# Optimization

## Using Analytic Solver Platform

**REVIEW BASED ON**
**MANAGEMENT SCIENCE**


FrontlineSolvers

# What We'll Cover Today

- Introduction
    - Session III beta training program goals
- Automating Optimization in Excel
- Defining Optimization in Solver SDK Platform
- Dimensional Modeling

# Session III Online Beta Training Goals

**To familiarize you with the following concepts:**

- Automating your Excel spreadsheet optimization model
- Designing custom applications
- Modeling multi-dimensional business situations

**To empower you to achieve success**

- State of the art tools
- Online educational training
- User guides and video demos

# Typical Optimization Applications

***Industry***

- Energy
- Chemical
- Manufacturing
- Transportation
- Finance
- Agriculture
- Health
- Mining
- Defense
- Forestry

***Functional Area***

- Staff planning
- Scheduling
- Routing
- Blending
- Capacity planning
- Media planning
- Supply chain
  - Inventory optimization
  - Vendor selection
- Portfolio optimization
- Product mix

# Why Automate Optimization Models?

- Run predefined Solver models and get the results.
- Solve a complex problem by optimizing a “master” model and a “slave” model.
- Automatically update data, run the model, and retrieve the results.
- Run multi-period optimizations when the output of current model is the input to the next model.
- Create an application to distribute to end users.

# Why Use the Object-Oriented API?

- Create, modify and solve optimization models under the control of your custom application written in VBA.
- Work with objects that correspond to the Problem, Model, Solver, Engine, Variables, and Functions.
- You can access sets of variables and constraints in the current model directly with expressions such as myProb.VarDecision and myProb.FcnConstraint.
- You'll receive **IntelliSense** prompts as you write code.

# Examples in the User Guide

- **Refinery.xls** at C:\Program Files\Frontline Systems\Analytic Solver Platform\Examples.
  - Example of how to call the Solver, and how to retrieve sensitivity information from VBA.
  - Refinery Optimization - Model taken from Model Building In Mathematical Programming by H.P. Williams.
  - How should the operations of the refinery be planned to maximize total profit.
- **CuttingStock.xls** at C:\Program Files\Frontline Systems\Analytic Solver Platform\Examples.
  - Example of how VBA can be used to solve a cutting stock problem, using a column-generation algorithm.

- Press the "Run Model" button to execute the RunModel macro that calls the Solver programmatically through VBA.

# Automating Optimization Modeling in VBA Example

- Standard Examples - Example 4  Portfolio Optimization - Markowitz Method.
- Find the optimal allocation of funds to stocks that minimizes the portfolio risk, measured by portfolio Variance (a quadratic function).

| | Stock 1 | Stock 2 | Stock 3 | Stock 4 | Stock 5 | Total |
|---|---|---|---|---|---|---|
| **Portfolio %** | 20.00% | 20.00% | 20.00% | 20.00% | 20.00% | 100.00% |
| **Expected Return** | 7.00% | 8.00% | 9.50% | 6.50% | 14.00% | |
| ***Linear QP Terms*** | *0* | *0* | *0* | *0* | *0* | |

**Variance/Covariance Matrix**

| | Stock 1 | Stock 2 | Stock 3 | Stock 4 | Stock 5 |
|---|---|---|---|---|---|
| **Stock 1** | 2.50% | 0.10% | 1.00% | -0.50% | 1.00% |
| **Stock 2** | 0.10% | 4.00% | -0.10% | 1.20% | -0.85% |
| **Stock 3** | 1.00% | -0.10% | 1.20% | 0.65% | 0.75% |
| **Stock 4** | -0.50% | 1.20% | 0.65% | 8.00% | 1.00% |
| **Stock 5** | 1.00% | -0.85% | 0.75% | 1.00% | 7.00% |

# Summary Automating Predefined Optimization Models in VBA

- Step 1 – Press Alt+F11 to open the Visual Basic Editor.
- Step 2 – Add a reference to the Analytic Solver Platform COM server
  **Analytic Solver Platform 2014 Type Library**.
- Step 3 – Create a new macro
- Step 4 – Add two lines of code for active worksheet

```
Dim prob As New Problem
prob.Solver.Optimize
```

# VBA Example Summary – Create and Solve the Model

- Step 1 – Create a command button on the worksheet.
- Step 2 – Assign a macro to the command button.
- Step 3 – Add a reference to the Analytic Solver Platform COM server.
- Step 4 – Create an instance of the problem.

```
Dim myProb As New RSP.Problem
```

- Step 5 – Clear the existing model.

```
myProb.Functions.Clear
myProb.Variables.Clear
```

- Step 6 – Set Cell or Objective Variable

```
Dim objective As New RSP.Function
objective.Init Range("Portfolio_Variance")
objective.FunctionType = Function_Type_Objective
```

# VBA Example Summary – Create and Solve the Model

- Step 7 – Add the Objective to the problem

  myProb.Functions.Add objective

  Set objective = Nothing

- Step 8 – Set up the Variables and add the non-negativity constraint

  Dim vars As New Variable

  vars.Init Range("Allocations")

  vars.NonNegative

- Step 9 – Add Variables to problem

  myProb.Variables.Add vars

  Set vars = Nothing

# VBA Example Summary – Create and Solve the Model

- Step 10 – Set up the return threshold constraint

```
Dim constraint As New RSP.Function
constraint.Init Range("Portfolio_Return")
constraint.LowerBound(0) = 0.095
```

- Step 11 – Set up the budget constraint

```
Dim constraint1 As New RSP.Function
constraint1.Init Range("Total_Portfolio")
constraint1.UpperBound(0) = 1
constraint1.LowerBound(0) = 1
```

- Step 12 – Add the constraints to the problem

```
myProb.Functions.Add constraint
myProb.Functions.Add constraint1
```

# VBA Example Summary – Create and Solve the Model

- Step 13 – Set the problem type

  myProb.Solver.SolverType = Solver_Type_Minimize
- Step 14 – Perform the optimization

  myProb.Solver.Optimize
- Step 15 – Display the Solver Objective function result

  MsgBox myProb.FcnObjective.FinalValue(0)
- Save the code. Now click on the command button and see the results.

- You can also set the desired engine and adjust its parameters.

  myProb.Engine = prob.Engines("Standard LP/Quadratic")

  myProb.Engine.Params("MaxTime") = 600

# Solver SDK Platform

- A complete toolkit to move your optimization model from Excel to a custom desktop or web-based application.
- Enables you to develop and deploy custom applications using optimization and Monte Carlo simulation, with most popular platforms and languages: Microsoft .NET, Java, MATLAB and COM, as well as C/C++, Visual Basic and other languages.
- The SDK can load an Excel workbook containing an optimization model, solve the model on a server without using Excel, and save the solution in the Excel workbook.
- SDK exposes a standards-based Web Service API - enabling you to create models in PHP and JavaScript, even in a web browser or mobile phone, solve them "over the wire," and deliver the solution in your "zero-footprint" application!

# Loading and Solving Excel Optimization Models in SDK

- To load a model built with Premium Solver Platform (C#)

Step 1 – Create an instance of the problem:

```
Using (Problem prob = new Problem()) {
```

Step 2 – Point to the problem:

```
prob.Load("C:\\....",File_Format.XLStd);
```

Step 3 – Solve the model:

```
prob.Solver.Optimize();
```

# Solver SDK Evaluators

- Use Solver SDK evaluators or callback functions for:
- Checking the progress of the Solver during the optimization process (Progress Information Evaluators)
  - Check the optimization iteration number, report the current objective, check for a user abort code, etc.
- Provide the crucial role of computing constraint function values or passing gradient or Hessian values when solving an optimization problem (Computing Evaluators).
  - Calculate the objective or constraint function values.
  - Use an evaluator of type Eval_Type_Function.

# SDK Example Summary – Creating a C# Project

- Step 1 – Create a new project.
  - Start Microsoft Visual Studio, select File | New Project, select C# on the left of the dialog and Windows Forms Application from the right of the dialog under Templates.
  - Type a name of your choice in the Project name field, and then click OK.
  - Construct a form of your choice using the Visual Studio Toolbox.
- Step 2 – Set a reference.
  - Click References in the Solution Explorer and select Add Reference. Choose your .NET version: 1.1, 2.0-3.5, or 4.0-4.5.

# SDK Example Summary – Creating a C# Project

- Step 3 – add an (optional) directive
  - *Using SolverPlatform*
- Step 4 – Choose your approach to create the model
  - In a linear model or quadratic model, you can supply the Solver SDK with the objective and constraint coefficient matrices.
  - Or you can supply an evaluator (Eval_Type.Function) to compute the function (objective and constraint) values.
    - Create and pass the function evaluator. prob.Evaluators[Eval_Type.Function].OnEvaluate +=
    - This evaluator will be called at every iteration, to compute the values of the objective and constraints.

# SDK Example
# Summary - Creating a C# Project

- Step 5 – Create the model
  - Create an instance of the Problem class.

  *Problem prob = new problem(Solver_Type.Minimize, nvars, ncons)*.
  - Add variable and constraint definitions to the problem.
  - Call prob.Solver.Optimize().

# SDK Example
# Summary - Creating a C# Project

- Step 6 – Write the function evaluator if used.

  Problem p = evaluator.Problem;

  - Obtain a pointer, pVar, to the variables.

  double[] pVar = p.VarDecision.Value.Array;

  - Next, obtain a pointer to the constraint functions and calculate the constraints.

  double[] pFcn = p.FcnConstraint.Value.Array;

  pFcn[0]= *formula*; pFcn[1]= *formula*;

  - Calculate the objective.

  p.FcnObjective.Value[0]= *formula*;

  - Pass the constraint values back to the SDK.

  p.FcnConstraint.Value.Array = pFcn;

  - Tell the SDK to continue optimizing.

  Return Engine_Action.Continue;

# Building Multi-Dimensional Analytical Models

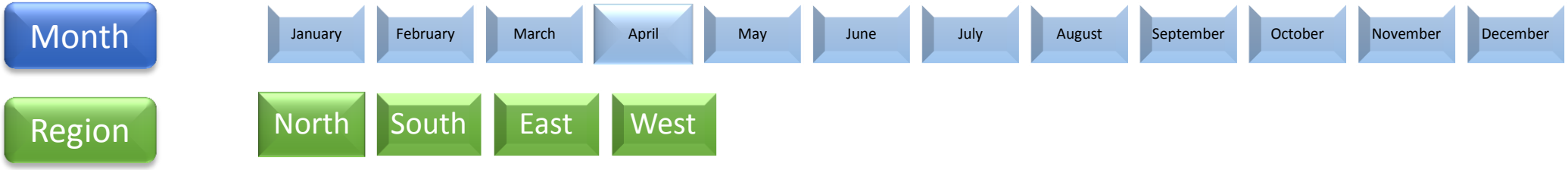
## ASP Dimensional Modeling

# Why Use Dimensional Modeling?

- Dimensional modeling turns a very complex, hard-to-maintain spreadsheet problem into a simple, well-structured, easily maintained and expandable model.
  - Where there are multiple products, projects or investments, multiple customer types, geographic regions, or time periods, multiple sources and destinations.
- Excel with PowerPivot is an industrial-strength tool for slicing and dicing multi-dimensional data from a variety of data sources.
- Now create multi-dimensional models, with formulas that reflect the dynamics of existing and future business situations, and easily link those models to Pivot Table data.

# Elements of Dimensional Modeling – Dimensions

- A set of elements.




- Give a name ( “Region”) for the dimension, and provide names/labels or numbers for the elements of the dimension – “North,” “South,” “East” and “West.”
- A dimension doesn’t define the data itself – it defines a structure, relevant for your business situation, for the data.

# Elements of Dimensional Modeling – Cubes

- Cubes are multi-dimensional arrays holding the data.
- Other names used in data warehouse and business intelligence systems, are *measures* and/or *fact tables*.
- A cube holds a single attribute or measure (a set of numbers).
- Example: create a cube named “Sales,” defined over the “Regions” dimension, with numeric values representing sales in the North, South, East and West regions.
- The structure of a cube is very much like a Pivot Table, and you can create a cube by placing a PsiPivotCube() function in a cell, referencing an existing Pivot Table.

# Cube Formulas – Operations and Dimensions

- You can multiply, add, subtract, divide, and perform other operations on cubes.
- If all participating cubes have the *same dimensions*, result cube will have those dimensions.
  - Cube (4 dimensions) X Cube (same 4 dimensions) = Cube (same 4 dimensions).
- Otherwise, the result cube will be a *union* of the participating dimension sets.
  - Cube B2 (2 dimensions) X Cube C3 (1 dimension that also appears in the first cube) = B2*C3 (2 dimensional).
  - Cube B2 (2 dimensions) X Cube C3 (1 dimension that doesn't appear in the first cube) = B2*C3 (3 dimensional).

# Cube Formulas – Reduction

- Cube Reduction is an operation of eliminating one or more dimensions by aggregating (for example summing) values along those dimensions.
- PSI supports reduction along one dimension only or along all dimensions.
- A cube reduced along all its dimensions will be a single value, or scalar.
- A cube can be aggregated using : Average, Sum, Maximum, Minimum, Variance, std. Deviation, Element, or Index.

Dimension: Region | North | South | East | West

Cube: “Sales” 100 120 150 300

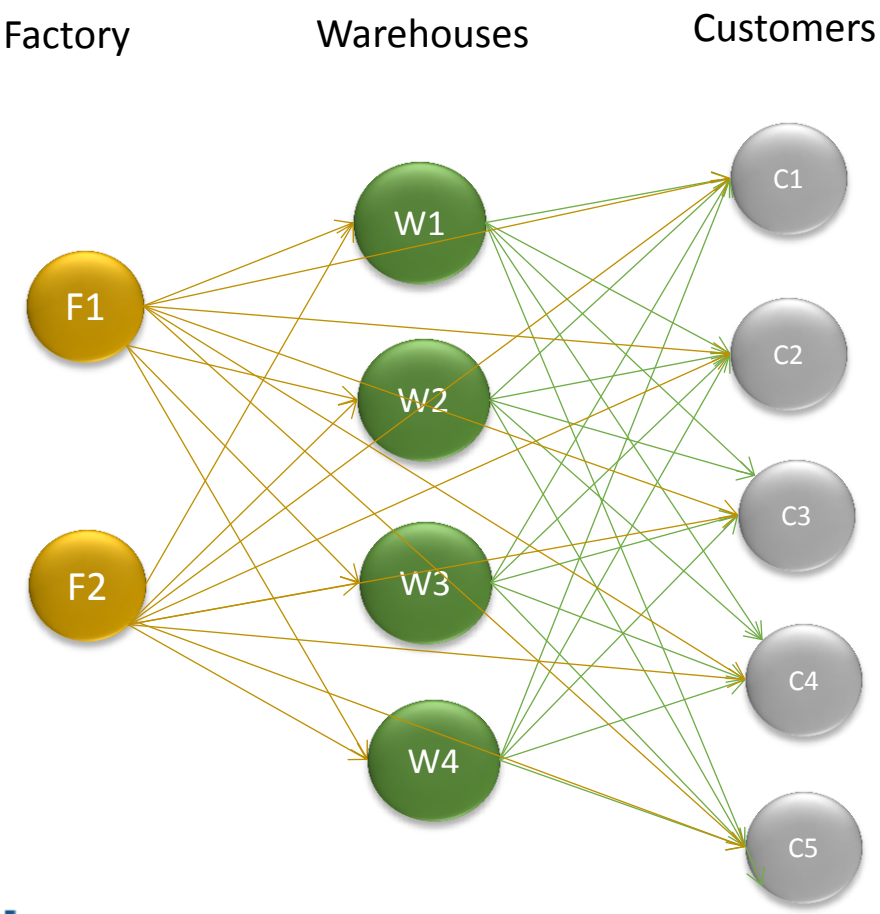
Reduce Cube using Sum: 100+120+150+300 = 670

# Elements of Dimensional Modeling – Output Cell Range

- When we want to see results of optimization of a dimensional model on the spreadsheet, we use {PsiOptData()}.
- When we want to see other computed cube results on the spreadsheet, we use {PsiCubeData()}.
- The cell range will display all the values of the cube that is an argument to PsiOptData() or PsiCubeData().
  - PsiOptData() results appear after optimizing.
  - PsiCubeData() results appear after Model – Cube Result – Calculate.

# Transshipment Problem (2-Stage-Transport, Multi-Commodity)

- Minimize the costs of shipping 3 different goods from factories to warehouses and customers, and warehouses to customers.
- While not exceeding the supply available from each factory or the capacity of each warehouse, and meeting the demand from each customer.

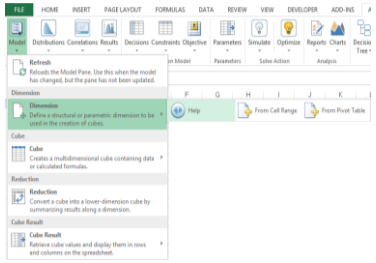




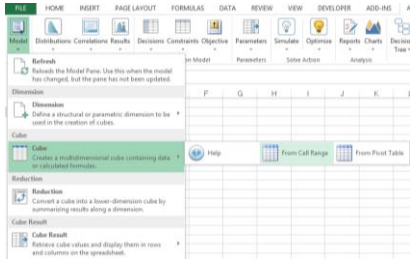

FrontlineSolvers
WE DEMOCRATIZE ANALYTICS

# Dimensional Modeling – Summary of Steps

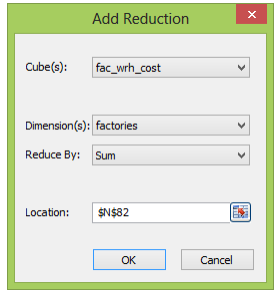
- To add a dimension click Model – Cube – From Cell Range.



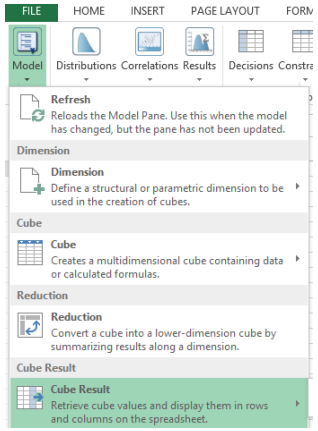
- To add a cube click on Model Tab on the RSP Ribbon. Choose Cube.



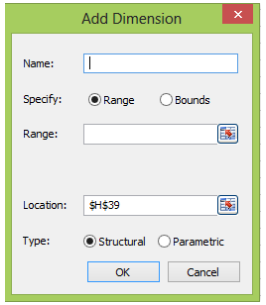
- Click Model – Reduction.



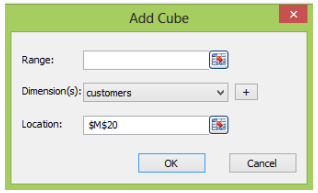

- Define Outputs.




- Select from Cell Range. Input name, range, location, and type.




- Select the data range, related dimension/s and the location of the cube.




FrontlineSolvers

WE DEMOCRATIZE ANALYTICS

# Third Session Summary

- Automating Excel spreadsheet optimization models using Analytic Solver Platform Object-Oriented API.
  - Create, modify and solve optimization models under the control of your custom application written in VBA.
- Moving optimization models from Excel to a custom desktop or web-based application using Solver SDK Platform.
  - Develop and deploy custom applications with most popular platforms and languages: Microsoft .NET, Java, MATLAB and COM, as well as C/C++, Visual Basic and other languages.
- Modeling multi-dimensional business situations.
  - Define Dimensions, Cubes, and Cube Outputs to turn a very complex, hard-to-maintain spreadsheet problem into a simple, well-structured, easily maintained and expandable model.

# Final Recap

- It is increasingly important to efficiently use limited resources. Optimization can often improve efficiency with no capital investment.
- Building a model often reveals relationships and yields a greater understanding of the situation being modeled.
- Having built a model, it is possible to apply analytic methods to suggest courses of action that might not otherwise be apparent.
- Experimentation is possible with a model, whereas it is often not possible, or desirable, to experiment with the situation being modeled.
- Analytic Solver Platform is a complete toolset for creating analytic models in Excel.
- Solver SDK Platform is a complete toolset for deploying analytic applications to end users.

# Contact Info

- Dr. Sima Maleki
- Best way to contact me: Consulting@Solver.com
- You may also download this presentation from our website at www.solver.com/training/premsolver-3.
- You can download a free trial version of Analytic Solver Platform at Solver.com.

# References

- **Solver SDK User Guide**

C:\Program Files\Frontline Systems\Solver SDK Platform\Help

- **Solver User Guide**

C:\Program Files\Frontline Systems\Analytic Solver Platform\Help

- **Management Science-The Art of Modeling with Spreadsheets, 4th Edition**

http://www.wiley.com/WileyCDA/WileyTitle/productCd-EHEP002883.html

FRONTLINE
solvers

# Q & A

FRONTLINE
solvers

# Thank You!