AMPL: A Mathematical Programming Language

Anas Abdelkarim
Electromobility Research Group
Technische Universität Kaiserslautern

Outlines

- **Introduction**
- What is AMPL?
- AMPL features.
- The Syntax
- Run AMPL

- **Examples**
- Example 1:   general picture of using AMPL
- Example 2 : AMPL for modeling MPC
- Example 3: advanced example of MPC

- **Installation**

Introduction

- **What is AMPL**

- AMPL is high-level algebraic modelling language to describe, implement and solve the optimization problems.
- AMPL invokes the solvers (optimization algorithms) to find the solution of OP, i.e., AMPL is an interface environment not solving algorithms.

Introduction

- **AMPL features**

- AMPL syntax is similar to the mathematical notation of optimization problems
- It supports a lot of solvers:

  CONDOR, CONOPT, Couenne, **CPLEX**, DONLP2, FilMINT, FILTER, MINLP, FortMP, FSQP, Gecode, **Gurobi**, **IPOPT**, JaCoP, KNITRO, LANCELOT, L-BFGS-B, LGO, LINDO, Global, LOQO, LP_SOLVE, MINLP, **MINOS**, MINTO, MOSEK, NPSOL, NSIPS, OOQP, PATH, PCx, PENNON, RAPOSa, SCIP, SNOPT, SOPT, Sulum, TRON, WSAT(OIP), XA, XLSOL, LS-XLSOL, Xpress, ACRS, ALGENCAN, BARON, BLMVM, Bonmin, BPMPD, CBC.

  https://ampl.com/products/solvers/all-solvers-for-ampl/

- It has IDE (Integrated Development Environment) supports Microsoft Windows, Linux and, macOS.
- It has API (Application Programming Interface) for Python, R, C++, C#, Java, MATLAB
- Treats large scale optimization

Introduction

$a = 2$

$\underset{x}{\text{minimiz}}\ x^2$

$\text{subject to } x \geq a$

```
model;
param a;
var x;
minimize cost: x**2;
s.t.          restriction: x >= a;
data;
 param a := 2;
```

- **Syntax**

- Parameters:

  param parameter_name;

- Variable:

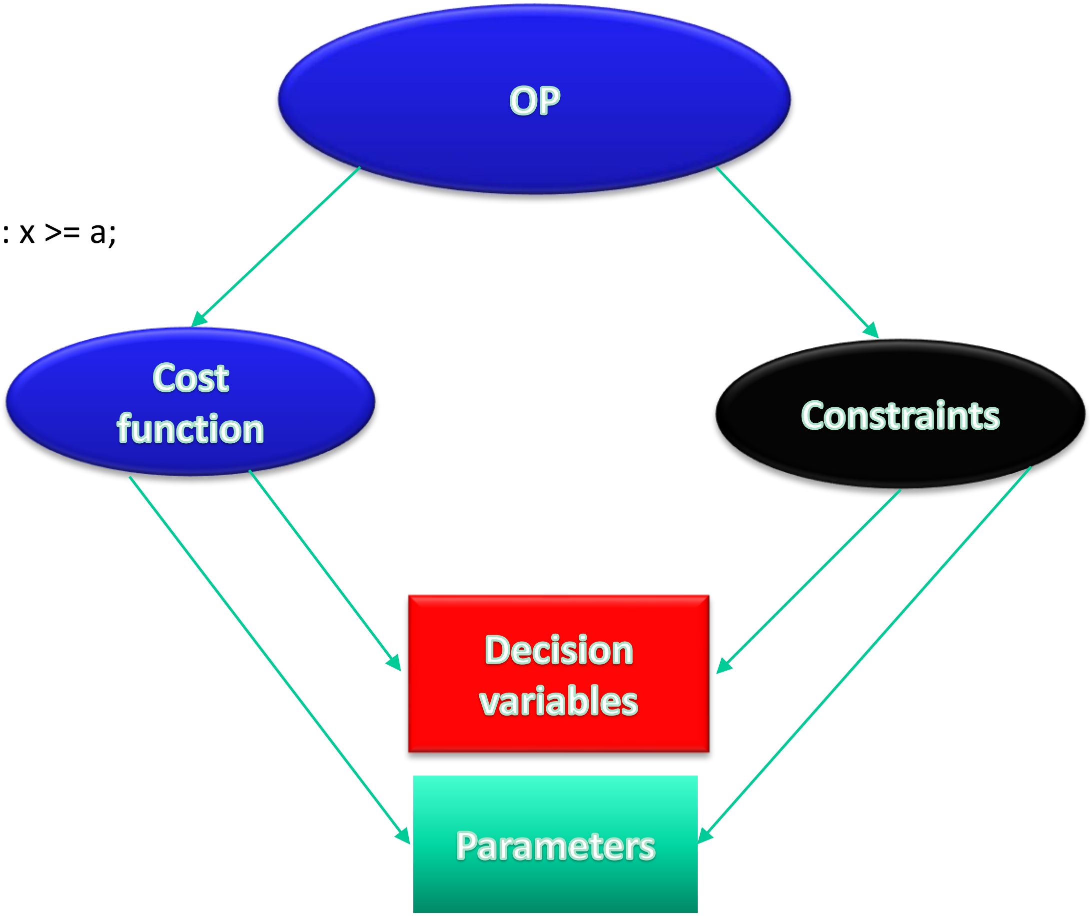
  var variable_name;

- Cost function:

  minimized/maximize cost_function_name:

- Constraints:

  (subject to/s.t.) constrait_name:



Introduction

$a = 2$

$\underset{x}{\text{minimiz}}\ x^2$

$\text{subject to } x \geq a$

```
model;
param a;
var x; # the decision variable
minimize cost: x**2;
s.t.         restriction: x >= a;
data;
 param a := 2;
```

- **Syntax**
- Parameters:

  param parameter_name;
- Variable:

  var variable_name;
- Cost function:

  minimize/maximize cost_function_name:
- Constraints:

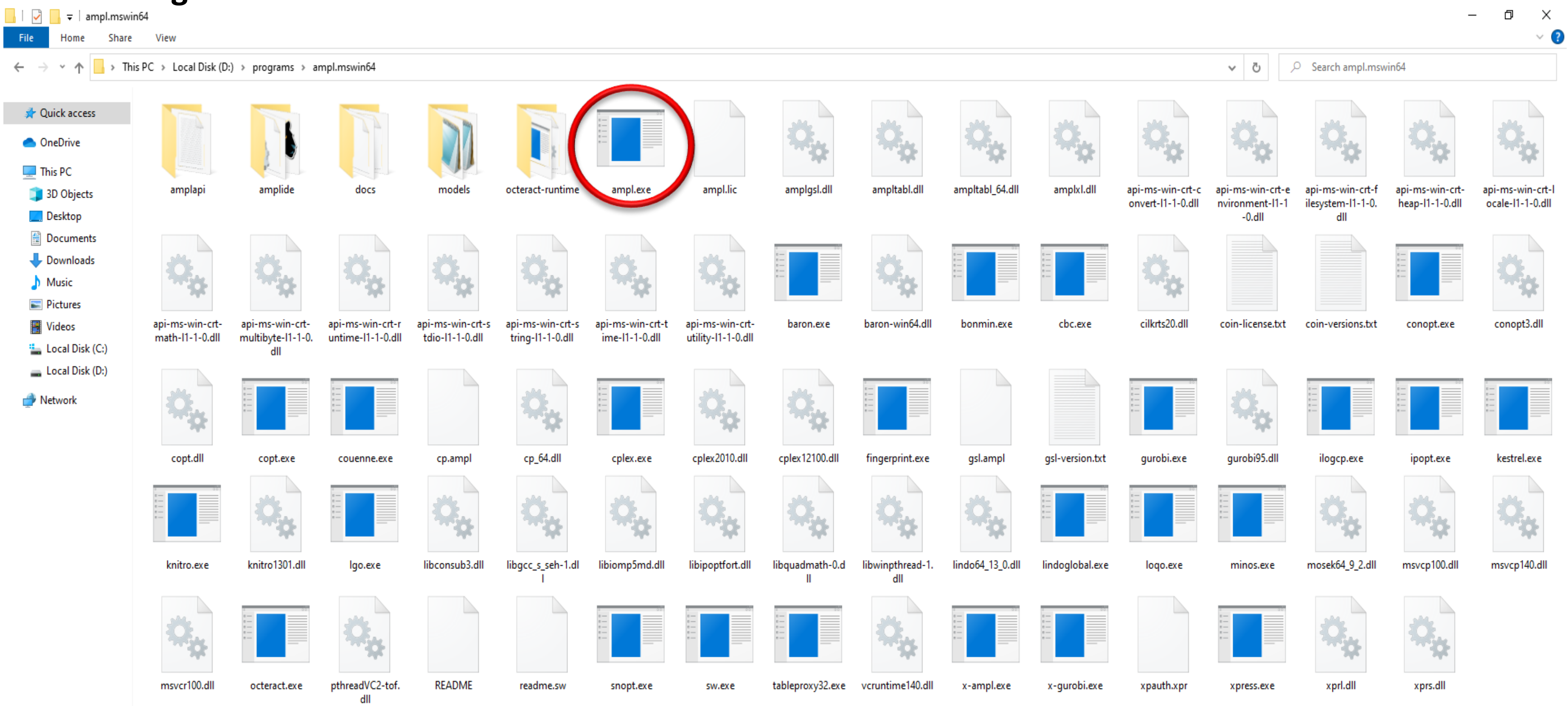
  (subject to/s.t.) constraint_name:

- **Remarks**
- Every line instruction must be terminated with ";".
- Line comments are preceded by the symbol "#".
- AMPL is "case-sensitive".
- Variable names must be unique

- **AMPL Modes**
- Model mode: this is where we define the parameters, variables, sets, cost function, and constraints.
- Data mode: this is where the data is given a value

Run AMPL

- **Using AMPL Terminal:**



Run AMPL

- **Using AMPL Terminal:**

```
D:\programs\ampl.mswin64\ampl.exe
ampl: reset;
ampl: ###########################
ampl: model;
ampl: param a;
ampl: var x; # the decision variable
ampl: minimize cost: x**2;
ampl: s.t.              restriction: x >= a;
ampl: ###########################
ampl: data;
ampl data: param a := 2;
ampl data: ###########################
ampl data: solve;
MINOS 5.51: optimal solution found.
0 iterations, objective 4
Nonlin evals: obj = 3, grad = 2.
ampl: expand cost, restriction;
minimize cost:
        x^2;

subject to restriction:
        x >= 2;

ampl: display x, a;
x = 2
a = 2
```

```
reset;
###########################
model;
param a;
var x; # the decision variable
minimize cost: x**2;
s.t.          restriction: x >= a;
###########################
data;
param a := 2;
###########################
solve;
expand cost, restriction;
display x, a;
###########################
option solver cplex;
solve;
display x;
###########################
option solver ipopt;
solve;
display x;
```

Introduction

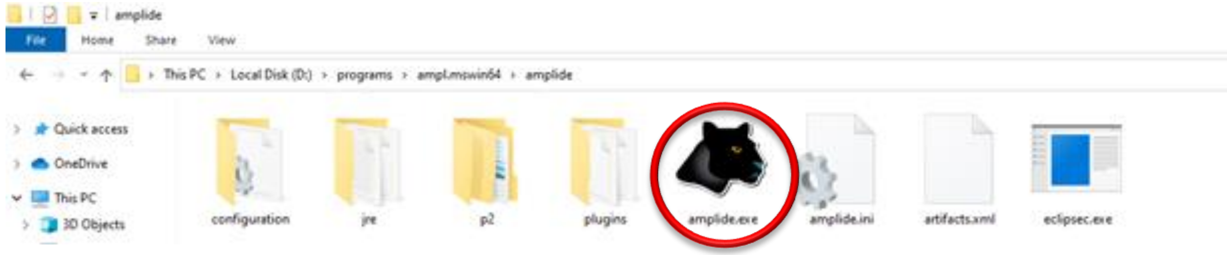
D:\programs\ampl.mswin64\ampl.exe

```
******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit http://projects.coin-or.org/Ipopt
******************************************************************************

This is Ipopt version 3.12.13, running with linear solver mumps.
NOTE: Other linear solvers might be more efficient (see Ipopt documentation).

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        1

Total number of variables............................:        1
                     variables with only lower bounds:        1
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  4.0803999e+00 0.00e+00 3.04e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  4.1177774e+00 0.00e+00 1.85e-02  -1.0 1.85e-02    -  1.00e+00 5.00e-01f  2
   2  4.0049122e+00 0.00e+00 2.83e-08  -2.5 2.80e-02    -  1.00e+00 1.00e+00f  1
   3  4.0001532e+00 0.00e+00 1.50e-09  -3.8 1.19e-03    -  1.00e+00 1.00e+00f  1
   4  4.0000018e+00 0.00e+00 1.84e-11  -5.7 3.79e-05    -  1.00e+00 1.00e+00f  1
   5  3.9999999e+00 0.00e+00 2.49e-14  -8.6 4.61e-07    -  1.00e+00 1.00e+00f  1

Number of Iterations....: 5

                                   (scaled)                 (unscaled)
Objective...............:   3.9999999225063294e+00    3.9999999225063294e+00
Dual infeasibility......:   2.4868995751603507e-14    2.4868995751603507e-14
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   2.5063293419916432e-09    2.5063293419916432e-09
Overall NLP error.......:   2.5063293419916432e-09    2.5063293419916432e-09


Number of objective function evaluations             = 11
Number of objective gradient evaluations             = 6
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 5
Total CPU secs in IPOPT (w/o function evaluations)   =      0.011
Total CPU secs in NLP function evaluations           =      0.000

EXIT: Optimal Solution Found.

Ipopt 3.12.13: Optimal Solution Found
```

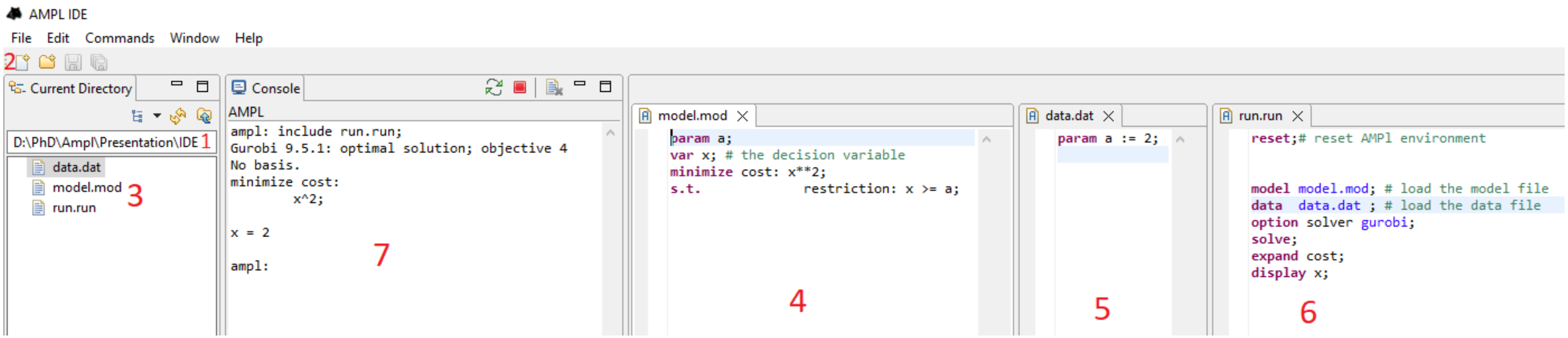
```
reset;
##############################
model;
param a;
var x; # the decision variable
minimize cost: x**2;
s.t.          restriction: x >= a;
##############################
data;
param a := 2;
##############################
solve;
expand cost, restriction;
display x, a;
##############################
option solver cplex;
solve;
display x;
##############################
option solver ipopt;
solve;
display x;
```

Run AMPL

- **Using AMPL IDE:**



$$a = 2$$
$$\underset{x}{\text{minimiz}}\ x^2$$
$$\textit{subject to}\ x \geq a$$



Run AMPL

- **AMPL on NEOS server:**
  - It is free internet-based service for solving OP
  - It is hosted by the Unvisited of Wisconsin, and provides access to more than 60 well-established solvers
  - The submitted tasks (jobs) to the NEOS sever run on distribute high-performance computers.

1. Go to: https://neos-server.org/neos/solvers/index.html

2. Choose the target solver and click on AMPL

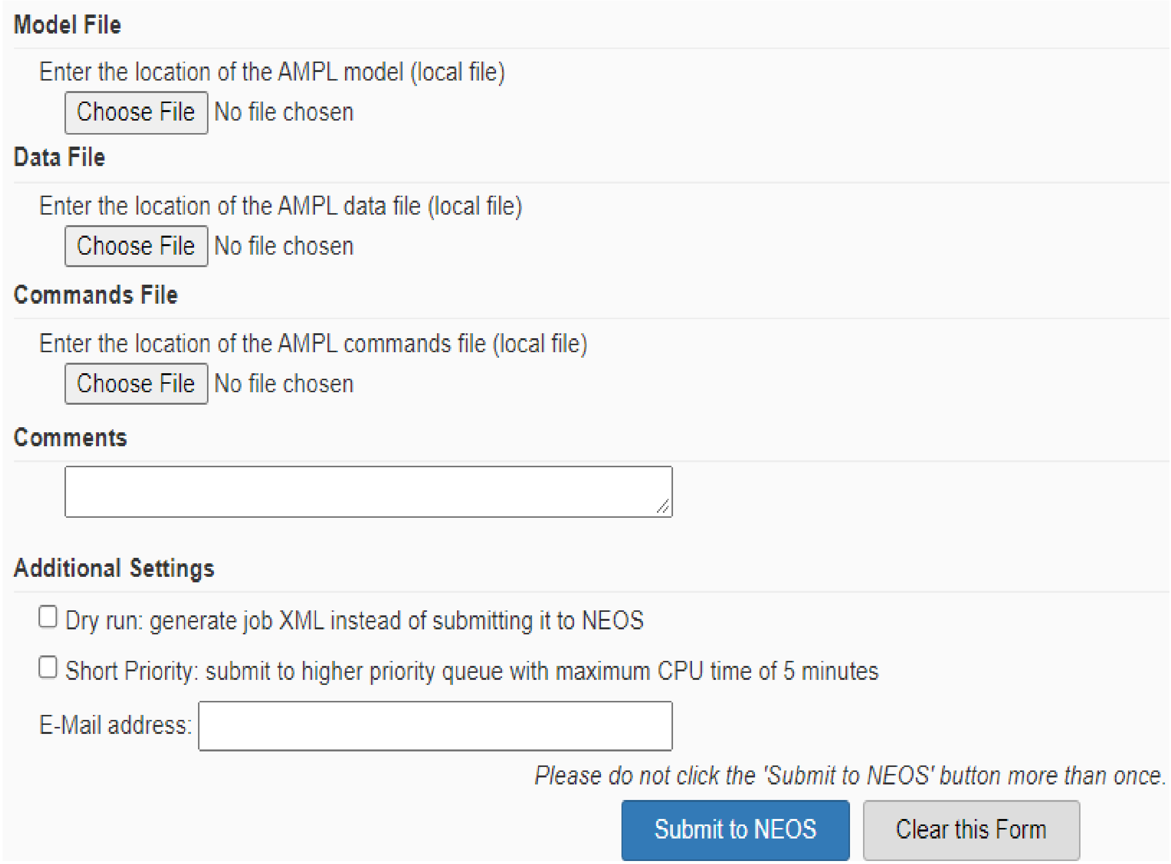
Mixed Integer Linear Programming

- Cbc [AMPL] [GAMS] [MPS]
- COPT [AMPL] [GAMS] [LP] [MPS] [NL]
- CPLEX [AMPL] [GAMS] [LP] [MPS] [NL]
- feaspump [AMPL] [CPLEX] [MPS]
- FICO-Xpress [AMPL] [GAMS] [MOSEL] [MPS] [NL]
- Gurobi [AMPL] [GAMS] [LP] [MPS] [NL]
- MINTO [AMPL]
- MOSEK [AMPL] [GAMS] [LP] [MPS] [NL]

Run AMPL

- **AMPL on NEOS server:**

  3. Upload AMPL files, fill the email
  4. Click on Submit to NEOS



Model File

Enter the location of the AMPL model (local file)

Choose File No file chosen

Data File

Enter the location of the AMPL data file (local file)

Choose File No file chosen

Commands File

Enter the location of the AMPL commands file (local file)

Choose File No file chosen

Comments

Additional Settings

☐ Dry run: generate job XML instead of submitting it to NEOS

☐ Short Priority: submit to higher priority queue with maximum CPU time of 5 minutes

E-Mail address:

Please do not click the 'Submit to NEOS' button more than once.

Submit to NEOS | Clear this Form

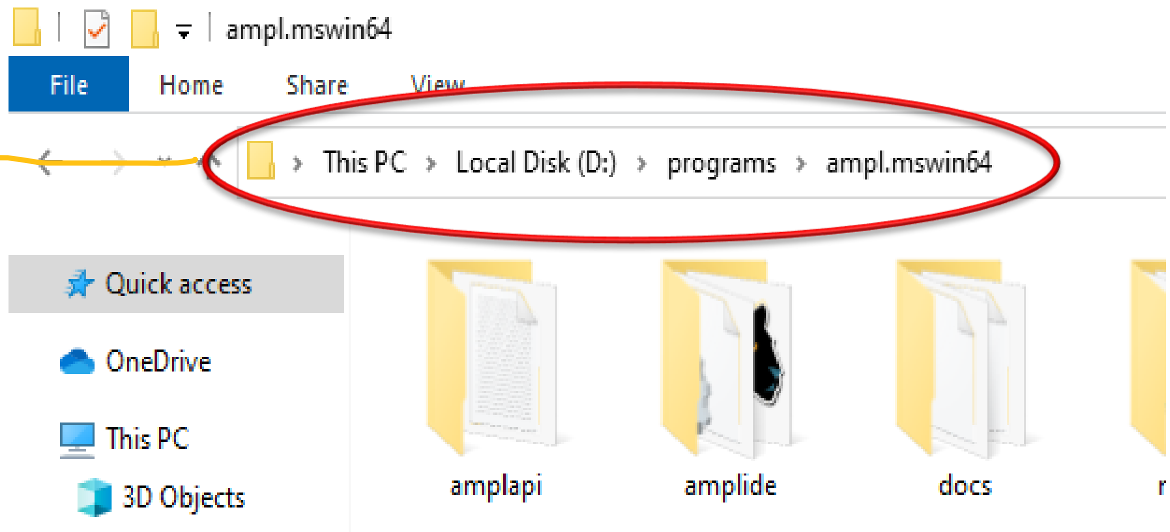
- **Remark:**
  - Run Should not have load *.mod, load *.dat, and option solver * code lines.

Run AMPL

- **AMPL on Matlab (after installation of API and customized codes)**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AMPL Environment %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
amplPathDir = 'D:\programs\ampl.mswin64';
%% add path for the customized codes
addpath(amplPathDir+ "\amplapi\matlab\customizedCodes")
run(amplPathDir + "\amplapi\matlab"+ "\setUp.m")  % setup API for matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

$$a = 2$$
$$\underset{x}{minimiz}\; x^2$$
$$subject\ to\ x \geq a$$



Anas Abdelkarim
Electromobility Research Group
Technische Universität Kaiserslautern

Run AMPL

- **AMPL on MATLAB – Method 1**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Method 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% run AMPL
ampl = AMPL(amplPathDir);
% read the model
ampl.read(['D:\PhD\Ampl\Presentation\IDE\model.mod'])
% read the data
ampl.readData(['D:\PhD\Ampl\Presentation\IDE\data.dat'])
% Choose the solver
ampl.eval('option solver gurobi;')
% solve the optimization problem
ampl.solve()
% extract the results:
[value, astatus,result,exitcode,message,expand] = ampl.getObj('cost');
x = ampl.getVarValue('x');
a = ampl.getParamValue('a');
% close AMPL
ampl.close
```

$$a = 2$$

$$\underset{x}{minimiz}\; x^2$$

$$subject\ to\ x \geq a$$

Run AMPL

- **AMPL on MATLAB – Method 2**

```
% run AMPL
ampl = AMPL(amplPathDir);
% model
ampl.eval('model; param a; var x;')
ampl.eval('minimize cost: x**2;')
ampl.eval('restriction: x >= a;')
%data
ampl.eval('data; param a:=2;')
% Choose the solver
ampl.eval('option solver gurobi;')
% solve the optimization problem
ampl.solve()
% extract the results:
[value, astatus,result,exitcode,message,expand] = ampl.getObj('cost');
x = ampl.getVarValue('x');
a = ampl.getParamValue('a');
% close AMPL
ampl.close
```

$$a = 2$$
$$\underset{x}{\text{minimiz}}\; x^2$$
$$\textit{subject to } x \geq a$$

Examples – Example 1

- **Optimal Control: Mixed Integer Programming**
  - Abdelkarim, A. and Zhang, P., 2020, September. Optimal Scheduling of Preventive Maintenance for Safety Instrumented Systems Based on Mixed-Integer Programming. In *International Symposium on Model-Based Safety and Assessment* (pp. 83-96). Springer, Cham.

For more details, go to folder:

amplPathDir\amplapi\matlab\AMPL-Codes-main\Examples\Optimal-scheduling_example

Examples – Example 1 (MODEL)

- **Optimal Control: Mixed Integer Programming**

$$\min_{\alpha(k),\gamma(k),p(k)} \frac{1}{N}\sum_{k=0}^{N-1}(p_3(k)+p_4(k)+p_5(k)+p_6(k))$$

$$\text{s.t.}\quad p(k+1)=\gamma(k)(Qp(k))+\alpha(k)(Q(M_A p(k)),$$

$$\sum_{k=0}^{N-1}\alpha(k)\leq n_f,\qquad \alpha(k)\in\{0,1\}$$

$$\alpha(k)+\gamma(k)=1,\qquad k=0,1,\cdots,N-1$$

$$p(0)=p_0,$$

```
param N , integer                          ; # horizon length
var alpha { k in 0..N-1 } , binary         ; # full maintenance mode
var gamma { k in 0..N-1 } , binary         ; # dynamics mode
var p {i in 1..6 , k in  0..N  } >= 0      ; # states vector
param n_f                                  ; # num. of full maintenance
param initial_values {j in 1 ..6}          ; # initial values
param Q {i in 1..6, j in 1..6}             ; # system matrix
param Ma {i in 1..6, j in 1..6}            ; # full maintenance matrix

 minimize cost_function:
1/(N)*sum { k in 0..N-1} ( p[3,k]  +   p[4,k]  +  p[5,k]  +   p[6,k]  );

dynamics { i in 1..6,k in 0..N-1 } :
  p[i,k+1] =      (gamma[k] * sum { c in 1..6}  Q[i,c]  * p[c,k]) +
(alpha[k] * sum { c in 1..6} Q[i,c]*sum { j in 1..6}  Ma[c,j]*p[j,k]);

full_maintenance_active:          sum { j in 0..N-1} alpha [j] <= n_f;

one_active { k in 0..N-1}  :                    gamma[k] + alpha[k]  = 1;

subject to  initial_values_constraints {i in 1 .. 6}:
                                  p[i,0] = initial_values[i];
```

Examples – Example 1 (MODEL)

- **In MATLAB**

```
ampl.defineParam("N",1,"integer")
ampl.defineVar("alpha gamma ","0..N","binary")
ampl.defineVar("p",[6,"0..N"],">=0")
ampl.defineParam("n_f")
ampl.defineParam("Q Ma",[6,6])
ampl.defineParam("initial_values ",6)

cost_function = ['  1/(N)*sum { k in 0..N-1} ( ' ...
                  ' p[3,k] + p[4,k+1] + p[5,k] + x[6,k+1] ' ];
ampl.defineObj("min","cost_function",cost_function);

dynamics = ['  p[i,k+1] = ' ...
 '  (gamma[k] * sum { c in 1..6} Q[i,c] * p[c,k]) + ' ...
 '  (alpha[k] * sum { c in 1..6} Q[i,c]*sum { j in 1..6} Ma[c,j]*p[j,k]) ' ];
ampl.defineCons("dynamics",dynamics,"i in 1..6, k in 0..N-1" );

full_maintenance_active= 'sum { j in 0..N-1} alpha[j] <= n_f';
ampl.defineCons("full_maintenance_active",full_maintenance_active)

one_active =  ' gamma[j]  + alpha[j] = 1' ;
ampl.defineCons("one_active",one_active,"j in 0..N-1")
ampl.defineCons("initial_values_constraints", 'x[i,0] = initial_values[i]' ,"i in 1..6")
% Note: ampl.defineParam, defineVar, defineCons and
% ampl.defineCons are customized codes
```

```
param N , integer                         ; # horizon length
var alpha { k in 0..N-1 } , binary        ; # full maintenance mode
var gamma { k in 0..N-1 } , binary        ; # dynamics mode
var p {i in 1..6 , k in  0..N  } >= 0     ; # states vector
param n_f                                 ; # num. of full maintenance
param initial_values {j in 1 ..6}         ; # initial values
param Q {i in 1..6, j in 1..6}            ; # system matrix
param Ma {i in 1..6, j in 1..6}           ; # full maintenance matrix

 minimize cost_function:
1/(N)*sum { k in 0..N-1} ( p[3,k]  +   p[4,k]  +  p[5,k]  +   p[6,k]  );

dynamics { i in 1..6,k in 0..N-1 } :
  p[i,k+1] =      (gamma[k] * sum { c in 1..6}  Q[i,c]  * p[c,k]) +
(alpha[k] * sum { c in 1..6} Q[i,c]*sum { j in 1..6}  Ma[c,j]*p[j,k]);

full_maintenance_active:          sum { j in 0..N-1} alpha [j] <= n_f;

one_active { k in 0..N-1}  :                  gamma[k] + alpha[k]  = 1;

subject to  initial_values_constraints {i in 1 .. 6}:
                                   p[i,0] = initial_values[i];
```

Examples – Example 1 (DATA)

- **Optimal control: mixed integer programming**

$$\min_{\alpha(k),\gamma(k),p(k)} \frac{1}{N}\sum_{k=0}^{N-1}(p_3(k)+p_4(k)+p_5(k)+p_6(k))$$

$$\text{s.t.}\quad p(k+1)=\gamma(k)(Qp(k))+\alpha(k)(Q(M_A p(k)),$$

$$\sum_{k=0}^{N-1}\alpha(k)\le n_f, \qquad \alpha(k)\in\{0,1\}$$

$$\alpha(k)+\gamma(k)=1, \qquad k=0,1,\cdots,N-1$$

$$p(0)=p_0,$$

```
param n_f   := 1 ;
param initial_values      :=  1  1
                              2  0
                              3  0
                              4  0
                              5  0
                              6  0 ;
param N := 131400;         # 15 years
param Q :   1      2        3        4      5       6 :=
1           1    0.005    0.001      0      0       0
2           0      1        0        0      0       0
3           0      0        1        0      0       0
4           0      0        0        1      0       0
5           0      0        0        0      1       0
6     0         0        0       0       0       1;

param Ma  :  1  2  3  4  5  6   :=
          1   1  0  0  0  0  0
          2   0  1  0  0  0  0
          3   0  0  1  1  1  0
          4   0  0  0  0  0  0
          5   0  0  0  0  0  0
          6   0  0  0  0  0  1;
```

Examples – Example 1 (DATA)

- **In MATLAB**

```
n_f= 1 ;
initial_values = [ 1 0 0 0 0 0];
N = 131400;

Q = [1   0.005   0.001   0   0   0
     0   1       0       0   0   0
     0   1       0       0   0   0
     0   0       0       1   0   0
     0   0       0       0   1   0
     0   0       0       0   0   1];

Ma = [ 1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 1 1 0;0 0 0 0 0 0;
       0 0 0 0 0 0 ; 0 0 0 0 0 1];

ampl.assignParam("N n_f", [N n_f])
ampl.assignParam("initial_values",initial_values)
ampl.assignParam("Ma",Ma)
ampl.assignParam("Q",Q)

% Note: ampl.defineParam, defineVar, defineCons and
% ampl.defineCons are customized codes
```

```
param n_f   := 1  ;
param initial_values      := 1  1
                             2  0
                             3  0
                             4  0
                             5  0
                             6  0 ;
param N := 131400;        # 15 years
param Q :   1     2       3       4     5     6 :=
1           1     0.005   0.001   0     0     0
2           0     1       0       0     0     0
3           0     0       1       0     0     0
4           0     0       0       1     0     0
5           0     0       0       0     1     0
6   0       0     0       0       0     1;

param Ma  :  1  2  3  4  5  6   :=
          1  1  0  0  0  0  0
          2  0  1  0  0  0  0
          3  0  0  1  1  1  0
          4  0  0  0  0  0  0
          5  0  0  0  0  0  0
          6  0  0  0  0  0  1;
```

Examples – Example 1 (SOLVE)

- **In MATLAB**

```
%% solve the optimization problem
ampl.solve


%% Harvest the results
% check the value of N
ampl.getParamValue("N")
ampl.getParamValue("initial_values")
ampl.getParamValue("Ma")
[gamma, index,rowdata] = ampl.getVarValue("gamma");
alpha = ampl.getVarValue("alpha");
p = ampl.getVarValue("p");


% objective function
[value, astatus,result,exitcode,message,expandCost] = ampl.getObj('cost_function')
```

Examples – MPC

- **Go to folder: amplPathDir\amplapi\matlab\AMPL-Codes-main\Examples\MPC**
- Check and run the matlab codes:

Example_9_Tracking_Mass_Spring_Damper_System.m

Example_9_Regulation.m

Helpful functions

Customized codes for AMPL in MATLAB:

- expand(input): results in a string array of a matrix, vector, or scalar elements
- Examples:

Line command	expand('Q[1..3,1..2]')	expand('y[0..2,k]')	expand('s[1..1]')	expand('v.T[0..%u]',1)
Data Type Output	3×2 string array	3×1 string array	String	1×2 string array
Output	"Q[1,1]" "Q[1,2]" "Q[2,1]" "Q[2,2]" "Q[3,1]" "Q[3,2]"	"y[0,k]" "y[1,k]" "y[2,k]"	"s"	"v[0]" "v[1]"

Helpful functions

Customized codes for AMPL in MATLAB:

- multiply(A,B): results in a string array of a product of A and B
- Examples:

Line command	Data Type Output	Output
multiply('x.T[1..2,k] ','Q[1..2,1..2] ')	1×2 string array	"(x[1,k]*Q[1,1]+ x[2,k]*Q[2,1])" "(x[1,k]*Q[1,2]+ x[2,k]*Q[2,2])"
multiply('x.T[1..2,k] ','Q[1..2,1..2] ', 'x[1..2,k] ',)	string	"((x[1,k]*Q[1,1]+ x[2,k]*Q[2,1])*x[1,k]+ (x[1,k]*Q[1,2]+ x[2,k]*Q[2,2])*x[2,k])"

Installation - 1

1. - Install AMPL program: https://ampl.com/try-ampl/download-a-free-demo/

    choose the download link based on the operating system

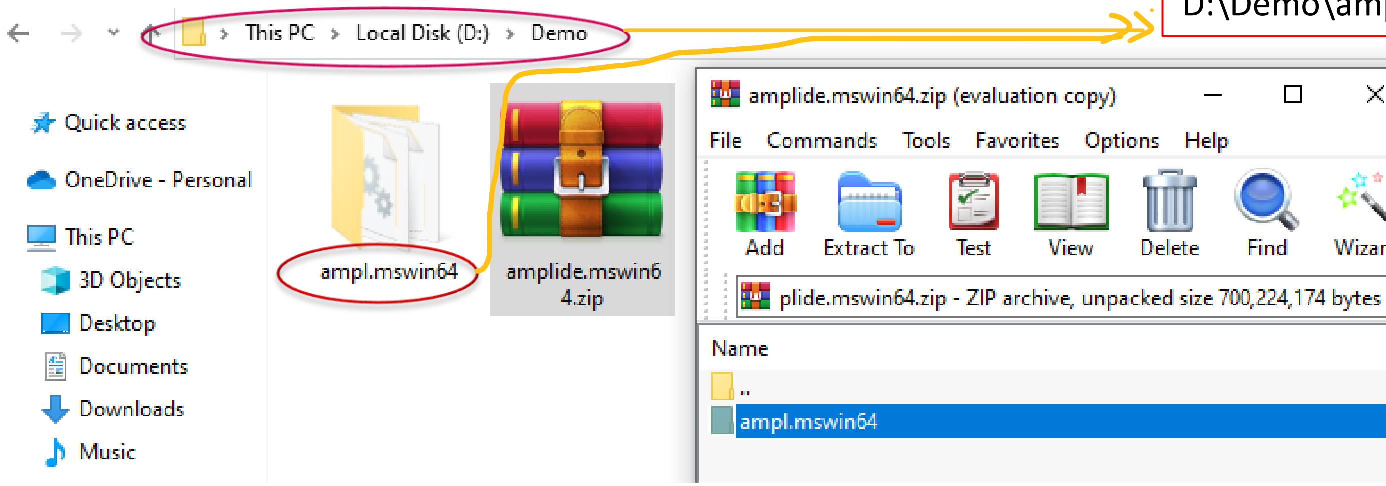
    for 64 bit – Windows and 32 bit Windows the download links are respectively:

    https://ampl.com/demo/ampl.mswin64.zip

    https://ampl.com/demo/ampl.mswin32.zip

    - Unzip (extract the folder: ampl.mswin64) inside the zip file.

Note that amplPathDir is:
D:\Demo\ampl.mswin64



1.

Installation - 2

2. - Install Ampl API using the link: https://ampl.com/products/api/

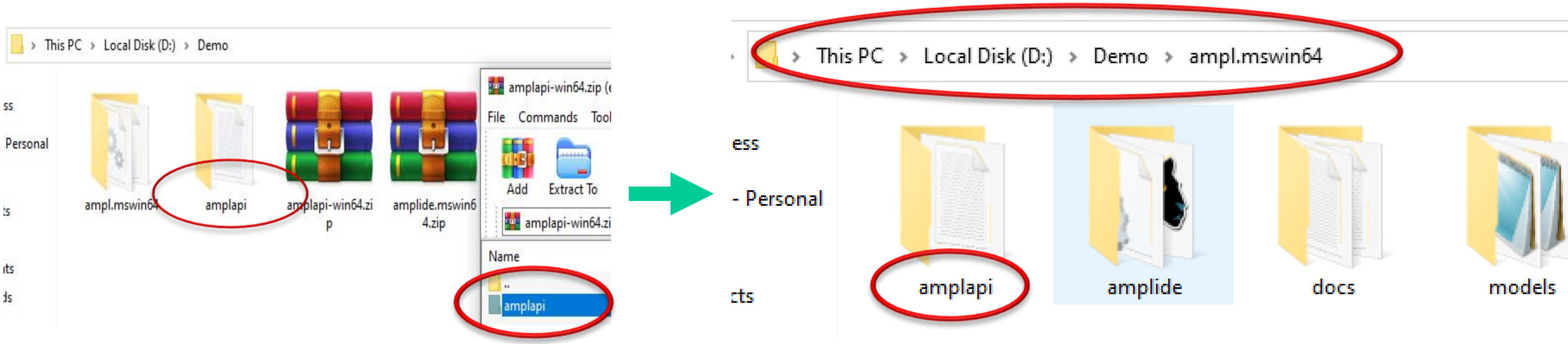
   choose the download link based on the operating system and programming language

   download links for C++, C#, Java, and MATLAB APIs for Windows 32 bit and 64 bit are respectively:

   https://ampl.com/dl/API/latest/amplapi-win64.zip

   https://ampl.com/dl/API/latest/amplapi-win32.zip

   - Unzip (extract the folder: amplapli) inside the zip file and move to ampl.mswin64 folder.

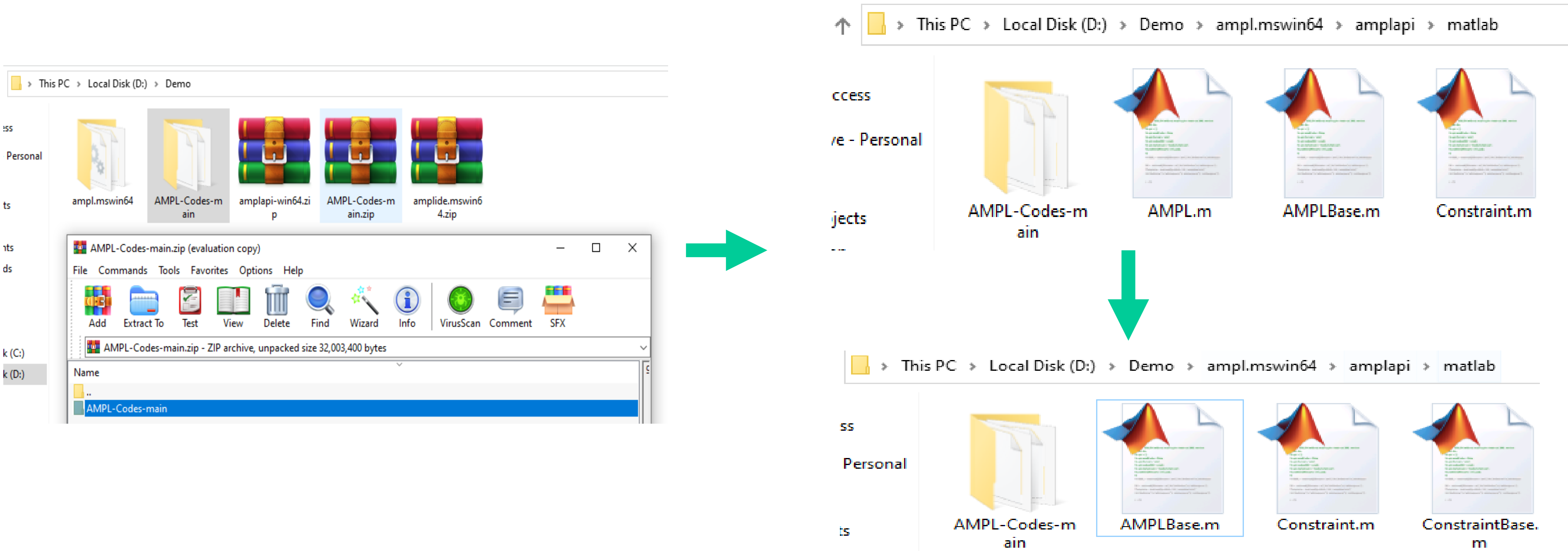


Installation - 3

3. - install the repository: AMPL-codes on the link:

   https://github.com/Anas-Abdelkarim/AMPL-Codes

   - unzip (extract the file: AMPL-Codes-main) and move to: ampl.mswin64\amplapi
   - delete AMPL.m form the folder: ampl.mswin64\amplapi\matlab



Installation - note

Note: the students can request 30-day trail over the link: https://ampl.com/products/ampl/ampl-for-students/#Trial

Referncecces

1. https://www.youtube.com/watch?v=WxGBdCPf5vM&ab_channel=YongWang
2. https://www.youtube.com/watch?v=hrqsflMu4z8&ab_channel=AMPLOptimization
3. https://neos-server.org/neos/
4. Abdelkarim, A. and Zhang, P., 2020, September. Optimal Scheduling of Preventive Maintenance for Safety Instrumented Systems Based on Mixed-Integer Programming. In *International Symposium on Model-Based Safety and Assessment* (pp. 83-96). Springer, Cham.