

New AMPL Interfaces for Enhanced Optimization Model Development and Deployment



Robert Fourer

AMPL Optimization Inc.

www.ampl.com — +1 773-336-AMPL

INFORMS Annual Meeting

Minneapolis — 6-9 October 2013

Session MC29, *Software Demonstrations*

New Interface Developments in the AMPL Modeling Language & System

We introduce the new AMPL IDE, which provides an integrated command interpreter and file editor for more convenient optimization model development. Additionally we present details of the first AMPL APIs, which facilitate model deployment by letting your applications invoke many AMPL functions directly, using calls written in popular programming languages.

Outline

Developing models

- ❖ More natural formulations
 - * Logical conditions
 - * Quadratic constraints
- ❖ **AMPL IDE** (Integrated Development Environment)
 - * Unified editor & command processor
 - * Built on the Eclipse platform
- ❖ SolverStudio
 - * Access to AMPL within Excel

Deploying models

- ❖ **AMPL API** (Application Programming Interfaces)
 - * Programming languages: C++, Java, .NET, Python
 - * Analytics languages: MATLAB, R

Logical Conditions: Multi-Commodity

Minimum-shipment constraints

- ❖ From each origin to each destination, *either* ship nothing *or* ship at least minload units

Conventional linear mixed-integer formulation

```
var Trans {ORIG,DEST,PROD} >= 0;
var Use {ORIG, DEST} binary;
.....
subject to Multi {i in ORIG, j in DEST}:
    sum {p in PROD} Trans[i,j,p] <= limit[i,j] * Use[i,j];
subject to Min_Ship {i in ORIG, j in DEST}:
    sum {p in PROD} Trans[i,j,p] >= minload * Use[i,j];
```

Multi-Commodity

Zero-One Alternatives

Mixed-integer formulation using implications

```
subject to Multi_Min_Ship {i in ORIG, j in DEST}:  
    Use[i,j] = 1 ==>  
        minload <= sum {p in PROD} Trans[i,j,p] <= limit[i,j]  
        else sum {p in PROD} Trans[i,j,p] = 0;
```

Solved directly by CPLEX

```
ampl: model multmipImpl.mod;  
ampl: data multmipG.dat;  
ampl: option solver cplex;  
ampl: solve;  
CPLEX 12.5.1.0: optimal integer solution; objective 235625  
176 MIP simplex iterations  
0 branch-and-bound nodes
```

Multi-Commodity

Non-Zero-One Alternatives

Disjunctive constraint

```
subject to Multi_Min_Ship {i in ORIG, j in DEST}:  
    sum {p in PROD} Trans[i,j,p] = 0 or  
    minload <= sum {p in PROD} Trans[i,j,p] <= limit[i,j];
```

Solved by CPLEX?

```
AMPL: model multmipDisj.mod;  
AMPL: data multmipG.dat;  
AMPL: solve;  
CPLEX 12.5.1.0: logical constraint not indicator constraint.  
AMPL: option solver ilogcp;  
AMPL: option ilogcp_options 'optimizer cplex';  
AMPL: solve;  
ilogcp 12.5.0: optimal solution  
0 nodes, 175 iterations, objective 235625
```

Logical Conditions: Sequencing

Optimally order a collection of jobs

- ❖ Given a set of adjacency preferences, maximize the number that are satisfied

Decision variables

- ❖ For each preference “i1 adjacent to i2”:
Sat [i1, i2] = 1 iff this is satisfied in the sequence
- ❖ Pos [i] is the position of job i in the sequence

. . . fewer variables, larger domains

Arrangement

“CP-Style” Alternative

All-different constraint

```
param nJobs integer > 0;
set PREFS within {i1 in 1..nJobs, i2 in 1..nJobs: i1 <> i2};

var Sat {PREFS} binary;
var Pos {1..nJobs} integer >= 1, <= nJobs;

maximize NumSat: sum {(i1,i2) in PREFS} Sat[i1,i2];

subject to OneJobPerPosition:
    alldiff {i in 1..nJobs} Pos[i];

subject to SatDefn {(i1,i2) in PREFS}:
    Sat[i1,i2] = 1 <==> -1 <= Pos[i1]-Pos[i2] <= 1;

subject to SymmBreaking:
    Pos[1] < Pos[2];
```


Arrangement

“CP-Style” Alternative (*cont’d*)

11 jobs, 20 preferences

```
ampl: model jobseq.mod;
ampl: data jobseq11.dat;
ampl: option solver ilogcp;
ampl: solve;
ilogcp 12.5.1: optimal solution
8654397 choice points, 4299824 fails, objective 12
ampl: option solver gecode;
ampl: solve;
gecode 4.2.0: optimal solution
3282579 nodes, 1641270 fails, objective 12
ampl:
```

More Natural Modeling

Logical Conditions: Current

Expressions recognized by AMPL

- ❖ Disjunctions (or), implications (\implies)
- ❖ Counting expressions (count),
Counting constraints (atleast, atmost)
- ❖ Aggregate constraints (alldiff, numberof)

Solvers supported

- ❖ IBM CPLEX mixed-integer programming solver
 - * Applied directly
 - * Applied after automatic conversion to MIP
- ❖ Constraint programming solvers
 - * IBM ILOG CP
 - * Gecode
 - * JaCoP

More Natural Modeling

Logical Conditions: Planned

What the AMPL-solver interface will do

- ❖ Recognize transformable “not linear” expressions
 - * Logical operators
 - * Piecewise operators: abs, min, max
- ❖ Automatically transform to LPs or MILPs

New forms to be recognized

- ❖ Object-valued variables
 - * `var JobForSlot {1..nSl+1} in JOBS;`
- ❖ Variables in subscripts
 - * `minimize TotalCost:`
`sum {k in 1..nSl} setupCost [JobForSlot [k], JobForSlot [k+1]] + ...`
- ❖ Set membership constraints
 - * `subject to SeqRestrictions {k in 1..nSl}`
`(JobForSlot [k], JobForSlot [k+1]) in ALLOWED;`

More Natural Modeling

Quadratic Constraints: Traffic Flow

Given a traffic network

N Set of nodes representing intersections

e Entrance to network

f Exit from network

$A \subseteq N \cup \{e\} \times N \cup \{f\}$

Set of arcs representing road links

with associated data

b_{ij} Base travel time for each road link $(i, j) \in A$

s_{ij} Traffic sensitivity for each road link $(i, j) \in A$

c_{ij} Capacity for each road link $(i, j) \in A$

T Desired throughput from e to f

Traffic Network

Formulation

Determine

x_{ij} Traffic flow through road link $(i, j) \in A$

t_{ij} Actual travel time on road link $(i, j) \in A$

to minimize

$$\sum_{(i,j) \in A} t_{ij} x_{ij} / T$$

Average travel time from e to f

Traffic Network

Formulation (*cont'd*)

Subject to

$$t_{ij} = b_{ij} + \frac{s_{ij}x_{ij}}{1 - x_{ij}/c_{ij}} \quad \text{for all } (i,j) \in A$$

Travel times increase as flow approaches capacity

$$\sum_{(i,j) \in A} x_{ij} = \sum_{(j,i) \in A} x_{ji} \quad \text{for all } i \in N$$

Flow out equals flow in at any intersection

$$\sum_{(e,j) \in A} x_{ej} = T$$

Flow into the entrance equals the specified throughput

AMPL Formulation

Symbolic data

```
set INTERS;           # intersections (network nodes)

param EN symbolic;   # entrance
param EX symbolic;   # exit

    check {EN,EX} not within INTERS;

set ROADS within {INTERs union {EN}} cross {INTERs union {EX}};

                                # road links (network arcs)

param base {ROADS} > 0; # base travel times
param sens {ROADS} > 0; # traffic sensitivities
param cap {ROADS} > 0;  # capacities
param through > 0;     # throughput
```

AMPL Formulation (*cont'd*)

Symbolic model

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
    (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
    Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
    sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
    sum{(EN,j) in ROADS} Flow[EN,j] = through;
```


Traffic Network

AMPL Data

Explicit data independent of symbolic model

```
set INTERS := b c ;  
  
param EN := a ;  
param EX := d ;  
  
param: ROADS: base cap sens :=  
    a b    4   10   .1  
    a c    1   12   .7  
    c b    2   20   .9  
    b d    1   15   .5  
    c d    6   10   .1 ;  
  
param through := 20 ;
```

Traffic Network

AMPL Solution

Model + data = problem to solve, using KNITRO

```
ampl: model traffic.mod;  
ampl: data traffic.dat;  
ampl: option solver knitro;  
ampl: solve;
```

KNITRO 7.0.0: Locally optimal solution.

```
objective 61.04695019; feasibility error 3.55e-14  
12 iterations; 25 function evaluations
```

```
ampl: display Flow, Time;
```

:	Flow	Time	:=
a b	9.55146	25.2948	
a c	10.4485	57.5709	
b d	11.0044	21.6558	
c b	1.45291	3.41006	
c d	8.99562	14.9564	
;			

Traffic Network

AMPL Solution (*cont'd*)

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
ampl: solve;
```

```
KNITRO 7.0.0: Locally optimal solution.
```

```
objective 76.26375; integrality gap 0
```

```
3 nodes; 5 subproblem solves
```

```
ampl: display Flow, Time;
```

```
:      Flow      Time      :=  
a b      9      13  
a c     11     93.4  
b d     11     21.625  
c b      2       4  
c d      9      15  
;
```

Traffic Network

AMPL Solution (*cont'd*)

Model + data = problem to solve, using Gurobi?

```
ampl: model traffic.mod;  
ampl: data traffic.dat;  
ampl: option solver gurobi;  
ampl: solve;
```

Gurobi 5.5.0:

Gurobi can't handle nonquadratic nonlinear constraints.

Traffic Network

AMPL Solution (*cont'd*)

Look at the model again . . .

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Time {ROADS} >= 0;

minimize Avg_Time:
    (sum {(i,j) in ROADS} Time[i,j] * Flow[i,j]) / through;

subject to Travel_Time {(i,j) in ROADS}:
    Time[i,j] = base[i,j] + (sens[i,j]*Flow[i,j]) / (1-Flow[i,j]/cap[i,j]);

subject to Balance_Node {i in INTERS}:
    sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
    sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

AMPL Solution (*cont'd*)

Quadratically constrained reformulation

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= (1 - Flow[i,j]/cap[i,j]) * Delay[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```

Traffic Network

AMPL Solution (*cont'd*)

Model + data = problem to solve, using Gurobi?

```
ampl: model trafficQUAD.mod;  
ampl: data traffic.dat;  
  
ampl: option solver gurobi;  
ampl: solve;
```

Gurobi 5.5.0:

quadratic constraint is not positive definite

AMPL Solution (*cont'd*)

Simple conic quadratic reformulation

```
var Flow {(i,j) in ROADS} >= 0, <= .9999 * cap[i,j];
var Delay {ROADS} >= 0;
var Slack {ROADS} >= 0;

minimize Avg_Time:
  sum {(i,j) in ROADS} (base[i,j]*Flow[i,j] + Delay[i,j]) / through;

subject to Delay_Def {(i,j) in ROADS}:
  sens[i,j] * Flow[i,j]^2 <= Slack[i,j] * Delay[i,j];

subject to Slack_Def {(i,j) in ROADS}:
  Slack[i,j] = 1 - Flow[i,j]/cap[i,j];

subject to Balance_Node {i in INTERS}:
  sum{(i,j) in ROADS} Flow[i,j] = sum{(j,i) in ROADS} Flow[j,i];

subject to Balance_Enter:
  sum{(EN,j) in ROADS} Flow[EN,j] = through;
```


Traffic Network

AMPL Solution (*cont'd*)

Model + data = problem to solve, using Gurobi!

```
ampl: model trafficSOC.mod;  
ampl: data traffic.dat;  
ampl: option solver gurobi;  
ampl: solve;
```

```
Gurobi 5.5.0: optimal solution; objective 61.04696953  
47 barrier iterations
```

```
ampl: display Flow;
```

```
Flow :=  
a b    9.55146  
a c    10.4485  
b d    11.0031  
c b     1.45167  
c d     8.99687  
;
```

Traffic Network

AMPL Solution (*cont'd*)

Same with integer-valued variables

```
var Flow {(i,j) in ROADS} integer >= 0, <= .9999 * cap[i,j];
```

```
AMPL: solve;
```

```
Gurobi 5.5.0: optimal solution; objective 76.26374998
```

```
32 simplex iterations
```

```
AMPL: display Flow;
```

```
Flow :=
```

```
a b    9
```

```
a c   11
```

```
b d   11
```

```
c b    2
```

```
c d    9
```

```
;
```

More Natural Modeling

Convex Quadratics: Current

Problem types

- ❖ Elliptic: quadratic programs (QPs)
- ❖ Conic: second-order cone programs (SOCPs)

What the AMPL-solver interface does

- ❖ Recognize quadratic objectives & constraints
- ❖ Multiply out products of linear terms
- ❖ Send linear & quadratic coefficient lists to solver

What the solver does

- ❖ Detect elliptic forms numerically
- ❖ Detect conic forms by structural analysis

More Natural Modeling

Convex Quadratics: Planned

What the AMPL-solver interface will do

- ❖ Recognize nonquadratic SOCP-equivalent problems
- ❖ Automatically transform to SOCPs recognizable by solvers

Forms to be recognized

- ❖ Sum of norms
- ❖ Sum of squares divided by linear
- ❖ Generalized geometric means
- ❖ Generalized p -norms
- ❖ log-Chebyshev objectives

*... combinations by sum, max, positive multiple
(where possible)*

AMPL IDE

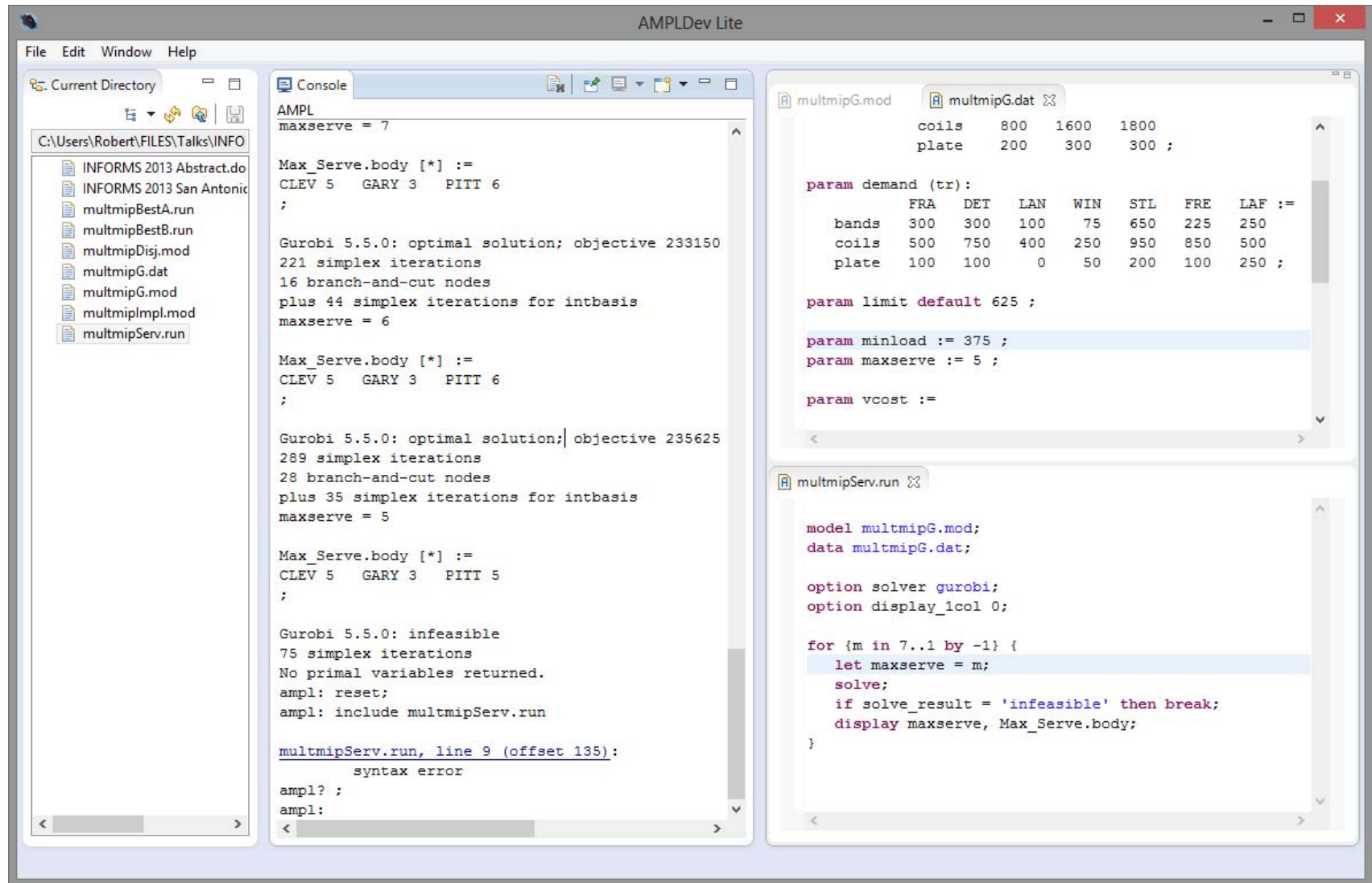
Integrated Development Environment

- ❖ Unified editor & command processor
- ❖ Included in the AMPL distribution
 - * Easy upgrade path
 - * Command-line, batch versions remain available
- ❖ Built on Eclipse
 - * Runs under Windows, Linux, MacOS

Initial release

- ❖ Simplified for easy transition
- ❖ Works with existing installations

Sample Screenshot



AMPL IDE

Version 1.0

Rollout dates

- ❖ Beta test
 - * Completed last summer
- ❖ Release
 - * October/November target
 - * Available with all AMPL distributions

Development details

- ❖ Partnership with OptiRisk Systems
- ❖ Long-term development & maintenance by AMPL
- ❖ “AMPLDEV” advanced IDE to be marketed by OptiRisk
 - * Offers full stochastic programming support

AMPL IDE

Version *x.y*

More help

- ❖ Option selection dialogs
 - * AMPL options
 - * Solver options
- ❖ AMPL language quick reference

NEOS Server access

Enhanced displays

- ❖ Parameter view windows
- ❖ Graphs

Suggestions from users . . .

SolverStudio

Solver interface from Excel

- ❖ MD08 Modeling Systems II, 16:30-18:00, CC 200H
- ❖ **Andrew Mason,**
**SolverStudio: A Free Integrated Excel Environment
for Optimization using Modelling Languages**

SolverStudio

The screenshot shows the SolverStudio for AMPL interface within Microsoft Excel. The main window displays a spreadsheet with the following data:


Ingredients	Costs	Contributes					Amount
		Protein	Fat	Fibre	Salt	OneCan	
Chicken	0.013	0.1	0.08	0.001	0.002	1	0
Beef	0.008	0.2	0.1	0.005	0.005	1	10
Mutton	0.01	0.15	0.11	0.003	0.007	1	0
Rice	0.002	0	0.01	0.1	0.002	1	0
Wheat bran	0.005	0.04	0.01	0.15	0.008	1	0
Gel	0.001	0	0	0	0	1	90

The SolverStudio window shows the following AMPL model:

```
set INGREDIENTS;
set REQUIREMENTS;
param Cost {INGREDIENTS};
param Contributes {REQUIREMENTS, INGREDIENTS};
param Lower {REQUIREMENTS};
param Upper {r in REQUIREMENTS} >= Lower[r];
var Amount {INGREDIENTS} >= 0;
default InFINITY;
minimize TotalCost:
  sum {i in INGREDIENTS} Cost[i] * Amount[i];
subject to MeetRequirements {r in REQUIREMENTS}:
  Lower[r]
  <= sum {i in INGREDIENTS} Contributes[r,i] * Amount[i];
  <= Upper[r];
data SheetData.dat: # Get data from the spreadsheet;
option solver cplexamp;
solve;
display Amount > Sheet; # Write the solution to the spreadsheet;
```

The Model Output window shows the following results:

```
AMPL Version 20080102 (x86_win32)
CPLEX 11.0.0: optimal solution; objective 0.17
2 dual simplex iterations (0 in phase I)
## AMPL run completed.
```

- Build & solve AMPL models using Excel
- Solve AMPL models in the cloud using 
- Free download: <http://solverstudio.org>

SolverStudio

SolverStudio for AMPL.xlsx - Microsoft Excel

Home Insert Page Layout Formulas Data Review View Developer Team

Get External Data Refresh All Sort Filter Reapply Advanced Text to Columns Remove Duplicates Outline Data Analysis Solver Solve Model Show Model Show/Hide Data Edit Data Show Data in Color About SolverStudio

INGREDIENTS Chicken

Ingredients	Costs	Prote
Chicken	0.013	0.1
Beef	0.008	0.2
Mutton	0.01	0.1
Rice	0.002	0
Wheat bran	0.005	0.0
Gel	0.001	0

SolverStudio - Data Items Editor

Name:	Cell Range:	Index Range(s):	Value if Missing:
<Add New Data Item >			
Amount	J4:J9	INGREDIENTS	<Report an error >
Contributes	D4:H9	REQUIREMENTS, INGREDIENTS	<Report an error >
Cost	C4:C9	INGREDIENTS	<Report an error >
INGREDIENTS	B4:B9		
Lower	D13:H13	REQUIREMENTS	<Report an error >
REQUIREMENTS	D3:H3		
Upper	D12:H12	REQUIREMENTS	<Report an error >

Name: Cell Range: Index Range(s): Unknown Indices return

INGREDIENTS B4:B9 an error

Delete Data Item Update Data Item Cancel

This dialog allows you to define cell ranges ('data items') on your spreadsheet that you want to use in your optimisation model. You can define sets of items, indexed lists (a cell range containing one entry for each value in an associated index range) and indexed tables (which need 2 index ranges).

Count: 6 100%

- Easily define params & sets on spreadsheet
- Automatic data exchange with model
- Free download from <http://solverstudio.org>

SolverStudio

AMPL Modeling in Excel

SolverStudio is an integrated Excel add-in that makes it easy to develop and deliver AMPL optimization models using the familiar Excel environment.

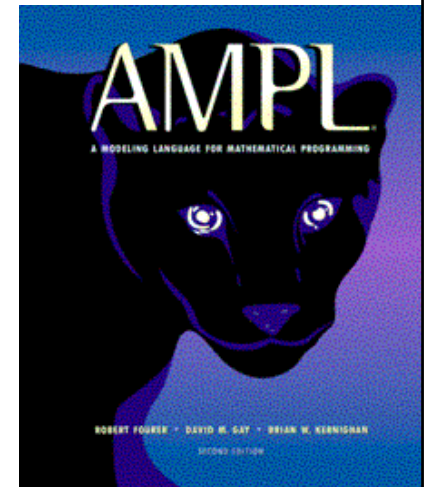
SolverStudio adds a text editor to Excel that allows an optimization model to be created using AMPL and then embedded and saved within a spreadsheet. SolverStudio also provides an integrated data editor that allows model data (AMPL parameters and sets) to be stored and edited on the spreadsheet.

SolverStudio's *Solve* button runs the AMPL model while seamlessly managing data transfers with the spreadsheet. AMPL models can be solved locally, or in the cloud using NEOS.

SolverStudio is being developed and supported by Andrew Mason at the Department of Engineering Science, University of Auckland, New Zealand.

MD08 Modeling Systems II, 16:30-18:00, CC 200H

<http://solverstudio.org>



**Solver
Studio
for Excel**

AMPL API

Application Programming Interface

- ❖ Programming languages: C++, Java, .NET, Python
- ❖ Analytics languages: MATLAB, R

Facilitates use of AMPL for

- ❖ Complex algorithmic schemes
- ❖ Embedding in other applications
- ❖ Deployment of models

Deployment Alternatives

Stand-alone: Give (temporary) control to AMPL

- ❖ Write needed files
- ❖ Invoke AMPL to run some scripts
- ❖ Read the files that AMPL leaves on exit

API: Interact with AMPL

- ❖ Execute AMPL statements individually
- ❖ Read model, data, script files when convenient
- ❖ Exchange data tables directly with AMPL
 - * populate sets & parameters
 - * invoke any available solver
 - * extract values of variables & result expressions

. . . all embedded within your program's logic

Example: Java

Efficient frontier: Initialize, read files

```
AMPL ampl = createAMPL();
int steps = 30;
try
{
    ampl.interpretFile(Utills.getResFileName("qpmv.mod", "qpmv", true), false);
    ampl.interpretFile(Utills.getResFileName("qpmv.dat", "qpmv", true), true);
}
catch (IOException e)
{
    e.printStackTrace();
    return -1;
}

VariableMap portfolioReturn = ampl.getVariable('portret');
ParameterMap averageReturn = ampl.getParameter('averret');
ParameterMap targetReturn = ampl.getParameter('targetret');
ObjectiveMap deviation = ampl.getObjective('cst');
```

Example: Java (*cont'd*)

Efficient frontier: Solve, set up for loop

```
AMPL.interpret("option solver cplex;");
AMPL.interpret("let stockopall:={ }; let stockrun:=stockall;");
AMPL.interpret("option relax_integrality 1;");

AMPL.solve()

double minret = portfolioReturn.get().value();
double maxret = findMax(averageReturn.getDouble());
double stepsize = (maxret-minret)/steps;

double[] returns = new double[steps];
double[] deviations = new double[steps];
```


Example: Java (*cont'd*)

Efficient frontier: Loop over solves

```
for(int i=0; i<steps; i++)
{
    System.out.println(String.format
        ("Solving for return = %f", maxret - (i-1)*stepsize));
    targetReturn.let(maxret - (i-1)*stepsize);
    ampl.interpret("let stockopall:={}; let stockrun:=stockall;");
    ampl.interpret("options relax_integrality 1;");
    ampl.solve();
    ampl.interpret("let stockrun2:={i in stockrun:weights[i]>0};");
    ampl.interpret(" let stockrun:=stockrun2;");
    ampl.interpret(" let stockopall:={i in stockrun:weights[i]>0.5};");
    ampl.interpret("options relax_integrality 0;");
    ampl.solve();
    returns[i] = maxret - (i-1)*stepsize;
    deviations[i] = deviation.get().value();
}
```

Example: MATLAB

Efficient frontier: Initialize, read files

```
AMPL = initAMPL;
steps = 30;
AMPL.interpretFile('qpmv.mod', false)
AMPL.interpretFile('qpmv.dat', true)
portfolioReturn = AMPL.getVariable('portret');
averageReturn = AMPL.getParameter('averret');
targetReturn = AMPL.getParameter('targetret');
deviation = AMPL.getObjective('cst');
```

Example: MATLAB (*cont'd*)

Efficient frontier: Solve, set up for loop

```
AMPL.interpret('option solver afortmp;');
AMPL.interpret('let stockopall:={ }; let stockrun:=stockall;');
AMPL.interpret('option relax_integrality 1;');

AMPL.solve()

minret = portfolioReturn.getDouble();
maxret = max(averageReturn.getDouble());
stepsize = (maxret-minret)/steps;

returns = zeros(steps, 1);
deviations = zeros(steps, 1);
```

Example: MATLAB (*cont'd*)

Efficient frontier: Loop over solves

```
for i=1:steps
    fprintf('Solving for return = %f\n', maxret - (i-1)*stepsize)
    targetReturn.let(maxret - (i-1)*stepsize);
    ampl.interpret('let stockopall:={}; let stockrun:=stockall;');
    ampl.interpret('option relax_integrality 1;');
    ampl.solve();

    ampl.interpret('let stockrun2:={i in stockrun:weights[i]>0};');
    ampl.interpret('let stockrun:=stockrun2;');
    ampl.interpret('let stockopall:={i in stockrun:weights[i]>0.5};');
    ampl.interpret('option relax_integrality 0;');
    ampl.solve();

    returns(i) = maxret - (i-1)*stepsize;
    deviations(i) = deviation.getDouble();
end
plot(returns, deviations)
```

Data Transfer

Define symbolic data in AMPL

```
set ASSET;  
param nTimes;  
set TIME = 1 .. nTimes;  
param aReturns {rtime, ASSET};
```

Assign explicit values directly from API

```
ASSET = ampl.getSet('ASSET');  
nTimes = ampl.getParameter('nTimes');  
aReturns = ampl.getParameter('aReturns');  
  
ASSET.let(anames);  
nTimes.let(22);  
aReturns.let(returns(:,2:end));
```

AMPL API

Planned Availability

Initial languages: Java, MATLAB, C#

- ❖ Beta test
 - * End of 2013
 - * *Seeking beta testers now*
- ❖ Release
 - * Spring 2014
 - * Available with all AMPL distributions

More languages to follow: C++, Python, R

Development details

- ❖ Partnership with OptiRisk Systems
- ❖ Long-term development & maintenance by AMPL

Readings (*AMPL*)

- ❖ R. Fourer, “Modeling Languages versus Matrix Generators for Linear Programming.” *ACM Transactions on Mathematical Software* **9** (1983) 143–183.
- ❖ R. Fourer, D.M. Gay, B.W. Kernighan, “A Modeling Language for Mathematical Programming.” *Management Science* **36** (1990) 519–554.
- ❖ R. Fourer, D.M. Gay, B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Belmont, CA (first edition 1993, second edition 2003).
- ❖ R. Fourer, “Algebraic Modeling Languages for Optimization.” Forthcoming in Saul I. Gass and Michael C. Fu (eds.), *Encyclopedia of Operations Research and Management Science*, Springer (2012).
- ❖ Robert Fourer, On the Evolution of Optimization Modeling Systems. M. Groetschel (ed.), *Optimization Stories*. Documenta Mathematica (2012) 377-388.

Readings (*Interfaces*)

- ❖ G. Everett, A. Philpott, K. Vatn, R. Gjessing, “Norske Skog Improves Global Profitability Using Operations Research.” *Interfaces* **40**, 1 (Jan–Feb 2010) 58–70.
- ❖ F. Caro, J. Gallien, M. Díaz, J. García, J.M. Corredoira, M. Montes, J.A. Ramos, J. Correa, “Zara Uses Operations Research to Reengineer Its Global Distribution Process.” *Interfaces* **40**, 1 (Jan–Feb 2010) 71–84.
- ❖ P. Pekgün, R.P. Menich, S. Acharya, P.G. Finch, F. Deschamps, K. Mallery, J. Van Sistine, K. Christianson, J. Fuller, “Carlson Rezidor Hotel Group Maximizes Revenue Through Improved Demand Management and Price Optimization.” *Interfaces* **43**, 1 (Jan–Feb 2013) 21–36.