AMPL Quick Reference

AMPL is an *algebraic* mathematical programming language. One can write equations, inequalities and functions in the natural algebraic form using variables and mathematical operators. It provides an intuitive way of writing optimisation problems that are easy to read and modify by humans. For instances, unlike Scilab, we do not need to create any arrays for a linear optimisation problem. Instead we just write the linear inequalities. AMPL is **not** a solver. It can only be used to create optimisation models. We need additional software to solve these problems. For linear optimisation problems, we can use:

- Gurobi (http://www.gurobi.com)
- Cplex (http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/)
- CLP (https://projects.coin-or.org/Clp)
- Glpk (www.gnu.org/software/glpk/)
- and several others

Of these, Clp and Glpk are are open-source and free. Others are commercially available and proprietary. In our experiments, we will be using the Gurobi solver.

Using a text editor

- AMPL is very different from Scilab both in the way it is used and also its performance. Unlike Scilab, there is no "AMPL window" or working environment. You will have to write AMPL models using a text-editor and run them in a different window running the AMPL-shell.
- AMPL's model files contain the description of the optimisation problem. They should be written using a text-editor like 'gedit', 'vim' or 'emacs'. **Never** use open-office, ooffice, soffice, MS-Word or any other office-document editors to edit these files.
- To start gedit, just type at linux command prompt
  `gedit &`
- It will open a text editor in a new window. You may start writing any text.
- Click save to save the current file. Click File→Save As to save the file under a different name.

Writing an AMPL .mod model file

- Everything in a line following the '#' symbol is ignored. Thus, # may be used to write a comment.
- A statement in AMPL may extend over multiple lines. A semicolon ';' must follow every statement to mark its end.
- All variables must be declared using the keyword 'var' in the beginning. Lower and/or upper bounds are optional. If a bound is not specified then it is assumed to be infinite. e.g.
  var x;
  var y >= 1;
  var x23 <= 100;
  var x45 <= 12, >= 1;

Starting, exiting and running AMPL

- Please note again that AMPL is **not** a solver. It is just a modeling tool. You have to tell AMPL which solver to use for solving your problem. For linear problems, we will use the solver Gurobi. It is one of the faster LP solvers available today.
- Before starting AMPL, make sure you 'cd' into the directory where you saved the .mod file.
- Type at the command prompt
  `ls`
- If you do not see the .mod file that you wish to solve, first 'cd' into the right directory. Then go to the next step.
- To start AMPL, just type at the command prompt
  `ampl`
- You should see the AMPL prompt
  `ampl:`
- Every AMPL command or statement must end in a semicolon ';'.
- To quit AMPL, type
  `ampl: exit;`
- Notice the ';' at the end. Generally, if a ';' is missing, ampl will prompt will have a question mark at the end
  `ampl?`
  If you see this prompt, just type a semicolon and press enter. The previously incomplete command will then be executed.
- Usually, an optimisation model is written a '.mod' file.
- To run a '.mod' file, for instance 'test.mod', just type at the AMPL prompt
  ```
  ampl: model test.mod;
  ampl: option solver gurobi;
  ampl: solve;
  ```
- To clear all previously defined variables, arrays, sets and data, you should use the reset command
  `ampl: reset;`

Additional reference information for AMPL is available in the freely available first chapter of the AMPL book (http://www.ampl.com/BOOK/ch1-2.pdf) and also available on Moodle. Some examples illustrating advanced uses of AMPL are available at the netlib repository (http://www.netlib.org/ampl/models/ and http://www.netlib.org/ampl/looping/.)