

IEOR 240 DISCUSSION

2022 Fall

Shunan Jiang

AMPL's Role

- We write a linear programming problem as

$$\begin{aligned}
 &\min(\max) \quad \sum_{i=1}^n c_i x_i \\
 \text{s.t.} \quad &\sum_{i=1}^n a_{1i} x_i \quad \left(\begin{smallmatrix} \leq \\ = \\ \geq \end{smallmatrix} \right) b_1 \\
 &\sum_{i=1}^n a_{2i} x_i \quad \left(\begin{smallmatrix} \leq \\ = \\ \geq \end{smallmatrix} \right) b_2 \\
 &\quad \vdots \\
 &\sum_{i=1}^n a_{mi} x_i \quad \left(\begin{smallmatrix} \leq \\ = \\ \geq \end{smallmatrix} \right) b_m
 \end{aligned}$$

x_i - decision variables,

a_{ji}, c_i, b_j - problem data

but the algorithm needs input like

$$c = [2.5, 2.0, 1.7, 0, 0, 0]$$

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & -30 & 0 & 0 \\ 0 & 1 & 0 & 0 & -10 & 0 \\ 0 & 0 & 1 & 0 & 0 & -15 \end{bmatrix}$$

$$b = [15 \quad 20 \quad 0 \quad 0 \quad 0]'$$

AMPL's Role

- We write a linear programming problem as

$$\begin{aligned}
 &\min(\max) \quad \sum_{i=1}^n c_i x_i \\
 \text{s.t.} \quad &\sum_{i=1}^n a_{1i} x_i \begin{pmatrix} \leq \\ = \\ \geq \end{pmatrix} b_1 \\
 &\sum_{i=1}^n a_{2i} x_i \begin{pmatrix} \leq \\ = \\ \geq \end{pmatrix} b_2 \\
 &\vdots \\
 &\sum_{i=1}^n a_{mi} x_i \begin{pmatrix} \leq \\ = \\ \geq \end{pmatrix} b_m
 \end{aligned}$$

x_i - decision variables,

a_{ji}, c_i, b_j - problem data

AMPL does the
translating:



but the algorithm needs input like

$$c = [2.5, 2.0, 1.7, 0, 0, 0]$$

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 & 0 \\ 1 & 0 & 0 & -30 & 0 & 0 \\ 0 & 1 & 0 & 0 & -10 & 0 \\ 0 & 0 & 1 & 0 & 0 & -15 \end{bmatrix}$$

$$b = [15 \quad 20 \quad 0 \quad 0 \quad 0]'$$

Installation Instructions

1. Download AMPL

- MacOS: <http://ampl.com/demo/amplide.macosx64.tgz>
- Windows: <http://ampl.com/demo/amplide.mswin64.zip>
- Linux: <http://ampl.com/demo/amplide.linux64.tgz>

2. Extract the archive.

3. In the `amplide` folder, you should find the `amplide` application.

4. Open the `amplide` application.

5. Type into the console: `printf "Hello World!\n";`

6. Press Enter.

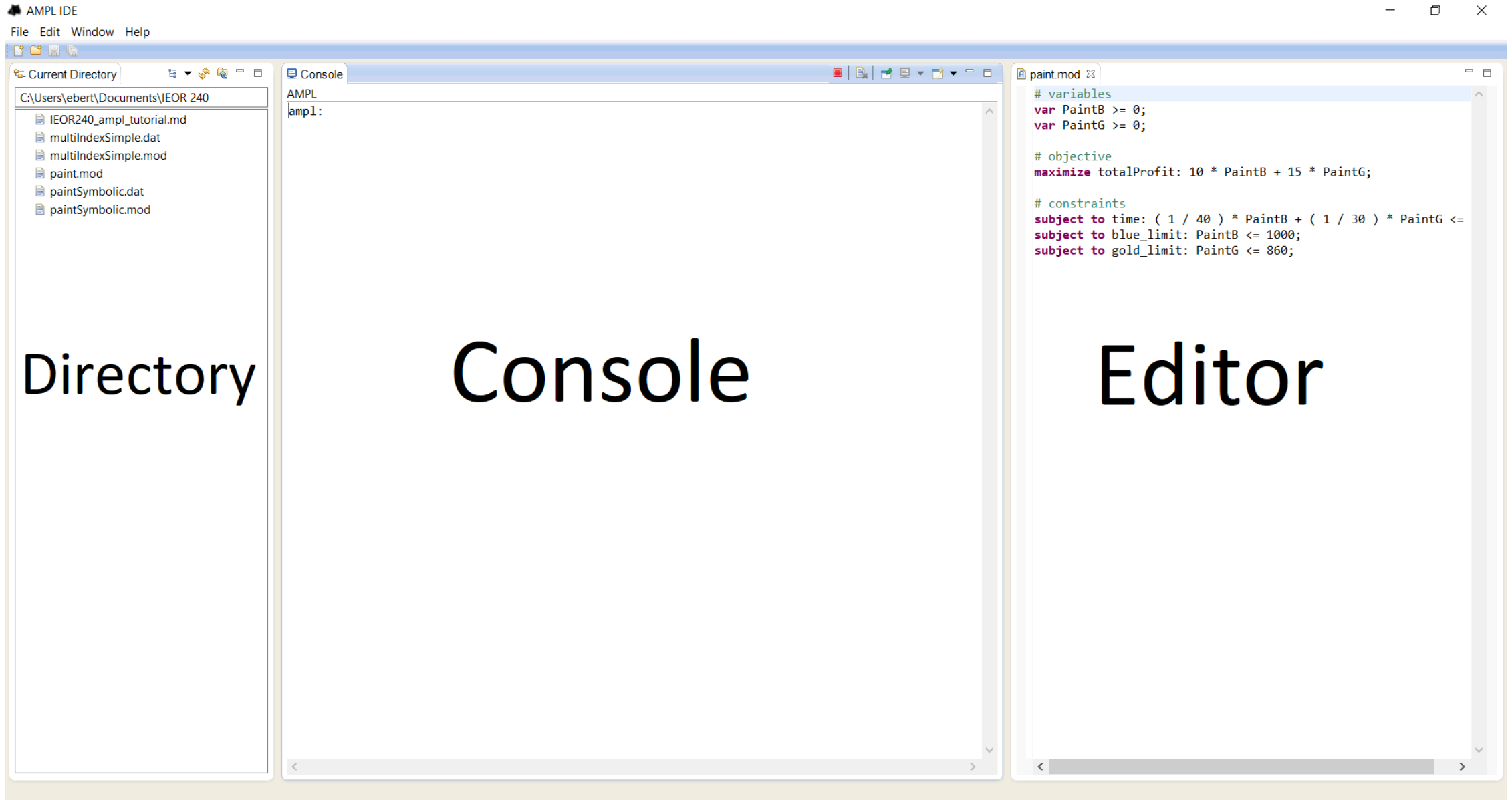
7. If it prints `Hello World!` , then you are done!

Special Instruction for Mac OSX

Special note for users of macOS 10.12 Sierra: As a side-effect of a new security feature introduced with this version, you may see an error message beginning `The IDE cannot find the AMPL executable...` To fix this problem, quit the IDE application and then follow these steps:

- (1) In your `amplide.macosx64` folder, find the `Amplide` file (with a cat's head icon).
- (2) Drag the `Amplide` file to your desktop.
- (3) Drag the `Amplide` file back into the `amplide.macosx64` folder.
- (4) Then double-click the file icon to start the AMPL IDE again.

AMPL IDE



Modelling: Berkeley Paint Company

Berkeley Paint Company makes two colors of paint: Blue and Gold

The type of paints have the following characteristics (per gallon):

Paint	Blue	Gold
Profit (p_i)	10	15
Production/hour (r_i)	40	30
Maximum demand (u_i)	1000	860

Berkeley Paint Company has 40 hours of production capacity available to produce paint.

Formulate an LP to help Berkeley Paint Company maximize profit.

Model

x_1 = Amount of Blue paint to produce

x_2 = Amount of Gold paint to produce

$$\begin{array}{ll}\max & 10x_1 + 15x_2 \\ \text{s.t.} & (1/40)x_1 + (1/30)x_2 \leq 40 \\ & x_1 \leq 1000 \\ & x_2 \leq 860 \\ & x_1 \geq 0 \\ & x_2 \geq 0\end{array}$$

Formulating the model in AMPL

```
# variables
var PaintB;
var PaintG;

# objective function
maximize totalProfit: 10 * PaintB + 15 * PaintG;
# constraints
subject to time: (1/40)*PaintB + (1/30)*PaintG <= 40;
subject to blue_limit: 0 <= PaintB <= 1000;
subject to gold_limit: 0 <= PaintG <= 860;
```

Another way to write it

```
# variables
var PaintB >= 0;
var PaintG >= 0;

# objective function
maximize totalProfit: 10 * PaintB + 15 * PaintG;
# constraints
subject to time: (1/40)*PaintB + (1/30)*PaintG <= 40;
subject to blue_limit: PaintB <= 1000;
subject to gold_limit: PaintG <= 860;
```

Save as `paint.mod`

Comments on Syntax

- # starts a comment
- All lines of code end with a ;
- Variables are declared with var
- The objective function has the following format:
maximize <name>: <objective>; or
minimize <name>: <objective>;
- The constraints have the following format:
subject to <name>: <constraint>;
- Names must be unique.
Also, a variable and constraint cannot have the same name.
- AMPL is case sensitive. Keywords must be lower case.

Setup AMPL

Type the following commands into the **console**:

1. Set the solver to be CPLEX:

```
option solver cplex;
```

There are many solvers included with AMPL but we will mostly use CPLEX

Time to solve!

Type the following commands into the **console**:

1. Load the model:

```
model paint.mod;
```

2. Solve!

```
solve;
```

3. Output:

```
CPLEX 12.7.0.0: optimal solution; objective 17433.33333  
0 simplex iterations (0 in phase I)
```

What is the solution?

- Display the objective function, constraint or variable:

```
display <name>;
```

For example:

```
display totalProfit;
```

- Display all variables:

```
display _varname, _var;
```

Reset AMPL

- ```
reset;
```

# Modelling: Berkeley Paint Company

Berkeley Paint Company makes two colors of paint: Blue and Gold

The type of paints have the following characteristics (per gallon):

| Paint                     | Blue | Gold |
|---------------------------|------|------|
| Profit ( $p_i$ )          | 10   | 15   |
| Production/hour ( $r_i$ ) | 40   | 30   |
| Maximum demand ( $u_i$ )  | 1000 | 860  |

Berkeley Paint Company has 40 hours of production capacity available to produce paint.

Formulate a **symbolic** LP to help Berkeley Paint Company maximize profit.

# Symbolic model

$n$  = number of paints to produce

$x_i$  = Amount of paint  $i$  to produce

$$\begin{aligned} \max \quad & p_1 x_1 + p_2 x_2 + p_3 x_3 + \dots p_n x_n \\ \text{s.t.} \quad & \frac{1}{r_1} x_1 + \frac{1}{r_2} x_2 + \frac{1}{r_3} x_3 + \dots + \frac{1}{r_n} x_n \leq t \\ & 0 \leq x_i \leq u_i \text{ for } i = 1 \dots n \end{aligned}$$

Or equivalently:

$$\begin{aligned} \max \quad & \sum_{i=1}^n p_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^n \frac{1}{r_i} x_i \leq t \\ & 0 \leq x_i \leq u_i \text{ for } i = 1 \dots n \end{aligned}$$



# Writing a symbolic model in AMPL

```
parameters
param n;
param capacity;
param profit{i in 1..n};
param r{i in 1..n};
param maxDemand{i in 1..n};

variables
var paint{i in 1..n} >= 0;

objective function
maximize totalProfit:
 sum{i in 1..n} profit[i]*paint[i];

constraints
subject to time:
 sum{i in 1..n} (1 / r[i]) * paint[i] <= capacity;
subject to demand_limit {i in 1..n}:
 paint[i] <= maxDemand[i];
```

Save as `paintSymbolic.mod`

# Syntax comments

- Use more elaborate names than in your LP.
- `param` is used to define a parameter.
- We define an indexed variable / parameter with:
  - `<name>{<index> in <range>}`
  - Typical indices are: `i,j,k`
  - Range: `1..n` **TWO dots!**
- Use `<name>[2]` to access the second index.
- Summation are defined similarly:  
`sum{<index> in <range>}`
- Indexed constraints are defined with:  
`subject to <name> {<index> in <range>}:`

**What is missing? Data!**

# Specifying data in AMPL

```
param n := 2;
param capacity := 40;

param profit :=
 1 10
 2 15;
param r :=
 1 40
 2 30;
param maxDemand :=
 1 1000
 2 860;
```

Save as `paintSymbolic.dat`

# Syntax comments:

- Defining the value of a simple parameter:

```
param <name> := <value>;
```

- Defining the value of an indexed parameter:

```
param <name> :=
 <index> <value>
 ...
 <index> <value>;
```

# Parameters with multiple indices:

- Definition: `<name>{<ind1> in <range1>, <ind2> in <range2>}`
- Accessing an index: `<name>[<index1>,<index2>]`
- Example definition:

```
var paint{i in 1..m, j in 1..n} >= 0;
```

- Example constraint:

```
subject to demandConstraint {j in 1..n}:
 sum{i in 1..m} paint[i,j] = demand[j];
```

- Data with multiple indices:

```
param <name> :=
 <index1> <index2> <value>
 ...
 <index1> <index2> <value>;
```

# Solving a symbolic model

- Load model:

```
model paint.mod;
```

- Load data:

```
data paintSymbolic.dat;
```

- Solve: `solve;`

# Setup AMPL - Sensitivity Analysis

Type the following commands into the **console**:

1. Set the solver to be CPLEX:

```
option solver cplex;
```

2. Enable sensitivity analysis:

```
option cplex_options 'sensitivity';
```

3. Turn off presolve (needed for sensitivity analysis):

```
option presolve 0;
```

4. Load model and solve as usual.

```
model paint.mod;
```

```
solve;
```



# Sensitivity Analysis Outputs

- Display the objective function, constraint or variable:

```
display <name>;
```

For example:

```
display totalProfit;
```

- Display all variables:

```
display _varname, _var, _var.rc, _var.down, _var.current, _var.up;
```

- Display all constraints:

```
display _conname, _con, _con.slack, _con.up, _con.current, _con.down;
```