ENT 305A — AMPL Tutorial

Laurent Pfeiffer (Inria and CentraleSupélec, University Paris-Saclay)

1. GENERAL STRUCTURE OF THE CODE

The code is organized in three files, saved in the same folder:

- the **model** file, where the optimization variables and the parameters are declared, the cost function and the constraints are programmed, and the optimization variables are initialized;
- the **parameter** file, where the parameters are instantiated;
- the **instruction** file.

Consider the following optimization problem:

$$\inf_{(x,y)} \; x^2 + a(x+y) + 2y^2$$
$$\text{s.t.:} \begin{cases} x + y = b \\ x \geq 0, \end{cases}$$

where $a = -4$ and $b = 2$. A model file could be as follows.

```
# Declaration of optimization variables
var x;
var y;
# Declaration of parameters
param a;
param b;
# Cost function
minimize f: x^2 + a*(x+y) + 2*y^2;
# Constraints
s.t. g: x+y = b;
s.t. h: x >= 0;
# Initialization of optimization variables
let x:= 1;
let y:= 2.5;
```

The corresponding parameter file can be written in this way.

```
# Instantiation of parameters variables
param a:= -4;
param b:= 2;
```

Save the model file and the parameter file in two files called `model1.txt` and `param1.txt`.

The instruction file (call it `script1.txt`) can be as follows:

```
reset;
model model1.txt;
data param1.txt;
solve;
display x,y;
display f;
display g.dual;
display h.dual;
```

Finally, to solve to the problem, write `include script1.txt` in the terminal.

Remarks.

- There is a semicolon `;` at the end of each instruction.
- One has to give a name to the cost function (here, `f`) and a name to each constraint (here, `g` and `h`).
- It is not mandatory to initialize the optimization variables, by default, the initial value is 0.
- Real-valued parameters can be declared and instantiated directly in the model file, for example with `param a:=-4;` in place of `param a;`.
- No algebraic operation can be done in the parameter file, for example, `y:= 1/3;` is not understood in the parameter file.
- Standard operations can be used with AMPL, for example: `exp(x)`, `log(x)`, `min(x,y)`, `sqrt(x)`,...
- For the multiplication, the sign `*` cannot be omitted.

2. VECTORS AND MATRICES

The command `var x;` declares a real-valued optimization variable called `x`. Write `var x{1..10};` to declare an optimization variable in $\mathbb{R}^{10}$, with indices 1, 2,...,10. The i-th coordinate of `x` can be called with `x[i]`.

Consider for example the problem

$$\inf_{(x_2,x_3,x_4)\in\mathbb{R}^3} \; x_2^4 + 2x_2 + x_3^2 + 2x_3x_4 + x_4^2.$$

The model file is as follows.

```
var x{2..4};
minimize f: x[2]^4 + 2*x[2] + x[3]^2
        + 2*x[3]*x[4] + x[4]^2;
```

Similarly, matrix-valued optimization variables can be used. For example, the command `var x{1..3,0..5};` is used to declare a optimization variable `x` with two indices running from 1 to 3 and from 0 to 5. The coordinate corresponding to the indices 2 and 4 can be called with `x[2,4]`.

Vector- and matrix-valued parameters are declared in the same fashion, using the keyword `param` instead of `var`, in the model file. Consider for example a parameter y defined by:

$$y = (y(4), y(5), y(6)); \quad y(4) = 0,\ y(5) = 1,\ y(6) = 3.$$

It is declared in the model file with

```
param y{4..6};
```

It is instantiated in the parameter file with

```
param y:= 4 0
          5 1
          6 3;
```

One writes alternatively the value of the index and the corresponding value of the vector. Consider now a parameter y defined by

y	0	1	2
1	4	5	6
2	5	6	7

where the first index runs from 1 to 2 and the second one from 0 to 2. It is declared in the model file with

```
param y{1..2,0..2};
```

and instantiated in the parameter file with

```
param y:= 1 0 4
          1 1 5
          1 2 6
          2 0 5
          2 1 6
          2 2 7;
```

Alternatively, y can be instantiated as follows:

```
param y: 0 1 2 :=
        1 4 5 6
        2 5 6 7;
```

3. OPERATIONS INVOLVING SETS

3.1 Constraints

Consider an optimization problem involving the following constraints:

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0.$$

This can be programmed in the model file as follows:

```
s.t. g{j in 1..3}: x[j] >= 0;
```

3.2 Sums

Sums can be programmed with the keyword `sum`, followed by an index set in brackets. Consider for example the problem

$$\inf_{x\in\mathbb{R}^8} \sum_{i=1}^{8} \Big(x_i^2 + 2x_i\Big).$$

The model file is:

```
var x{1..8};
minimize f: sum{i in 1..8} (x[i]^2 + 2*x[i]);
```

Note that here the use of parentheses is mandatory.

3.3 Instantiating parameters

The parameter $x = (1, 2, 3, 4, 5)$ can be declared and instantiated with one command in the model file:

```
param x{i in 1..5} := i;
```

3.4 Initializing an optimization variable

For example:

```
let {i in 1..3} x[i]:= 2;
```

Remark. The syntax is the same for constraints parametrized by two indices, for sums over a pair of indices in two sets, for matrix-valued parameters and variables. Consider for example:

```
param z{i in 1..5,j in 1..3} := i+ 2*j;
```

4. CALCULATED VARIABLES

Some optimization problems may involve variables which can be written in function of some other variables (and/or parameters) in an explicit fashion. They are called calculated variables. Consider for example:

$$\begin{aligned} \inf_{(x,y)\in\mathbb{R}^2} \; & x^2 + y^2 \\ \text{s.t. } & y = x + 3; \end{aligned}$$

The corresponding model file is

```
var x;
var y;
minimize f: x^2 + y^2;
s.t. g: y= x+3;
```

Here the variable y could be (mathematically) eliminated, which would allow to simplify the resolution of the problem. This elimination can be realised with AMPL with the following commands:

```
var x;
var y=x+3;
minimize f: x^2 + y^2;
```

Sometimes, it is convenient to introduce calculated variables to improve the readability of the program without slowing down the resolution of the problem. Consider the optimization problem

$$\inf_{(a,b)\in\mathbb{R}^2} \sum_{i=1}^{N} \big(a + bx_i - y_i\big)^2,$$

where $N \in \mathbb{N}$, $x \in \mathbb{R}^N$, and $y \in \mathbb{R}^N$ are parameters. The model file can be written as follows:

```
param N;
param x{1..N};
param y{1..N};
var a;
var b;
minimize f: sum{i in 1..N} (a+b*x[i]-y[i])^2;
```

The optimization problem is equivalent to:

$$\begin{aligned} \inf_{(a,b)\in\mathbb{R}^2,\, z\in\mathbb{R}^N} \; & \sum_{i=1}^{N} z_i^2 \\ \text{s.t.: } & z_i = a + bx_i - y_i, \quad \forall i = 1, \ldots, N. \end{aligned}$$

The variable z can be treated as a calculated variable:

```
param N;
param x{1..N};
param y{1..N};
var a;
var b;
var z{i in 1..N}= a + b*x[i]- y[i];
minimize f: sum{i in 1..N} (z[i]^2);
```

5. SYNTAXIC COMMENTS

The following symbols should not be mistaken:

- The symbol `:` is used for the definition of the cost function and the constraints.
- The symbol `=` is used in equality constraints and calculated variables.
- The symbol `:=` is used for instantiating parameters or for initializing optimization variables.

The different delimiters play different roles:

- The parentheses `(` and `)` are used to prioritize mathematical operations.
- The brackets `[` and `]` are used to access to the component of a vector or a matrix.
- The curly brackets `{` and `}` are used whenever an index set is involved (declaration of vectors and parameters, sums, parametrized constraints).

6. DEBUGGING YOUR PROGRAM

Here is a list of common mistakes.

(1) The model and data files have not been saved before loading the script file.
(2) A model is loaded while the previous has not been erased (with the `reset` command).
(3) The character `;` is missing at one or several places.
(4) Some optimization variables and some parameters have not been declared.
(5) Some optimization variables have been declared without the key word `var`.
(6) Some parameters have been declared without the key word `param`.
(7) Some parameters have been instantiated without the key word `param`.
(8) A space has been introduced between `:` and `=`. One should write `:=` and not `: =` for the instantiation of parameters.
(9) For parameterized constraints, the index set must be put before `:` as for example:
```
g {i in 1..n}: x[i] >= 0;
```
(10) Misuse of parentheses. The following commands are not understood by AMPL:
```
sum ( {i in 1..n} x[i] );
g: (x >= 0);
```
(11) The character `*`, necessary for multiplications, is missing.
(12) Use of a strict inequality constraint.
(13) A vector-valued parameter (or optimization variable) has been declared as real-valued.
(14) Undefined parameters. AMPL does not understand:
```
param x{1..n};
param n;
```
The parameter n must be declared before x.