

Integrating Optimization Modeling with General-Purpose Programming for Efficient and Reliable Application Deployment

Robert Fourer, Filipe Brandão

AMPL Optimization
{4er, fdabrandao}@ampl.com

Christian Valente

OptiRisk Systems
christian@optirisk-systems.com

INFORMS Annual Meeting

Houston — 22-25 October 2017 — Session TB74

INFORMS Computing Society

4 Ways to Use an Optimization Modeling Language

Command language

- ❖ Browse results & debug model interactively
- ❖ Make changes and re-run

Scripting language

- ❖ Write programs using modeling language constructs

Programming interface (API)

- ❖ Embed a modeling language and system within a general-purpose programming language

Application-building toolkit

- ❖ Use an application-building system designed specifically for the modeling language



Features

- ❖ Algebraic modeling language
- ❖ Built specially for optimization
- ❖ Designed to support many solvers

APIs

- ❖ C++, C#, Java, MATLAB, *Python*
- ❖ R coming soon

Application-building toolkit

- ❖ *QuanDec* / built on Java API

Example

Roll cutting in Python API

- ❖ Solution by pattern enumeration
- ❖ Solution by pattern generation
- ❖ Tradeoff between waste and overruns

QuanDec

- ❖ Overview

Roll Cutting Problem

Motivation

- ❖ Fill orders for rolls of various widths
 - * by cutting raw rolls of one (large) fixed width
 - * using a variety of cutting patterns

Optimization model

- ❖ Decision variables
 - * number of raw rolls to cut according to each pattern
- ❖ Objective
 - * minimize number of raw rolls used
- ❖ Constraints
 - * meet demands for each ordered width

Roll cutting

Mathematical Formulation

Given

w width of “raw” rolls

W set of (smaller) ordered widths

n number of cutting patterns considered

and

a_{ij} occurrences of width i in pattern j ,
for each $i \in W$ and $j = 1, \dots, n$

b_i orders for width i , for each $i \in W$

o limit on overruns

Roll cutting

Mathematical Formulation (*cont'd*)

Determine

X_j number of rolls to cut using pattern j ,
for each $j = 1, \dots, n$

to minimize

$$\sum_{j=1}^n X_j$$

total number of rolls cut

subject to

$$b_i \leq \sum_{j=1}^n a_{ij} X_j \leq b_i + o, \text{ for all } i \in W$$

number of rolls of width i cut
must be at least the number ordered,
and must be within the overrun limit

Roll cutting

AMPL Formulation

Symbolic model

```
param rawWidth;  
set WIDTHS;  
  
param nPatterns integer > 0;  
set PATTERNS = 1..nPatterns;  
  
param rolls {WIDTHS,PATTERNS} >= 0, default 0;  
param order {WIDTHS} >= 0;  
param overrun;  
  
var Cut {PATTERNS} integer >= 0;  
  
minimize TotalRawRolls: sum {p in PATTERNS} Cut[p];  
  
subject to FinishedRollLimits {w in WIDTHS}:  
    order[w] <= sum {p in PATTERNS} rolls[w,p] * Cut[p] <= order[w] + overrun;
```

$$b_i \leq \sum_{j=1}^n a_{ij} X_j \leq b_i + o$$

Roll Cutting

AMPL Formulation (*cont'd*)

Explicit data (independent of model)

```
param rawWidth := 64.5 ;  
param: WIDTHS: order :=  
    6.77    10  
    7.56    40  
    17.46   33  
    18.76   10 ;  
param nPatterns := 9 ;  
param rolls: 1 2 3 4 5 6 7 8 9 :=  
    6.77  0 1 1 0 3 2 0 1 4  
    7.56  1 0 2 1 1 4 6 5 2  
    17.46 0 1 0 2 1 0 1 1 1  
    18.76 3 2 2 1 1 1 0 0 0 ;  
param overrun := 6 ;
```

Roll Cutting

AMPL Command Language

Model + data = problem instance to be solved

```
ampl: model cut.mod;
ampl: data cut.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.7.1.0: optimal integer solution; objective 20
3 MIP simplex iterations
0 branch-and-bound nodes
ampl: option omit_zero_rows 1;
ampl: option display_1col 0;
ampl: display Cut;
4 13  8 5  9 2
```

Roll Cutting

Command Language (*cont'd*)

Solver choice independent of model and data

```
ampl: model cut.mod;
ampl: data cut.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 7.5.0: optimal solution; objective 20
8 simplex iterations
1 branch-and-cut nodes
ampl: option omit_zero_rows 1;
ampl: option display_1col 0;
ampl: display Cut;
4 13   5 2   7 4   9 1
```

Roll Cutting

Command Language (*cont'd*)

Results available for browsing

```
ampl: display {j in 1..nPatterns, i in WIDTHS: Cut[j] > 0} rolls[i,j];
:      4   5   7   9   :=      # patterns used
6.77   0   3   0   4
7.56   1   1   6   2
17.46  2   1   1   1
18.76  1   1   0   0

ampl: display {j in 1..nPatterns} sum {i in WIDTHS} i * rolls[i,j];
1 63.84   3 59.41   5 64.09   7 62.82   9 59.66      # pattern
2 61.75   4 61.24   6 62.54   8 62.0          # total widths

ampl: display FinishedRollLimits.lslack;
6.77  0      # overruns
7.56  1
17.46 0
18.76 5
```

Cutting *via* Pattern Enumeration

Build the pattern list, then solve

- ❖ Read general model
- ❖ Read data
 - * demands, raw width
 - * orders, overrun limit
- ❖ Compute data: all “good” patterns
 - * extract widths from demand list
 - * enumerate all non-dominated patterns
- ❖ Solve problem instance

Pattern Enumeration

Example Using the AMPL API

Hybrid approach

- ❖ Control & pattern enumeration in Python
- ❖ Model & modeling expressions in AMPL

Key to program examples

- ❖ AMPL entities
- ❖ AMPL API Python objects
- ❖ AMPL API Python methods
- ❖ Python functions etc.

AMPL Model File

Same pattern-cutting model

```
param nPatterns integer > 0;

set PATTERNS = 1..nPatterns; # patterns
set WIDTHS; # finished widths

param order {WIDTHS} >= 0; # rolls of width j ordered
param overrun; # permitted overrun on any width

param rawWidth; # width of raw rolls to be cut
param rolls {WIDTHS,PATTERNS} >= 0, default 0; # rolls of width i in pattern j

var Cut {PATTERNS} integer >= 0; # raw rolls to cut in each pattern

minimize TotalRawRolls: sum {p in PATTERNS} Cut[p];

subject to FinishedRollLimits {w in WIDTHS}:
    order[w] <= sum {p in PATTERNS} rolls[w,p] * Cut[p] <= order[w] + overrun;
```

AMPL API

Some Python Data

A float, an integer, and a dictionary

```
roll_width = 64.5
overrun = 6
Orders = {
    6.77: 10,
    7.56: 40,
    17.46: 33,
    18.76: 10
}
```

*... can also work with
lists and Pandas dataframes*

Pattern Enumeration in Python

Load & generate data, set up AMPL model

```
def cuttingEnum(dataset):
    from amplpy import AMPL

    # Read orders, roll_width, overrun
    exec(open(dataset+'.py').read(), globals())

    # Enumerate patterns
    widths = list(sorted(orders.keys(), reverse=True))
    patmat = patternEnum(roll_width, widths)

    # Set up model
    ampl = AMPL()
    ampl.option['ampl_include'] = 'models'
    ampl.read('cut.mod')
```

Pattern Enumeration in Python

Send data to AMPL

```
# Send scalar values

AMPL.param['nPatterns'] = len(patmat)
AMPL.param['overrun'] = overrun
AMPL.param['rawWidth'] = roll_width

# Send order vector

AMPL.set['WIDTHS'] = widths
AMPL.param['order'] = orders

# Send pattern matrix

AMPL.param['rolls'] = {
    (widths[i], 1+p): patmat[p][i]
    for i in range(len(widths))
    for p in range(len(patmat))
}
```

AMPL API

Pattern Enumeration in Python

Solve and get results

```
# Solve
ampl.option['solver'] = 'gurobi'
ampl.solve()

# Retrieve solution
CuttingPlan = ampl.var['Cut'].getValues()
cutvec = list(CuttingPlan.getColumn('Cut.val'))
```

Pattern Enumeration in Python

Display solution

```
# Prepare solution data
summary = {
    'Data': dataset,
    'Obj': int(AMPL.obj['TotalRawRolls'].value()),
    'Waste': AMPL.getValue(
        'sum {p in PATTERNS} Cut[p] * \
          (rawWidth - sum {w in WIDTHS} w*rolls[w,p])'
    )
}

solution = [
    (patmat[p], cutvec[p])
    for p in range(len(patmat))
    if cutvec[p] > 0
]

# Create plot of solution
cuttingPlot(roll_width, widths, summary, solution)
```

Pattern Enumeration in Python

Enumeration routine

```
def patternEnum(roll_width, widths, prefix=[]):
    from math import floor
    max_rep = int(floor(roll_width/widths[0]))
    if len(widths) == 1:
        patmat = [prefix+[max_rep]]
    else:
        patmat = []
        for n in reversed(range(max_rep+1)):
            patmat += patternEnum(roll_width-n*widths[0], widths[1:], prefix+[n])
    return patmat
```

Pattern Enumeration in Python

Plotting routine

```
def cuttingPlot(roll_width, widths, summ, solution):  
    import numpy as np  
    import matplotlib.pyplot as plt  
  
    ind = np.arange(len(solution))  
    acc = [0]*len(solution)  
  
    colorlist = ['red', 'lightblue', 'orange', 'lightgreen',  
                'brown', 'fuchsia', 'silver', 'goldenrod']
```

Pattern Enumeration in Python

Plotting routine (cont'd)

```
for p, (patt, rep) in enumerate(solution):
    for i in range(len(widths)):
        for j in range(patt[i]):
            vec = [0]*len(solution)
            vec[p] = widths[i]
            plt.barh(ind, vec, 0.6, acc,
                    color=colorlist[i%len(colorlist)], edgecolor='black')
            acc[p] += widths[i]

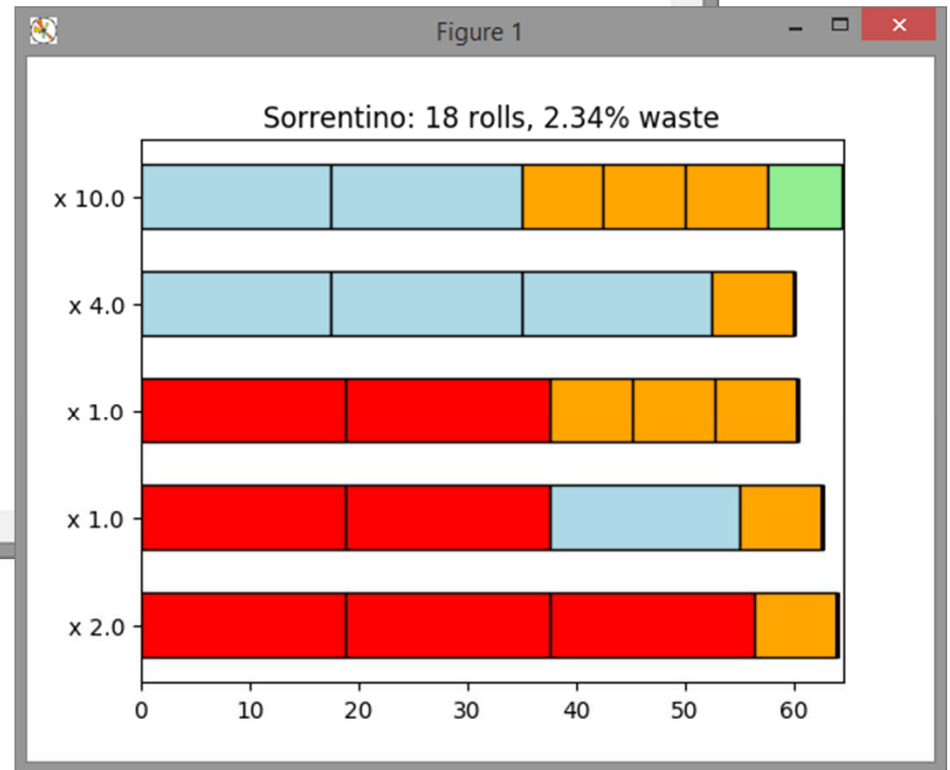
plt.title(summ['Data'] + ": " +
          str(summ['Obj']) + " rolls" + ", " +
          str(round(100*summ['Waste']/(roll_width*summ['Obj']),2)) + "% waste"
          )

plt.xlim(0, roll_width)
plt.xticks(np.arange(0, roll_width, 10))
plt.yticks(ind, tuple("x {}".format(rep) for patt, rep in solution))

plt.show()
```

Pattern Enumeration in Python

```
sw: running ipython
File Edit Help
sw: ipython
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.
In [1]: from pattern_enumeration import *
In [2]: cuttingEnum('Sorrentino')
Gurobi 7.5.0: optimal solution; objective 18
9 simplex iterations
1 branch-and-cut nodes
```



Cutting *via* Pattern Generation

Generate the pattern list by a series of solves

- ❖ Solve continuous relaxation using subset of “easy” patterns
- ❖ Add “most promising” pattern to the subset
 - * Minimize reduced cost given dual values
 - * Equivalent to a one-constraint (“knapsack”) problem
- ❖ Iterate as long as there are promising patterns
 - * Stop when minimum reduced cost is zero
- ❖ Solve IP using all patterns found

Pattern Generation

Example Using the API

Two AMPL objects

- ❖ **Master** is the cutting model with current pattern subset
- ❖ **Sub** is the one-constraint knapsack problem

Key to program examples

- ❖ AMPL entities
- ❖ AMPL API Python objects
- ❖ AMPL API Python methods
- ❖ Python functions etc.

Pattern Generation in Python

Get data, set up master problem

```
function cuttingGen(dataset)
  from amplpy import AMPL

  # Read orders, roll_width, overrun; extract widths
  exec(open(dataset+'.py').read(), globals())
  widths = list(sorted(orders.keys(), reverse=True))

  # Set up cutting (master problem) model
  Master = AMPL()
  Master.option['ampl_include'] = 'models'
  Master.read('cut.mod')

  # Define a param for sending new patterns
  Master.eval('param newPat {WIDTHS} integer >= 0;')

  # Set solve options
  Master.option['solver'] = 'gurobi'
  Master.option['relax_integrality'] = 1
```

Pattern Generation in Python

Send data to master problem

```
# Send scalar values
Master.param['nPatterns'] = len(widths)
Master.param['overrun'] = overrun
Master.param['rawWidth'] = roll_width

# Send order vector
Master.set['WIDTHS'] = widths
Master.param['order'] = orders

# Generate and send initial pattern matrix
Master.param['rolls'] = {
    (widths[i], 1+i): int(floor(roll_width/widths[i]))
    for i in range(len(widths))
}
```

Pattern Generation in Python

Set up subproblem

```
# Define knapsack subproblem
Sub = AMPL()
Sub.option['solver'] = 'gurobi'
Sub.eval('''
    set SIZES;
    param cap >= 0;
    param val {SIZES};
    var Qty {SIZES} integer >= 0;
    maximize TotVal: sum {s in SIZES} val[s] * Qty[s];
    subject to Cap: sum {s in SIZES} s * Qty[s] <= cap;
''')

# Send subproblem data
Sub.set['SIZES'] = widths
Sub.param['cap'] = roll_width
```

Pattern Generation in Python

Generate patterns and re-solve cutting problems

```
# Alternate between master and sub solves
while True:
    Master.solve()

    Sub.param['val'].setValues(Master.con['OrderLimits'].getValues())
    Sub.solve()
    if Sub.obj['TotVal'].value() <= 1.00001:
        break

    Master.param['newPat'].setValues(Sub.var['Qty'].getValues())
    Master.eval('let nPatterns := nPatterns + 1;')
    Master.eval('let {w in WIDTHS} rolls[w, nPatterns] := newPat[w];')

# Compute integer solution
Master.option['relax_integrality'] = 0
Master.solve()
```

Pattern Generation in Python

Display solution

```
# Retrieve solution

cutting_plan = Master.var['Cut'].getValues()
cutvec = list(cutting_plan.getColumn('Cut.val'))

# Prepare summary data

summary = {
    'Data': dataset,
    'Obj': int(Master.obj['TotalRawRolls'].value()),
    'Waste': Master.getValue(
        'sum {p in PATTERNS} Cut[p] * \
        (rawWidth - sum {w in WIDTHS} w*rolls[w,p])'
    )
}

# Retrieve patterns and solution

npatterns = int(Master.param['nPatterns'].value())
rolls = Master.param['rolls'].getValues().toDict()
cutvec = Master.var['Cut'].getValues().toDict()
```

Pattern Generation in Python

Display solution

```
# Prepare solution data
solution = [
    ([int(rolls[widths[i], p+1][0])
     for i in range(len(widths))], int(cutvec[p+1][0]))
    for p in range(npatterns)
    if cutvec[p+1][0] > 0
]

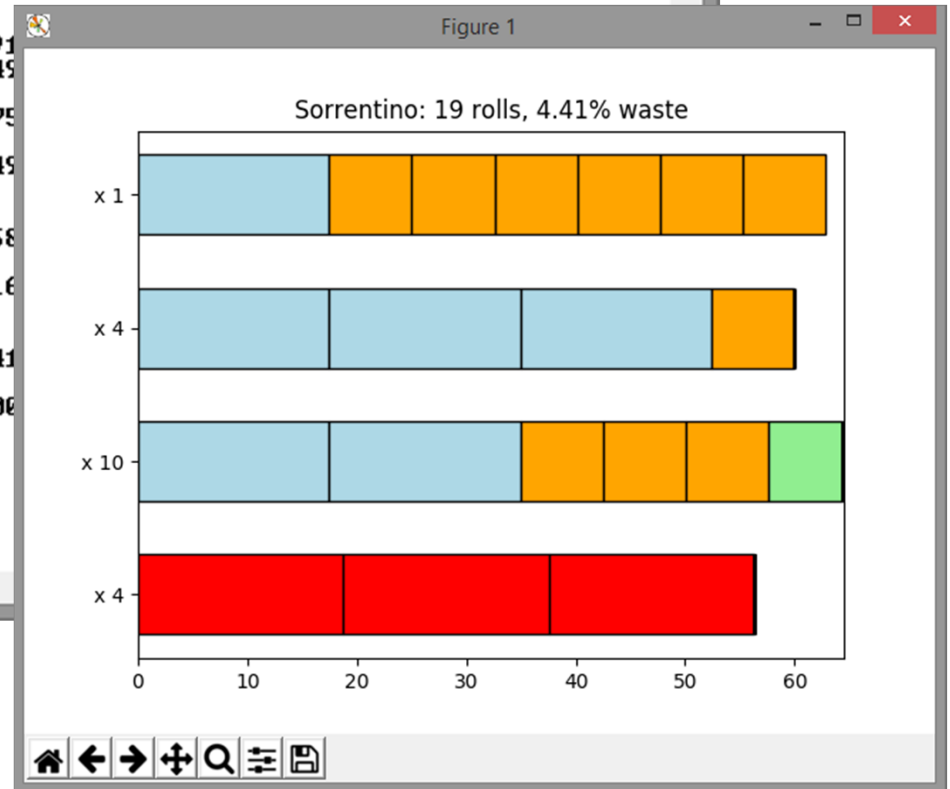
# Create plot of solution
cuttingPlot(roll_width, widths, summary, solution)
```


Pattern Generation in Python

```
sw: running ipython
File Edit Help
sw: ipython --no-banner

In [1]: from pattern_generation import *

In [2]: cuttingGen('Sorrentino')
Gurobi 7.5.0: optimal solution; objective 20.44444444
Gurobi 7.5.0: optimal solution; objective 1.152777
2 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 18.791
Gurobi 7.5.0: optimal solution; objective 1.1249
1 simplex iterations
Gurobi 7.5.0: optimal solution; objective 18.375
3 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.1249
1 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 17.958
5 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.0416
5 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 17.941
5 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.0000
1 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 19
3 simplex iterations
1 branch-and-cut nodes
```



QuanDec

Server side

- ❖ AMPL model and data
- ❖ Standard AMPL-solver installations

Client side

- ❖ Interactive tool for collaboration & decision-making
- ❖ Runs on any recent web browser
- ❖ Java-based implementation
 - * AMPL API for Java
 - * Eclipse Remote Application Platform

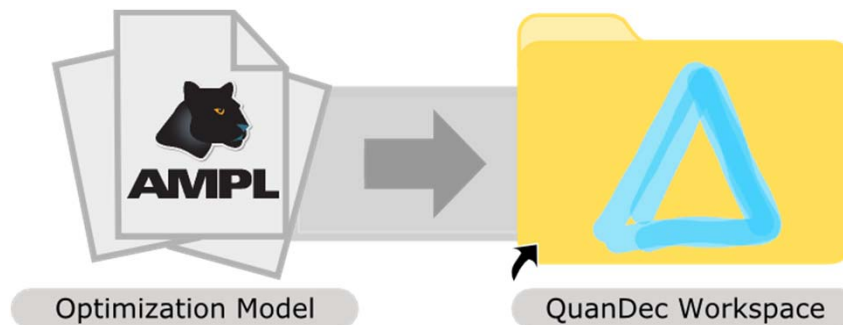
. . . developed / supported by Cassotis Consulting

Getting started

step 1: install QuanDec on a server

step 2: copy & paste your model files (.mod and .dat) into QuanDec's workspace

step 3: create AMPL tables and link them to QuanDec explorer



Quan ec

E-mail :

Password :

[Forgot?](#)

Enter your email to login

Version 2.3.1

CASSOTIS consulting

Login

Web-application

Multi-user

Secure access

Concurrent access

Workbench

Workspace
Admin

Explorer

section 1
∨

section 2
∧

category 2.1

category 2.2

category 2.3

Viewer

Charts

- Water
- Barley
- Hops
- Yeast

Report tables

- Export
- Edit bounds
- Comment
- Analyze sensitivity

Input tables

- Import
- Edit values
- Edit set:
new/remove/
duplicate
- Comment

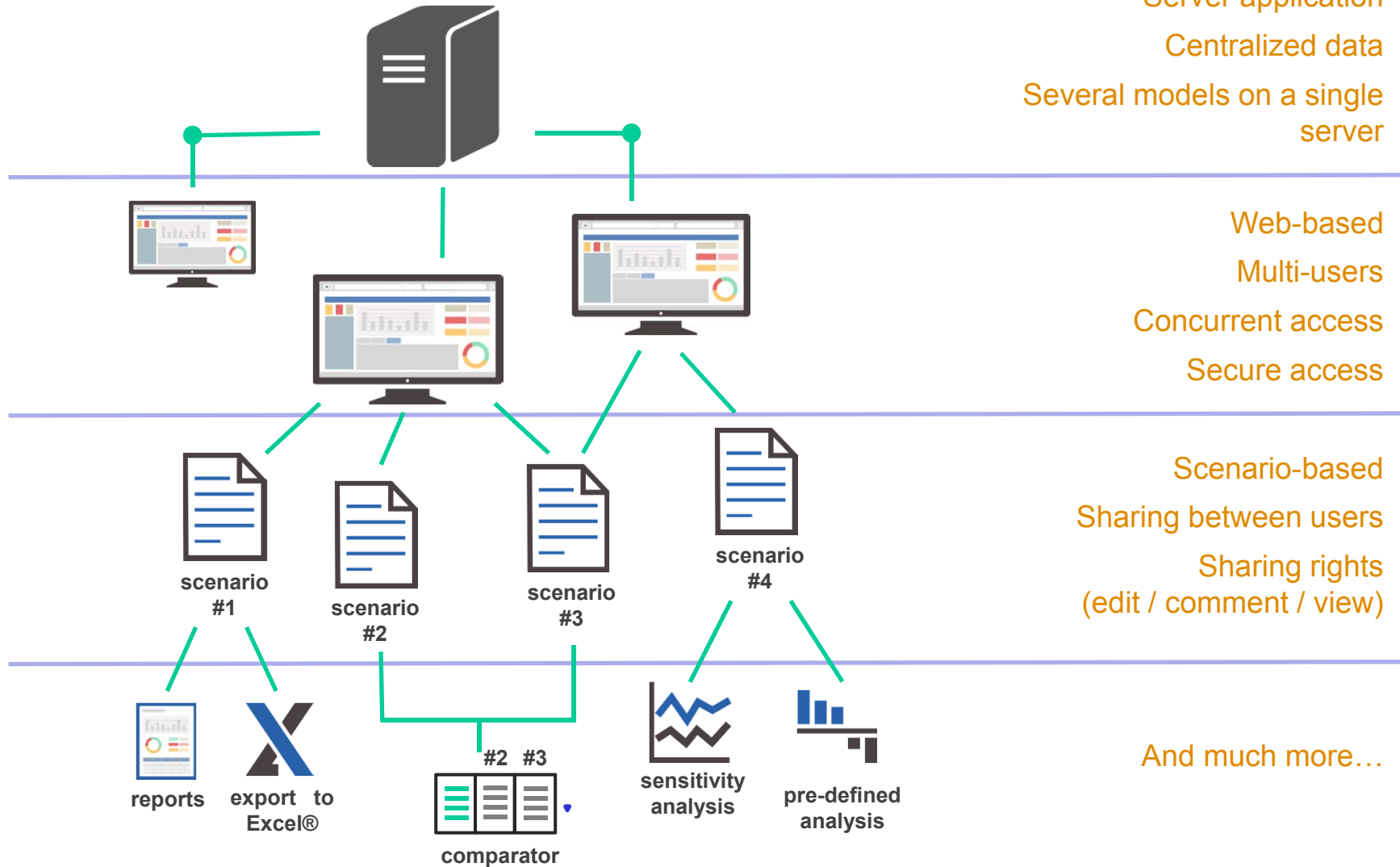
Journal | Bounds | Regressions | Comments

Console

```
> _
```


Fourer, Brandão, Valente, Integrating Optimization Modeling with Programming
INFORMS Annual Meeting, Houston, 22-25 October 2017 39

Features





Workspace Admin





Switch workspace New Master Import Master Compare

Quan  ec

This week

Name	Owner	Last change
 BUDGET 2016	Mary Torres	September 9, 2016 4:59 PM
 My Scenario	Me	Today 10:54 AM



All

Name	Owner
 BUDGET 2015	Mary Torres
 BUDGET 2016	Mary Torres
 My Scenario	Me
 FORECAST 2017	Mary Torres

Share with others

Anyone can comment

People or groups

 Robert Finn can edit 

OK

Scenario-based environment

Sharing system

**Permission:
Edit – Comment - View**

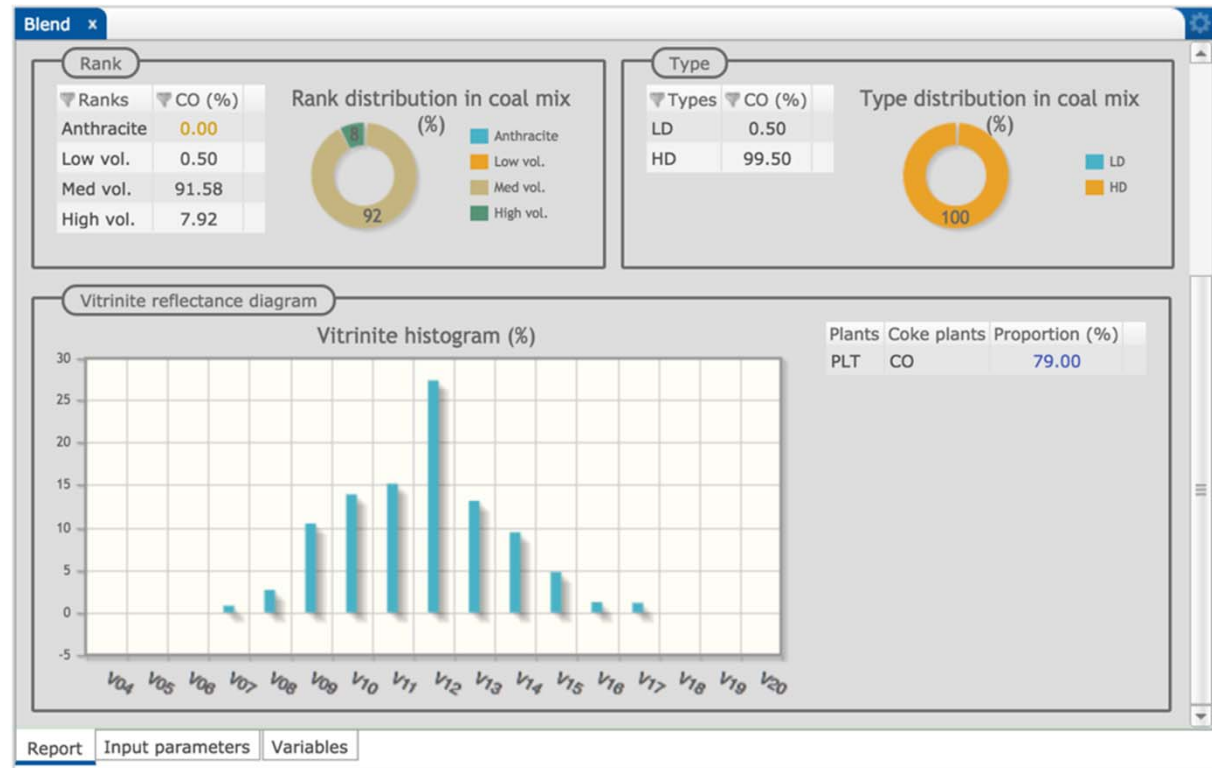
3 levels:

- Report
- Input parameters
- Variables

Chart and tables

Colored values
for easier analysis

Constraint (min/max)
on any variable



Collaborative work

Notification system

Comments between users

The screenshot displays a software interface with several components:

- Table:** A table with columns for 'blend', 'SM1 (kg/t)', and 'WAVG (kg/t)'.

blend	SM1 (kg/t)	WAVG (kg/t)
Hot metal	889.89	889.89
Lump ores	0.00	0.00
Pellets	0.00	0.00
- Chart:** A horizontal bar chart titled 'Blend at conver' showing the composition of materials. The x-axis ranges from 0 to 1200. The legend includes Hot metal (blue), Lump ores (orange), Pellets (yellow), Recyclings (green), Fluxes (dark green), and Ferroalloys (brown).
- Pop-up Windows:**
 - Comment this value:** A dialog box with a dropdown menu set to 'QUESTION'. It shows two previous comments: one from Mary Torres asking 'Do we use pellets at the converter?' and one from Benjamin Steward replying 'No, we exclusively use lump ores.' There is a text input field for a new message and 'Cancel' and 'OK' buttons.
 - New comment:** A notification box in the top right corner stating 'Robert Finn has commented this dataset.'
 - Share with others:** A dialog box with a dropdown menu set to 'Anyone can comment'. It lists 'People or groups' with 'Robert Finn' and 'can edit' (with a red X icon). There is a text input field and an 'OK' button.

Coke plants x

Operating costs

Plants	Coke plants	Costs	Fixed (MUS\$/year)	Variable (US\$/t)
PLT	CO	Maintenance	7.75	0.90
PLT	CO	Labour costs	3.95	0.00
PLT	CO	Utilities	0.05	0.11
PLT	CO	Water treatment	7.78	0.00
PLT	CO	Court yard	5.36	0.00
PLT	CO	Services	0.02	0.94
PLT	CO	Indirect costs	2.57	0.00
PLT	CO	Depreciation	4.92	0.00
PLT	CO	Electricity	0.00	0.03

Report Input parameters Variables

Journal Bounds Regressions Comments Error Log

Journal	Bounds	Regressions	Comments	Error Log
Operating cost at coke plant	PLT, CO1, co_elec, Variable	0.03	Today 11:26 AM	by Arthur Turner
CO operational costs	PLT, co_elec	Electricity	Today 11:26 AM	by Arthur Turner
CO operational costs	PLT	co_elec	Today 11:26 AM	by Arthur Turner
Vitrinite reflectance inside of range at coke plant	PLT, CO1	MAX 79.00	Today 10:49 AM	by Arthur Turner

Arthur Turner QuanDec STEEL BUDGET 2016 My Scenario

Scenarios with changes history

Traceability and undo system

Workspace Admin

New Report Show/Hide differences Export to Excel

Quan^Δec

Comparator

Variable	Unit	BUDGET 2016	My Scenario	Diff
Executive summaries				
Costs and Revenues				
Profit and Sales				
Production costs				
Absolute costs	MUS\$			
Detailed costs	US\$/t			
Internal price of intermedi	US\$/t			
Net production level	kt			
'PLT' 'CO'	kt	1763.98	1764.25	0.02%
'PLT' 'SI'	kt	4085.77	4084.46	-0.03%
'PLT' 'BF'	kt	5062.62	5060.91	-0.03%
'PLT' 'ST'	kt	5258.29	5256.75	-0.03%
'PLT' 'PO'				
Production cost of prod				
Production level				
Material blends				
Coke plants				
Sinter plants				
Blast furnaces				
Steel shops				
Power plant				
Raw materials				

Select the scenarios to compare:

- BUDGET 2015
- BUDGET 2016
- My Scenario
- FORECAST 2017

Cancel OK

Economics and Production

Variable	Index	Unit	BUDGET 2016	My Scenario	Diff
Economics per int. plant	'PLT' 'costs'	MUS\$	1515.59	1515.20	-0.03%
Economics per int. plant	'PLT' 'revenues'	MUS\$	1762.23	1761.77	-0.03%
Economics per int. plant	'PLT' 'profit'	MUS\$	246.64	246.56	-0.03%
Economics per int. plant	'PLT' 'margin'	%	14.00	14.00	-0.00%
Production cost of product	'PLT' 'coke'	US\$/t	164.48	164.54	0.04%
Production cost of product	'PLT' 'sinter'	US\$/t	77.55	77.50	-0.06%
Production cost of product	'PLT' 'hotmetal'	US\$/t	193.95	193.99	0.02%
Production cost of product	'PLT' 'slab'	US\$/t	286.27	286.28	0.00%
Production cost of product	'PLT' 'electricity'	US\$/MWh	125.75	125.75	0.00%
Production level of product	'PLT' 'coke'	kt	1818.54	1818.81	0.02%
Production level of product	'PLT' 'sinter'	kt	4085.77	4084.46	-0.03%

Report Structure

Reports

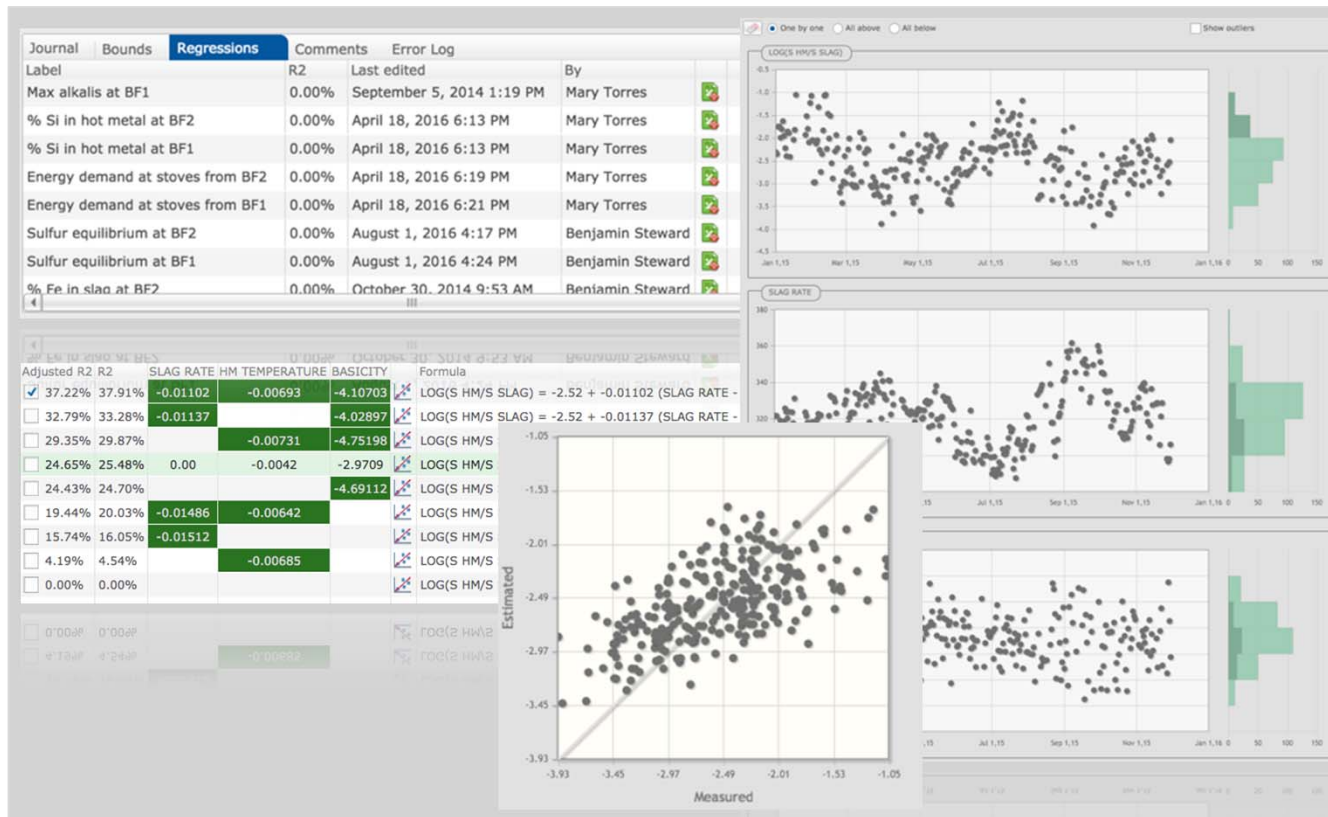
Name	User	Date	Action
Sulfur cycle	Benjamin Steward	March 18, 2016 3:45 PM	✖
Metallic blend at CV	Me	February 21, 2016 4:51 PM	✖
Raw material use at Reduction	Me	January 15, 2016 4:36 PM	✖
Economics and Production	Mary Torres	September 13, 2016 4:53 PM	✖
Flux consumption at Torpedo	Mary Torres	April 3, 2016 4:44 PM	✖
Slab sales	Robert Finn	January 30, 2016 5:30 PM	✖
Silicon cycle	Benjamin Steward	July 5, 2016 4:17 PM	✖

Scenario comparison

All variables can be compared

Display of relative difference

Custom reports



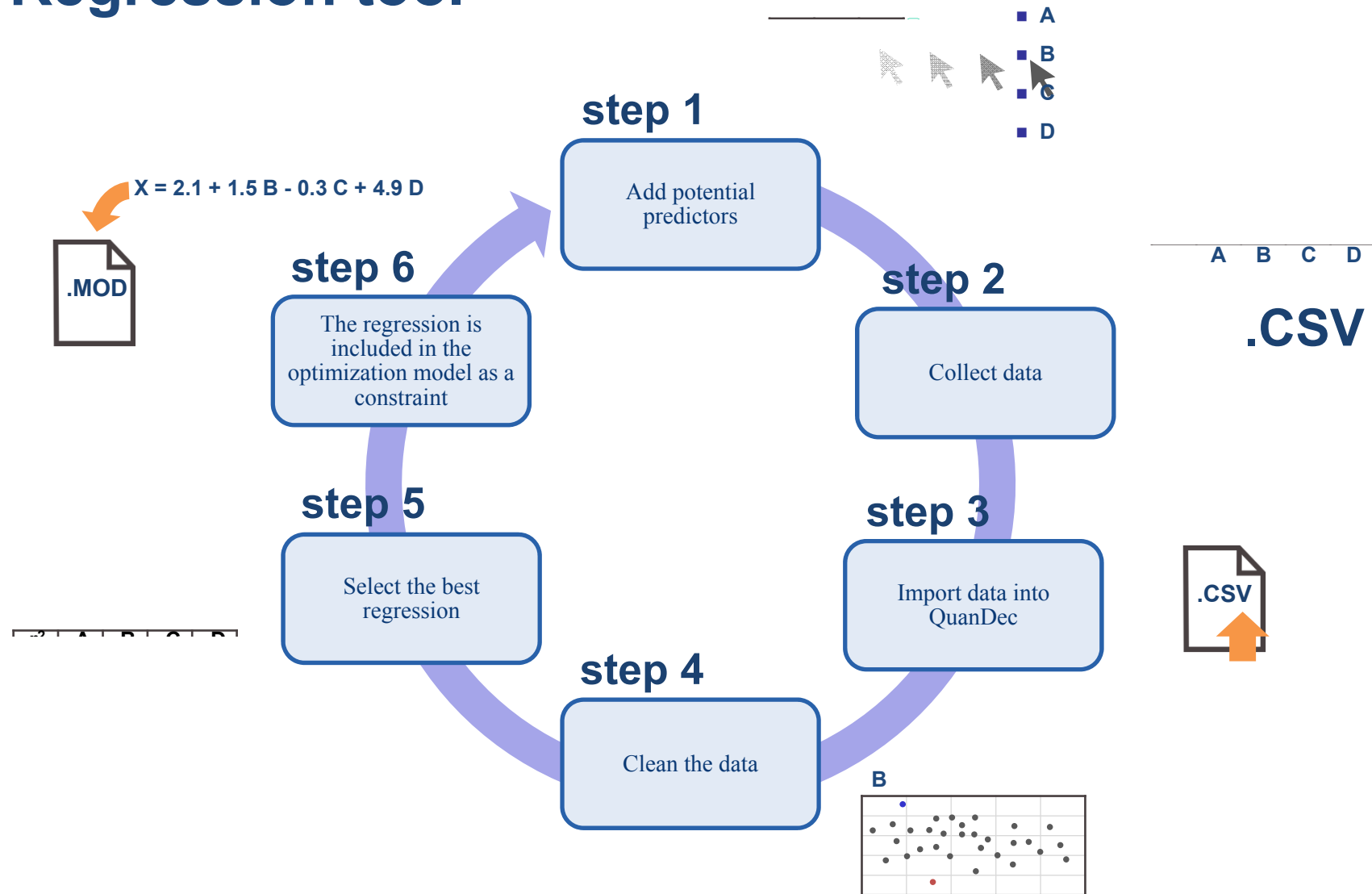
Regression tool

Data cleaning

Any variable can be added to a regression

Manual coefficients if no data available

Regression tool



Profit and Sales

Exchange rates

Currencies	Exchange rates (US\$)
eur	1.14
usd	1.00
brl	0.29

Horizon

Days (d) 365.00

- Add a comment
- Add a constraint
- Add to a regression
- Analyse sensitivity
- Export the table
- Download the template

Sensitivity analysis

Parameter : Exchange rates

Index : 'brl'

From : 0.3

To : 1

Pts : 3

Cancel OK

Sensitivity analysis

For both parameters AND variables

All variables can be compared

Display of relative difference

Workspace Admin

Back to edition New Report Show/Hide differences Export to Excel

QuanDec

Comparator

Variable	Unit	0.30	0.65	Diff	1.00	Diff
Executive summaries						
Costs and Revenues						
Profit and Sales						
Economics per int. plant	MUS\$					
'PLT' 'costs'	MUS\$	1515.39	1544.99	1.95%	1633.34	7.78%
'PLT' 'revenues'	MUS\$	1754.70	1679.96	-4.26%	1670.71	-4.79%
'PLT' 'profit'	MUS\$	239.31	134.97	-43.60%	37.37	-84.38%
'PLT' 'margin'	%	13.64	8.03	-41.09%	2.24	-83.60%
Global economics	MUS\$					
External costs per process	MUS\$					
External costs per type	MUS\$					
Detailed external costs	MUS\$					
External revenues per process	MUS\$					
External revenues per type	MUS\$					
Detailed external revenues	MUS\$					
Detailed revenues	MUS\$/t					
Production costs						
Material blends						
Coke plants						
Sinter plants						
Blast furnaces						
Steel shops						
Power plant						
Raw materials						
Gases						

Economics and Production

Variable	Index	Unit	0.30	0.65	Diff	1.00	Diff
Economics per int. plant	'PLT' 'costs'	MUS\$	1515.39	1544.99	1.95%	1633.34	7.78%
Economics per int. plant	'PLT' 'revenues'	MUS\$	1754.70	1679.96	-4.26%	1670.71	-4.79%
Economics per int. plant	'PLT' 'profit'	MUS\$	239.31	134.97	-43.60%	37.37	-84.38%
Economics per int. plant	'PLT' 'margin'	%	13.64	8.03	-41.09%	2.24	-83.60%
Production cost of product	'PLT' 'coke'	US\$/t	164.51	161.52	-1.82%	162.71	-1.10%
Production cost of product	'PLT' 'sinter'	US\$/t	77.68	83.23	7.15%	88.16	13.50%
Production cost of product	'PLT' 'hotmetal'	US\$/t	194.23	198.43	2.16%	202.93	4.48%
Production cost of product	'PLT' 'slab'	US\$/t	287.62	307.33	6.85%	326.85	13.64%
Production cost of product	'PLT' 'electricity'	US\$/MWh	125.62	125.73	0.08%	125.74	0.09%
Production level of product	'PLT' 'coke'	kt	1818.81	1815.95	-0.16%	1815.95	-0.16%
Production level of product	'PLT' 'sinter'	kt	4115.36	4007.25	-2.63%	4006.24	-2.65%
Production level of product	'PLT' 'hotmetal'	kt	5105.94	5051.71	-1.06%	5052.00	-1.06%
Production level of product	'PLT' 'trhotmetal'	kt	5025.36	4972.09	-1.06%	4972.37	-1.05%
Production level of product	'PLT' 'crudesteel'	kt	5657.39	5402.17	-4.51%	5372.49	-5.04%

Report Structure

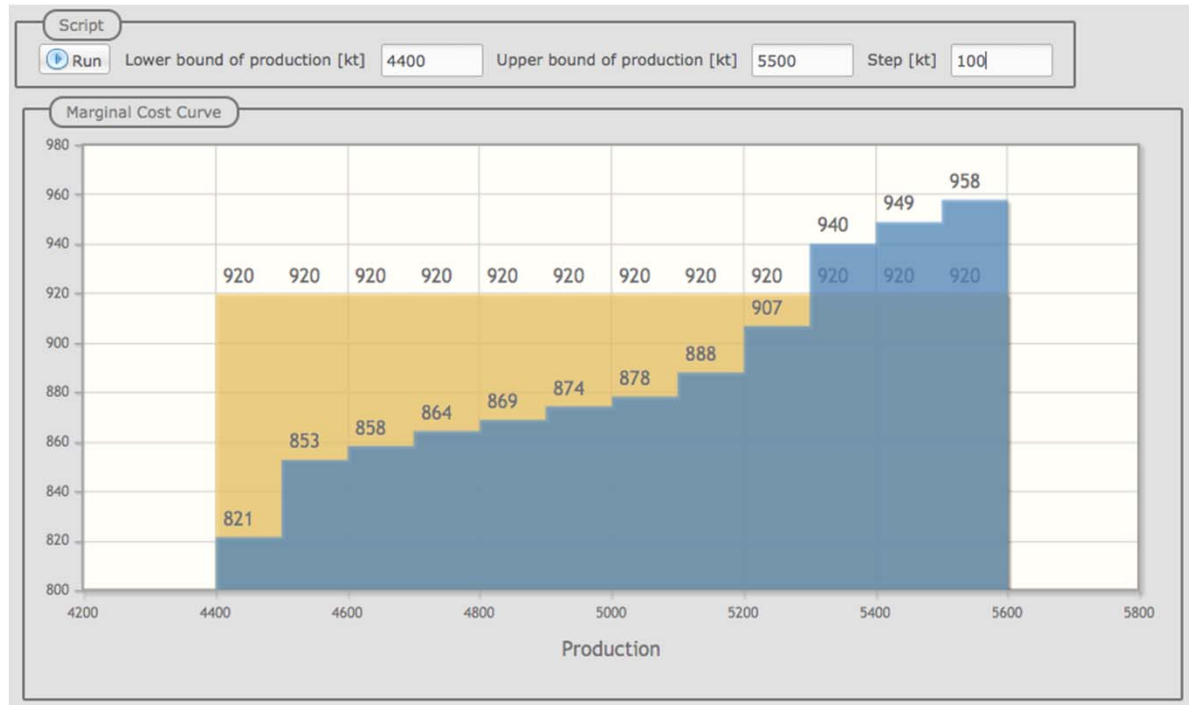
Reports

Name	User	Date	Action
Sulfur cycle	Benjamin Steward	March 18, 2016 3:45 PM	✖
Metallic blend at CV	Me	February 21, 2016 4:51 PM	✖
Raw material use at Reduction	Me	January 15, 2016 4:36 PM	✖
Economics and Production	Mary Torres	September 13, 2016 4:53 PM	✖
Flux consumption at Torpedo	Mary Torres	April 3, 2016 4:44 PM	✖
Slab sales	Robert Finn	January 30, 2016 5:30 PM	✖
Silicon cycle	Benjamin Steward	July 5, 2016 4:17 PM	✖

Arthur Turner QuanDec STEEL BUDGET 2016 My Scenario About QuanDec...

Predefined analyses

Script parameters



QuanDec Availability

Ready now for commercial applications

- ❖ Free trials available
- ❖ Pricing keyed to number of models & users

First year's support included

- ❖ Tailored setup support from Cassotis Consulting
- ❖ Customizations possible

... contact sales@ampl.com for details