16	1.09e-05	1.55e-07
Iter 8:	1.73e-12	1.38e-16	1.09e-11	1.57e-13

Optimization terminated.

```
>>
```

### Illustration 1.3

Une troisième alternative est d'écrire un M-file (programmation en Matlab) monfichier.m (Resolution\_sc50b.m) et vous déclarer les matrices et les vecteurs A, Aeq, b, beq, f et les limites inférieures lb.

Dans le même fichier vous écrivez

```
% Problème existe déjà dans Matlab sous le nom sc50b.mat
% a 48 variables, 30 inégalités et 20 égalités.
% Chargement des f,A,b,Aeq,beq,lb

load sc50b
options =optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output]=...
linprog(f,A,b,Aeq,beq,lb,[],[],optimset('Display','iter'))
```

Puis vous exécuter.

Après l'exécution du fichier.mat (Resolution\_sc50b.m) des paramètres et des variables sont créé automatiquement.

```
>> Resolution_sc50b
```

Residuals:	Primal	Dual	Duality	Total
	Infeas	Infeas	Gap	Rel
	A*x-b	A'*y+z-f	x'*z	Error
Iter 0:	1.50e+03	2.19e+01	1.91e+04	1.00e+02
Iter 1:	1.15e+02	2.37e-15	3.62e+03	9.90e-01
Iter 2:	1.61e-12	2.62e-15	4.32e+02	9.48e-01
Iter 3:	3.11e-12	4.26e-15	7.78e+01	6.88e-01
Iter 4:	6.00e-11	6.43e-16	2.38e+01	2.69e-01
Iter 5:	3.56e-11	9.43e-16	5.05e+00	6.89e-02
Iter 6:	7.35e-11	1.21e-16	1.64e-01	2.34e-03
Iter 7:	3.35e-12	1.20e-16	1.09e-05	1.55e-07
Iter 8:	1.73e-12	1.38e-16	1.09e-11	1.57e-13

```
Optimization terminated.
```

```
>> |
```

Pour les problèmes à grandes échelles, l'algorithme de programmation linéaire (large-scale linear programming) rapidement réduit les résidus échelonnés en dessous de la tolérance par défaut de  $1e-08$ .

La valeur `exitflag` est positive, vous indiquant que `linprog` a convergé. Vous pouvez obtenir également la valeur de fonction finale en `fval` et le nombre d'itérations dans (`output.iterations`):

```
fval =
```

```
-70.0000
```

```
exitflag =
```

```
1
```

```
output =
```

```
iterations: 8
algorithm: 'large-scale: interior point'
cgiterations: 0
message: 'Optimization terminated.'
constrviolation: 1.2790e-12
firstorderopt: 2.7698e-13
```

```
>> |
```



Les tableaux suivants fournissent les explications de chaque argument de la fonction linprog :

Arguments d'entrée	
f	Vecteur des coefficients de la fonction objectif linéaire.
Aineq	Matrice des coefficients pour les contraintes d'inégalité linéaires.
bineq	Vecteur du second membre pour les contraintes d'inégalité linéaires.
Aeq	Matrice des coefficients pour les contraintes d'égalité linéaires.
beq	Vecteur du second membre pour les contraintes d'égalité linéaires.
lb	Vecteur des bornes inférieures.
ub	Vecteur des bornes supérieures.
x0	Point initial pour x.

Tableau 1.1. Arguments d'entrée –linprog-

Arguments de sortie		
exitflag	Entier identifiant la raison pour laquelle l'algorithme à terminé.	
	1	Fonction convergée vers une solution x.
	0	Nombre d'itérations dépassé options.MaxIter.
	-2	Aucune solution réalisable n'a été trouvée.
	-3	Le problème est non borné.
	-4	Une valeur NaN a été rencontrée pendant exécution de l'algorithme.
	-5	Problèmes primal et dual sont irréalisables.
-7	La direction de recherche est devenue trop petite. Aucun autre progrès n'a pu être réalisé.	
lambda	Structure contenant les multiplicateurs de Lagrange à la solution x (séparée par type de contrainte). Les domaines de la structure sont:	
	lower	Bornes inférieures lb.
	upper	Bornes supérieures ub.
	ineqlin	Inégalités linéaires.
eqlin	Egalités linéaires.	
output	Structure contenant des informations sur l'optimisation. Les domaines sont:	
	iterations	Nombre d'itérations.
	algorithm	Algorithme d'optimisation utilisé.
	cgiterations	0 (algorithme à grande échelle seulement, inclus pour backward compatibility)
	message	Quitter le message.

Tableau 1.2. Arguments de sortie –linprog-

## Exercice A domicile

Reprenez les mêmes étapes de l'illustration précédente pour le programme linéaire avec des colonnes denses dans les égalités qui est sous le nom denscolumns.mat.