AN INTRODUCTION TO APML AND APPLICATION TO MIMO CONTEXT

HAMID MEGHDADI

ABSTRACT. In this document I will brief a brief explanation of AMPL and its utilization for the signal to noise ratio optimization in a MIMO two hop network as describled in [1]

1. INTRODUCTION

AMPL (for A Mathematical Programming Language) is a modeling language for mathematical programming. It is a commercial utility but a student version with fair limitation exists and is free to be downloaded from `www.ampl.com`. AMPL can be used further to pass the model to a solver, free or commercial, to be optimized, that is to be maximized or minimized with respecting a number of constraints. It is important to note that AMPL does not *solve* the problem by itself, it is simply used to model a system and to pass the system to a solver.

AMPL can define system variables, objective, and constraints. Constraints may be linear or non-linear, equalities or inequalities, ...

AMPL can be called from MATLAB and can save the results in a file that can be read later. AMPL is an easy to learn high level language.

2. USING AMPL

In this section a brief understanding of AMPL is given. Once the AMPL executable is installed and available, we call it from the command line and we are presented with the AMPL prompt. Valid AMPL commands may be entered at the AMPL prompt and must finish with a `;` sign.

2.1. AMPL files.

AMPL uses three file types:

- **.mod files:** are used to store system model containing constraints, objective and variables. Equations are written in AMPL syntax and can contain a large variety of functions and operators. Groups of simmilar constrains and variables can be formed easily. A model file must start with a `model;` command.
- **.dat files:** define system data like parameter values, initial points, etc. A data file must start with a `data;` command.
- **.ampl files:** are AMPL script files. They are sort of batch files that are used to encapsulate different commands and execute them one after another. The command file can be passed to AMPL as an argument. For example the command `AMPL myModel.ampl` executes AMPL commands in the file `myModel.ampl` one by one.

Key words and phrases. Ampl, Spnot, Optimization.

In the next sections I will give a brief description of AMPL syntax and structure.

2.2. **AMPL syntax.**

In AMPL, comments are declared with a `#` sign.

A variable is declared using `var` keyword. So `var x;` declares a variable called `x`. Variables are the system parameters that may be adjusted in order to achieve the objectives. A set of variables that satisfy all constraints are called a *candidate*, which is a potential solution to the problem. The solver will choose the best candidate with respect to the objective.

`let` is used to assign values to variables and parameters. `x:=3.14` is used to assign the value 3.14 to the variable `x`.

`param` is used to declare system constants and parameters. Parameters are useful when we want to define a general model and change the parameter values to see what happens to the results. `minimize` (respectively `maximize`) is used to define the target function to be minimized (resp. maximized) with respect to the constraints. `maximize area : a*b;` tells AMPL that we search to maximize the function *area* of two variables a and b.

Constraints are defined with the keyword `subject to`.
For example `subject to perimeter: 2*(a+b)<=20` means that we want that a function called *perimeter* which is defined by $2(a+b)$ be always less or equal to 20.

`display x` is used to ask AMPL to display `x`.

`sum` is used to introduce summation operator, $\sum_{i=1}^{N} a_i$ is written in APML as `sum{i in 1..N} a[i]`.

2.3. **A simple example.**

In this section we will use AMPL to solve a chain problem. A chain of m links is fixed at its two ends. we want to find the stable position of the chain.

2.3.1. *Mathematical formulation.*

A system is stable when its potential energy is minimum. We consider that the chain is fixed at one end at the point $(0,0)$ and at the other end at (x_e, y_e). We define m horizontal increments x_i and m vertical increments y_i. It means that kth point of chain is located at $\left(\sum_{i=1}^{k} x_i, \sum_{i=1}^{k} y_i\right)$. The sum of links must equal the total length of the chain: $\sum_{i=1}^{m} x_i^2 + y_i^2 = L^2$ with $L = length/m$. The second end also must be at (x_e, y_e), that is $\sum_{i=1}^{m} x_i = x_e$ and $\sum_{i=1}^{m} y_i = y_e$. We want to minimize the sum of potential energies of each link that is $\sum m_i g h_i$. Since all links have the same mass, and gravity is a constant we can minimize the sum of altitudes:

$$\text{energy} = \sum_{k=1}^{m} h_k = \sum_{k=1}^{m} \left(\sum_{i=1}^{k} y_i - \sum_{i=1}^{k-1} y_i \right)$$

2.3.2. *AMPL code.*

We define a `chain.mod` file containing:

```
# AMPL
# Chain problem

model;

# One end point is fixed at (0,0), other is fixed at (xe,ye)
param xe;
param ye;
```

```
# Chain length
param length >0;

# Number of links
param m >= 2 integer;

# Link length
param L := length/m;

# Variables: horizontal increments
var x {1..m};

# Variables: vertical increments
var y {1..m};

# Objective function
minimize energy:
 .5*sum{k in 1..m} ((sum {i in 1..k} y[i])+(sum{i in 1..k-1} y[i]));

# Nonlinear constraint:  links of constant length
subject to link_length {k in 1..m}: x[k]**2 + y[k]**2 = L**2;

# Linear constraints: position of the second endpoint
subject to endy: sum {k in 1..m} y[k] = ye;
```

We must define a `chain.dat` file containing parameter values and starting point:

```
# AMPL
# Data for the chain problem

data;

# End point coordinates
param xe := 2;
param ye := 0;

# Chain length
param length := 4;

# Number of links
param m := 25;

# Starting point of the minimization method
let {k in 1..m} x[k] := length/m;
let {k in 1..m} y[k] := 0;
```

We can use AMPL to send this system to SNOPT and solve it:

```
D:\ampl\AtelierOptim\mswin>ampl
ampl: model chain.mod;
ampl: data chain.dat;
ampl: option solver snopt;
ampl: solve;
SNOPT 7.2-8 : Optimal solution found.
112 iterations, objective -22.76721608
Nonlin evals: constrs = 85, Jac = 84.
ampl: display x,y;
:       x           y        :=
1    0.0372161   -0.155612
2    0.0403921   -0.154818
...
```

2.3.3. *AMPL script.* We can use AMPL scripts to do the above commands with less typing. a `.ampl` file is file that consists AMPL commands. It can be used to spare user from repeating long AMPL commands over and over. It suffices to create a `chain.ampl` file as follows and to type `ampl model.ampl` at the command prompt

```
model chain.mod;
data chain.dat;
option solver snopt;
solve;
display x,y;
```

2.3.4. *MATLAB link.* AMPL can be called from MATLAB. This helps user to do his programming in MATLAB and use APML to solve an optimization problem. The interconnection is done using the text files:

We need an APML command file:

```
# AMPL
# Solve the chain problem

# Read model file
model chain.mod;

# Read data file
data chain.dat;

# Select a solver;
option solver snopt;

# Solve the problem
```

```
solve;

# Display results
display energy;
display x;
display y;

# Write data to a file
printf "%5.2f\n%5.2f\n", xe, ye >chain.res;
printf "%3.0f\n", length >chain.res;
printf "%4i\n", m >chain.res;
printf {k in 1..m} "%14.6e\n", x[k] >chain.res;
printf {k in 1..m} "%14.6e\n", y[k] >chain.res;
```

No we create a MATLAB program and execute it. The AMPL files must be in the same directory as MATLAB program. Data is transfered between MATLAB and AMPL using text files in ASCII format.

```
% MATLAB
% Chain problem

% solve the problem
dos('ampl chain.ampl');

% read the results form a file
fid = fopen('chain.res','r');
v = fscanf(fid,'%f\n');
xe = v(1);
ye = v(2);
le = v(3);
m = v(4);
x = v(5:4+m);
y = v(5+m:4+2*m);
fclose(fid);

% plot the chain
cx = cumsum(x);
cy = cumsum(y);
cx = [0;cx];
cy = [0;cy];
plot(cx,cy,'or','LineWidth',2,'MarkerSize',10)
hold on
plot(cx,cy,'-b','LineWidth',2)
plot([0,xe],[0,ye],'xk','LineWidth',4,'MarkerSize',14)
axis square
title([num2str(m) ' links'],'FontSize',18)
hold off
```

We will obtain:

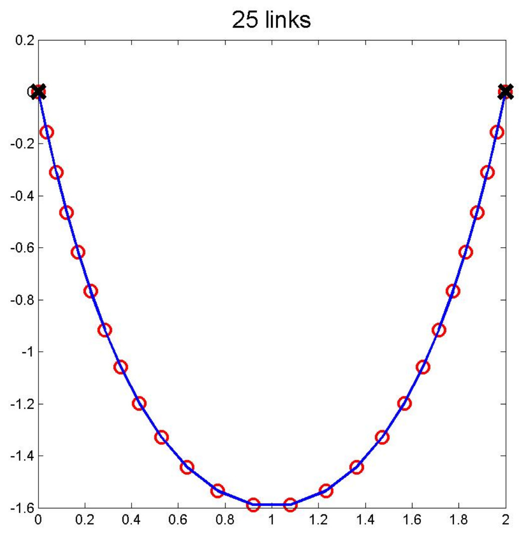

Figure 1. Results obtained for a 25 link chain

References

[1] Ahmed Zahoor et al, *Beamforming Applied to New Cooperative Decode-and-Forward relay Schemes*, ENSIL/VTC, 2007

E-mail address, Hamid Meghdadi: `hamid.meghdadi@gmail.com`