Cours 4. Tutoriel AMPL

A. Désilles

12 octobre 2012

Résumé

Installation

Syntaxe de base

Quelques détails de plus

Résumé

Installation

Syntaxe de base

Quelques détails de plus

Résumé

Installation

Syntaxe de base

Quelques détails de plus

Résumé

AMPL

- AMPL est une language qui permet de décrire un problème d'optimisation de façon normalisée pour permettre de le résoudre par un logiciel spécialisé
- AMPL peut être utilisé ave un grand nombre de solveurs pour la programmation linéaire et non linéaire (MINOS, CPLEX, NPOPT, IPOPT, etc)
- La console AMPL permet de :
  - décrire un problème ou de charger un problème depuis une description enregistrée dans un (ou plusieurs) fichier(s) ;
  - choisir un solveur et l'exécuter pour résoudre le problème
  - afficher les résultats sur écran
  - enregistrer les résultats dans des fichiers

Installer

- Pour utiliser AMPL on doit disposer de la console AMPL (exécutable ampl) et d'un ou plusieurs solveurs, placés dans le même répertoire
- Téléchargez le fichier amplKit.zip et décompressez son contenu dans un seul répertoire.
- Ouvrez un terminal dans le répertoire où sont installés tous les programmes.
- En ligne tapez ./ampl . la console AMPl est prête : ampl :

Structure générale

Pour décrire un problème de minimisation de type $\min_{x \in K} J(x)$ on doit :

- définir les variables de décision : mot-clé `var`

```
var nom1;
var nom2;
...
```

- définir la fonction objectif et le type d'optimisation :

```
minimize NomFonction: expression;
```

- décrire l'ensemble decontraintes K : mot-clé `s.t.`

```
s.t. nom1: expression1;
s.t. nom2: expression2;
...
```

- toutes les commandes doivent se terminer par " ;"

"Hello Word"

Considérons le problème de minimisation de la fonction de Rosenbrock

$$\min_{x,y \in \mathbb{R}^2} (1-x)^2 + 100(y-x^2)^2$$

Son minimum global est $x_0 = (1,1)$.
Ce problème de minimisation se définit avec AMPL comme suit :

```
 #   variables de décision
var x;
var y;

# Définition de la fonction objectif
minimize J_Rosen: 100*(y - x^2)^2 + (1-x)^2;

# Donner une valeur initiale pour l'algorithme (optionnel)
let x:=-1.;
let y:=1.1;
```

"Hello Word"

Enregistrez ce code dans une fichier texte `modeleR.mod` :

```
 #   variables de décision
var x;
var y;

# Définition de la fonction objectif
minimize J_Rosen: 100*(y - x^2)^2 + (1-x)^2;

# Donner une valeur initiale pour l'algorithme (optionnel)
let x:=-1.;
let y:=1.1;
```

Pour charger et résoudre un problème on utilise les commandes de console suivantes :

- `model` : permet de charger un problème depuis un fichier. Tapez :

  ```
  model modeleR.mod;
  ```

- `solve` : permet de résoudre le problème chargé. Tapez :

  ```
  solve;
  ```

- `display` : permet d'afficher les valeurs des variables ou paramètres. Tapez :

  ```
  display x,y;
  ```

  puis

  ```
  display J_Rosen;
  ```

Exercice

Ecrivez un modèle pour le problème suivant

$$\min_{x,y \in \mathbb{R}^2} \frac{x^3}{3} + \frac{x^2}{2} + 2xy + \frac{y^2}{2} - y + 9$$

Pour annuler le modèle précédent utilisez la commande `reset;` .
Calculez la solution et affichez la sur la console.

Un problème avec contraintes

Considérons le problème de minimisation suivant

$$\min_{x \in \mathbb{R}^4} 1 + x_1 + x_2 + x_3 + x_4$$

sous contraintes :

$$Ax \leq b$$

où

$$A = \begin{pmatrix} 4 & 2.25 & 1 & 0.25 \\ 0.16 & 0.36 & 0.64 & 0.64 \end{pmatrix}, \quad b = \begin{pmatrix} 0.0401 \\ 0.010085 \end{pmatrix}$$

et

$$\begin{pmatrix} 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \end{pmatrix} \leq x \leq \begin{pmatrix} 400000 \\ 300000 \\ 200000 \\ 100000 \end{pmatrix}$$

Déclaration de variables vectorielles

```
var x{1..4};
```

Déclare un vecteur x de $\mathbb{R}^4$. On peut accéder aux composantes de x avec

```
x[1], x[2], x[3], x[4]
```

On peut de même définir une matrice

```
var matrice{0..3, -3..3}
```

et accéder à ses éléments `matrice[0,-2]`.
Certaines opérations sur les vecteurs ont une syntaxe simplifiée :

```
sum {j in 1..4} x[j]
```

pour une somme et `prod {j in 1..4} x[j]` pour un produit. Pour répéter une expression dépendant d'indice on insère devant `{indice in début..fin}` ;

Déclaration de paramètres

Le mot clé `param` permet de définir des paramètres du problème, des coefficients constants par exemple. On peut leur attribuer une valeur à la déclaration

```
param n:=4;
```

et utiliser leur nom plus loin :

```
var x{1..n}>=0
```

On peut définir une matrice paramètre

```
param a: 1 2 3 4 :=
1 4 2.25 1 0.25
2 0.16 0.36 0.64 0.64
;
```

ou de manière équivalente

```
param a:=
1 1 4
1 2 2.25
1 3 1
1 4 0.25
2 1 0.16
2 2 0.36
2 3 0.64
2 4 0.64
;
```

Début du modèle EX.mod

```
#Variables avec une contrainte de borne
var x {1..4} >= 0.001;

#paramètres
param a {1..2, 1..4};
param b {1..2};

# Fonction objectif
minimize obj: 1 + sum {j in 1..4} x[j];
```

Déclaration des contraintes dans EX.mod

```
# Contraintes d'inégalité

subject to contrInegal {i in 1..2}:
 sum {j in 1..4} a[i,j]/x[j]<=b[i];

# Contraintes de borne

subject to BornSup {j in 1..4}: x[j] <= (5-j)*1e5;
```

Bloc de données EX.mod

```
data;
param a: 1 2 3 4 :=
1 4 2.25 1 0.25
2 0.16 0.36 0.64 0.64 ;


param b :=
1 0.0401
2 0.010085 ;

#condition initiale

let x[1] := 1;
let x[2] := 1;
let x[3] := 1;
let x[4] := 1;
```

Exercices

- Ecrivez un modèle pour le problème suivant

$$\min_{x,y \in \mathbb{R}^2} sin(x+y) + (x-y)^2 - 1.5 * x + 2.5 * y + 1$$

sous les contraintes :

$$-1.5 \leq x \leq 4.0, \quad -3 \leq y \leq 3$$

Calculez la solution et affichez la sur la console.

- Ecrivez un modèle pour le problème suivant

$$\min_{x,y \in \mathbb{R}^2} x_1^2 + 0.5x_2^2 + x_3^2 + 0.5x_4^4 - x_1x_2 + x_3x^4 - x_1 - 3x_2 + x_3 - x_4$$

sous les contraintes :

$$\begin{array}{l} 5 - x_1 - 2x_2 - x_3 - x_4 \geq 0 \\ 5 - 3x_1 - x_2 - 2x_3 + x_4 \geq 0 \\ x_2 + 4x_3 - 1.5 \geq 0 \\ x_i \geq 0, \quad i = 1, \ldots, 4 \end{array}$$

Calculez la solution et affichez la sur la console.

Fonctions mathématiques

```
 abs(x)      sin(x)    cos(x)          exp(x)
log(x)      log10(x)  min(x,y,...)    max(x,y,...)
round(x,n)  round(x)  precision(x,n)  trunc(x)
trunc(x,n)  sqrt(x)   floor(x)
```

et quelques fonctions spéciales

```
Beta(a,b)   Cauchy()   Exponential()   Gamma(a)
Irand224()  Normal()   Poisson()       Uniform(m,n)
```

Le nombre π

```
param pi := 4*atan(1);
```

Autres commandes de console

- Définition d'option

```
option optionName  value ;
```

Exemple : indiquer le choix de solveur :

```
option solver  snopt ;
```

- Ecrire les résultats dans un fichier :

```
printf [index:] "format",listeVariables
> nomFichier;
```

Exemple :

```
ampl: printf {i in 0..N-1}: "%10f %10f \n",i*h, v[i]
ampl? > vitesse.dat;
```