Outline

- Why Modeling Languages?
- Types of Modeling Languages
- Intro to Sample Problem
- Examination of:
  - AMPL (Kristin)
  - GAMS (Nael)
  - AIMMS (Olesya)
  - FlopC++ (Doron)
  - Creators, Aims, Applications
  - Key Features, Drawbacks
  - Compatible Solvers, Cost
  - Sample Problem Exercise
- Practicality
- Conclusions
Why Modeling Languages?

- Designed to eliminate the intermediate step of translating the “modeler’s form” to the “algorithm’s form”
  - Makes mathematical programming more economical and more reliable
- Particularly useful in model development (when the model is subject to revisions)
- Allows several solvers to be available at once
Types of Modeling Languages

- **Algebraic modeling languages**, which employ familiarity of traditional mathematical notation to describe objective functions and constraints.

- **Block-schematic diagrams**, which depict a linear constraint matrix as a collection of structured submatrices (or blocks).

- **Activity specifications**, which describe a model in terms of activities (variables) and their effects (constraints) on inputs and outputs.

- **Netforms**, which use graph or network diagrams to depict models involving flows and allocations.
Intro to Sample Problem

- Dofasco needs to schedule its two products on the rolling mill.
- If 40 hours of production time are available, what quantity of each product should be made in order to maximize profits?

<table>
<thead>
<tr>
<th>Product</th>
<th>Tons per hour</th>
<th>Profit per ton</th>
<th>Max tons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bands</td>
<td>200</td>
<td>$25</td>
<td>6,000</td>
</tr>
<tr>
<td>Coils</td>
<td>140</td>
<td>$30</td>
<td>4,000</td>
</tr>
</tbody>
</table>
Intro to Sample Problem

\[
\begin{align*}
\text{max} & \quad 25x_B + 30x_C \\
\text{s.t.} & \quad (1/200)x_B + (1/140)x_C \leq 40 \\
& \quad 0 \leq x_B \leq 6,000 \\
& \quad 0 \leq x_C \leq 4,000
\end{align*}
\]
AMPL: A Mathematical Programming Language

www.ampl.com

Creators
- Bell Laboratories (www.bell-labs.com)

Aims
- Linear and nonlinear optimization problems
- Either continuous or discrete variables (or both)

Applications
- Interactive command environment for setting up & solving mathematical programming problems
- Comprehensive & powerful algebraic modeling language
AMPL — A Mathematical Programming Language

Key Features

- **Modeling Language:**
  - Broad support for sets & set operators
  - General & natural syntax
  - NLP features
  - Convenient alternative notations

- **Modeling Environment:**
  - Interactive command environment
  - New looping and if-then-else commands
  - Separation of model & data
  - Interfaces to popular & sophisticated solvers
AMPL  A Mathematical Programming Language

Commercial/Free

- write to info@ampl.com for prices
  - Professional version
    - Single machine & floating licenses available
    - Solvers: KNITRO, CONOPT can be purchased directly from AMPL
    - Many other vendors
  - Student Edition
    - FREE!
    - limits: 300 variables, 300 (objectives + constraints)
    - Solvers: CPLEX, DONLP2, LOQO, lp_solve, MINOS, SNOPT, WSAT(OIP)
AMPL: A Mathematical Programming Language

Compatible Solvers

- **LP**
  - *Continuous*: BPMPD, CPLEX, LAMPS, LOQO, Ip_solve, MINOS, MOSEK, OSL, SOPT, XA, Xpress-MP
  - *Integer*: CPLEX, LAMPS, Ip_solve, MINTO, MOSEK, OSL, SOPT, XA, Xpress-MP
  - *Network*: CPLEX, OSL

- **NLP**
  - *Quadratic*: CPLEX, MOSEK, OSL
  - *Convex*: MOSEK, SOPT
  - *General continuous*: CONOPT, DONLP2, FILTER, FSQP, IPOPT, KNITRO, LANCELOT, LOQO, MINOS, NPSOL, PENNON, SNOPT
  - *General integer*: MINLP
Step 1.
Write file named prod0.mod

```AMPL
var XB;
var XC;
maximize Profit: 25 * XB + 30 * XC;
subject to Time: (1/200) * XB + (1/140) * XC <= 40;
subject to B_limit: 0 <= XB <= 6000;
subject to C_limit: 0 <= XC <= 4000;
```

Step 2.
In AMPL, use the following commands

```AMPL
ampl: model prod0.mod;
ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000
ampl: display XB, XC;
XB = 6000
XC = 1400
ampl: quit;
```

Is this more useful?
- No! Unless you are solving larger problems...
AMPL: A Mathematical Programming Language

Sample Problem Exercise – generalized model - notation

Given:
- \( P \), a set of products \( \text{set} \)
- \( a_j \) = tons per hour of product \( j \), for each \( j \in P \)
- \( b \) = hours available at the mill
- \( c_j \) = profit per ton of product \( j \), for each \( j \in P \)
- \( u_j \) = maximum tons of product \( j \), for each \( j \in P \)

Define variables:
- \( X_j \) = tons of product \( j \) to be made, for each \( j \in P \)

Maximize:
- \[ \sum_{j \in P} c_j X_j \] \( \text{objective} \)

Subject to:
- \[ \sum_{j \in P} (1/a_j)X_j \leq b \]
- \( 0 \leq X_j \leq u_j \), for each \( j \in P \)

Figure 1-1: Basic production model in algebraic form.
Step 1. Write the model file named prod.mod

Figure 1-2: Basic production model in AMPL (file prod.mod).

NOTE: Much easier to write this type of file rather than a specific one when dealing with many variables, parameters, etc.
Sample Problem Exercise – generalized model – in AMPL

Step 2.
Write the data file called prod.dat

```AMPL
set P := bands coils;
param: a c u :=
  bands  200  25  6000
  coils  140  30  4000 ;
param b := 40;
```

**NOTE:** Much easier to change parameters & to input large amounts of data.

Figure 1-3: Production model data (file `prod.dat`).

Step 3.
Execute commands in AMPL as shown previously.
References

- www.ampl.com
Generic Algebraic Modeling System
Nael El Shawwa
GAMS Development Corp.

- Located in:
  - Washington DC, U.S.A
  - Cologne, Germany

- GAMS distribution/support
- Research in new algorithms for GAMS
- Consulting for modeling, prototyping and for complete application development
Overview

- Problem types:
  - Linear, non-linear, mixed integer
- Useful for large and complex problems
- Can be used with Conopt, Cplex, Knitro, Mosek, Xpress, and many other solvers
- Current version 2.5 includes an IDE
- First version around 20 years ago
- Mentioned by INFORMS, CORS, IEEE, SIAM
- Very popular in academia / industry
Price

- **Base commercial:** $3,200
  - Includes links for 4 solvers
  - Links for other solvers available at additional cost

- **Base education:** $640
  - Includes links for same 4 solvers
  - Links for other solvers available at 60%-90% off commercial price
Operating System

- Windows (x86, x86_64)
- Linux (x86, x86_64)
- Solaris
- Mac PowerPC
- HP9000
- DEC Alpha
- AIX
- IRIX
Structure of a GAMS Model

- Sets
- Parameters, Tables, Scalars
- Variables
- Equations
- Inputs are declared then assigned types, values or members
Structure of a GAMS Model

- Model, Solve, Display
- To define the model, solve it and display results
- Other optional statements available for data checks or custom reports
Our ‘Dofasco’ example

Variables

Z
XB
XC;
Positive Variable XB;
Positive Variable XC;

Equations

\[
\begin{align*}
\text{max} & \quad 25x_B + 30x_C \\
\text{s.t.} & \quad (1/200)x_B + (1/140)x_C \leq 40 \\
& \quad 0 \leq x_B \leq 6,000 \\
& \quad 0 \leq x_C \leq 4,000
\end{align*}
\]
Our ‘Dofasco’ Example

\begin{align*}
\text{obj} \quad & Z = 25 \times XB + 30 \times XC; \\
\text{e1} \quad & \frac{1}{200} \times XB + \frac{1}{140} \times XC \leq 40; \\
\text{e2} \quad & XB \leq 6000; \quad \text{max} \; 25x_B + 30x_C \\
\text{e3} \quad & XC \leq 4000; \quad \text{s.t.} \; (\frac{1}{200})x_B + (\frac{1}{140})x_C \leq 40 \\
\text{Model} \quad & \text{toy /all/;} \\
\text{Solve} \quad & \text{toy using lp maximizing } Z; \\
\text{Display} \quad & Z.l, Z.m;
\end{align*}
A Complete Example

- Minimize shipping costs from 2 plants to 3 markets subject to supply and demand

i = plants
j = markets

\( a_i = \text{supply at plant } i \)
\( b_j = \text{demand at market } j \)
\( d_{i,j} = \text{distance between plant } i \text{ and market } j \)
\( C_{i,j} = \text{shipping cost per unit per unit distance} \)
A Complete Example

<table>
<thead>
<tr>
<th></th>
<th>Markets</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plants</td>
<td>NY</td>
<td>Chicago</td>
</tr>
<tr>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
</tr>
<tr>
<td>San Diego</td>
<td>2.5</td>
<td>1.8</td>
</tr>
<tr>
<td>Demand</td>
<td>325</td>
<td>300</td>
</tr>
</tbody>
</table>
A Complete Example

\( F = \$ \) per case per thousand miles

Decision variables:

\( x_{i,j} = \) amount of commodity to ship from plant \( i \) to market \( j \) (cases),
where \( x_{i,j} \geq 0 \) for all \( i, j \)

Constraints:

Supply limit at plant \( i \) (cases): \( \sum_j x_{i,j} \leq a_i \) for all \( i \)
Demand at market \( j \) (cases): \( \sum_i x_{i,j} \geq b_j \) for all \( j \)

Objective function:

Minimize \( \sum_i \sum_j c_{i,j} x_{i,j} \) (thousands of dollars)
GAMS Model

Sets
  i  canning plants / Seattle, San-Diego /
  j  markets / New-York, Chicago, Topeka / ;

Parameters
  a(i)  capacity of plant i in cases
        / Seattle 350
        San-Diego 600 /
  b(j)  demand at market j in cases
        / New-York 325
        Chicago 300
        Topeka 275 / ;
GAMS Model

Table d(i,j) distance in 1000s of miles

<table>
<thead>
<tr>
<th></th>
<th>New-York</th>
<th>Chicago</th>
<th>Topeka</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
</tr>
<tr>
<td>San-Diego</td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
</tr>
</tbody>
</table>

Scalar f freight in dollars per case per thousand miles /90/ ;

Parameter c(i,j) transport cost in 1000s of dollars per case ;

\[ c(i,j) = \frac{f \times d(i,j)}{1000} ; \]
GAMS Model

Variables
x(i,j) shipment quantities in cases
z total transportation costs in 1000s of dollars ;

Positive variable x ;

Equations
cost define objective function
supply(i) observe supply limit at plant i
demand(j) satisfy demand at market j ;
GAMS Model

cost..  
    z  =e=  sum((i,j), c(i,j)*x(i,j))  ;

supply(i)  ..  sum(j, x(i,j))  =l=  a(i)  ;

demand(j)  ..  sum(i, x(i,j))  =g=  b(j)  ;

Model transport  /all/  ;

solve transport using lp minimizing z  ;

display x.l, x.m  ;
GAMS Survival Remarks

1. Any entity cannot be referenced before it is declared
2. Do not combine declaration and assignments
3. Use a user friendly style
4. Beginners should terminate every statement with ;
5. Use comments – 2 forms ‘*’ or lower case words ‘text’
GAMS Survival Remarks

6. GAMS has no concept of ‘objective function’ – GAMS solves a model with respect to the variable to be optimized

    solve transport using lp minimizing z ;
GAMS References

- GAMS Development Corp. (U.S)  
  http://www.gams.com

- GAMS Software GmbH (Europe)  
  http://www.gams.de

- Richard E. Rosenthal, *A GAMS Tutorial*, Naval Postgraduate School, Monterey, California, USA
Modeling Library: FlopC++

Formulation of Linear Optimization Problems in C++

Doron Pearl
The Open Solver Interface

- Provides common interface to numerous solvers.
  - Cbc, CLP, CPLEX, DyLP, FortMP, GLPK, MOSEK, OSL, SoPlex, SYMPHONY, Volume, XPRESS-MP
  - Eliminates dependence on 3rd party software.
- Available from COIN-OP (CPL license).
OSI - Structure

Your code

OSI C++ API
OsiSolverInterface

Osi\textbf{CPLEX}Interface
Osi\textbf{XPRESS}Interface

Osi\textbf{CPLEX} - Implementation

CPLEX Solver

Osi\textbf{XPRESS} - Implementation

XPRESS Solver
OSI - Platforms

- AIX V4.3 using g++ V2.95.2, and AIX V5.3
- Windows using Microsoft Visual C++ V6 & V7
- Windows using the Cygwin toolchain
- Linux using g++ V2.95.2 and higher
- Solaris (SunOS 5.6 and 5.8) using g++ V2.95.2
OSI – What’s in the interface?

- Creating the LP formulation;
- Directly modifying the formulation by adding rows/columns;
- Modifying the formulation by adding cutting planes provided by CGL;
- Solving the formulation (and resolving after modifications);
- Extracting solution information;
- Invoking the underlying solver's branch-and-bound component.

Modeling Library
Why modeling library?

- Data comes from databases/GUI.
- Further processing/logic that is not available in the solver is in need.
- Need to embed models in your application.
- Compiled code – faster than scripts.
About FlopC++

- Open source algebraic modeling language implemented as a C++ class library.
- Declarative optimization modeling (similar to GAMS, AMPL and AIMMS), within a C++ program.
- Supports: LP & MIP
- Solvers: same as OIS
- License: CPL
Dofasco Example

```cpp
MP_model m1;

MP_variable Xb, Xc;
Xb.integer();
Xc.integer();

MP_constraint MaxTonsXb = Xb <= 6000;
MP_constraint MaxTonsXc = Xc <= 4000;
MP_constraint MaxHours = (1/200)*Xb + (1/140)*Xc;

MP_expression F = 25*Xb + 30*Xc;
m1.maximize(F);
```
A Complete Example

- Minimize shipping costs from 2 plants to 3 markets subject to supply and demand

\[ i = \text{plants} \]
\[ j = \text{markets} \]
\[ a_i = \text{supply at plant } i \]
\[ b_j = \text{demand at market } j \]
\[ d_{i,j} = \text{distance between plant } i \text{ and market } j \]
\[ C_{i,j} = \text{shipping cost per unit per unit distance} \]
# A Complete Example

<table>
<thead>
<tr>
<th>Distances</th>
<th>Markets</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plants</td>
<td>NY</td>
<td>Chicago</td>
</tr>
<tr>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
</tr>
<tr>
<td>San Diego</td>
<td>2.5</td>
<td>1.8</td>
</tr>
<tr>
<td>Demand</td>
<td>325</td>
<td>300</td>
</tr>
</tbody>
</table>
A Complete Example

\( F = \$ \) per case per thousand miles

Decision variables:

\[ x_{i,j} = \text{amount of commodity to ship from plant } i \text{ to market } j \text{ (cases),} \]
\[ \text{where } x_{i,j} \geq 0 \text{ for all } i, j \]

Constraints:

Supply limit at plant \( i \) (cases): \( \sum_j x_{i,j} \leq a_i \) for all \( i \)
Demand at market \( j \) (cases): \( \sum_i x_{i,j} \geq b_j \) for all \( j \)

Objective function:

Minimize \( \sum_i \sum_j c_{i,j} x_{i,j} \) (thousands of dollars)
Formulation

- See handouts...
Other Modeling Libraries

- MP’ (Søren S. Nielsen) C++
- Planner (ILOG) C++
- Concert Technology (ILOG) C++
- LP Toolkit (Euro-decision) C++
- XBSL (Dash) C
- AMMO (Drayton Analytics) COM
FlopC++/OSI - drawbacks

- Requires programming skills.
- OSI’s interface is huge (in order to comply with all solver’s features).
- Not a standard (not used that widely).
FlopC++ References

- FlopC++ Homepage
  [https://projects.coin-or.org/FlopC++](https://projects.coin-or.org/FlopC++)

- OSI Homepage
  [https://projects.coin-or.org/Osi](https://projects.coin-or.org/Osi)
Outline

- Overview
- AIMMS Structure
  - GUI
  - Model
  - Solvers
- Sample Problem
- AIMMS Implementation
About AIMMS

- **Advanced Integrated Multidimensional Modeling Software**
- Paragon Decision Technology B.V. was founded in 1989 by Johannes Bisschop in Haarlem, the Netherlands
- First version of AIMMS released in 1993
- In 2003 Paragon was sold to the Dutch Investment Group WARP and J. Bisschop became a President/Founder

Haarlem, the Netherlands

Kirkland, WA, USA
Graphical User Interface (GUI)

- Standard graphical user interface (GUI) tools:
  - E.g. tables, 2D- and 3D-charts, buttons, sliders

- Advanced GUI control (productivity) tools:
  - Page Manager
    - Guides a user through the application by navigation controls
  - Template Manager
    - Uniform look and feel: headers, buttons, footers are in hierarchically organized templates which can be inherited by multiple pages
  - Menu Builder
    - Customized menus and toolbars can be added to pages and templates
AIMMS Structure
Modeling Language

- Contains a large set of mathematical, statistical and financial functions
- Multidimensional – index notation
- Optimization modeling
  - Capable of specifying and solving linear, nonlinear or mixed-integer problems, easy transfer of the formulated problem to CPLEX, XPRESS or CONOPT
- Units of measurement
  - Supports SI and custom units, scale factors. Allows users to work on the same problem with different units
- Second order derivative evaluation
  - Evaluates Hessian to use by solvers to solve NLP more efficiently than with Jacobian only
Modeling Language II

- External procedures and functions
  - Can be called within any AIMMS procedure, e.g. from a constraint to provide derivative information

- Data management
  - Database connectivity – link to Oracle, SQL, Sybase, MS Access, etc. using data connectivity interfaces
  - Data exchange with Excel
  - Supports reading and writing of XML files

- Web services
  - AIMMS supports agent technology which can be used for massive parallelization of very large problems (good timing)

- AIMMS optimization components
  - Integration of AIMMS model as an optimization component within other server or application: C/C++ API and a COM object
Modeling Productivity Tools

- **Model explorer**
  - Graphical representation of all the identifiers, procedures and functions in a model
  - Double-click on an identifier in the model tree to open attribute window

- **Identifier selector**
  - Select and simultaneously view the attributes of groups of identifiers that share functional aspects in the model

- **Data manager**
  - A user can develop a collection of cases to use with a model – different data with the same model

- **Data management setup**
  - Create custom types of cases and data sets
AIMMS Structure
Solvers

- CPLEX – LP, MIP, also QP, QCP, MIQP, MIQCP
- XPRESS – LP, MIP, also QP, MIQP
- XA – default solver for LP, MIP, QP in AIMMS
- CONOPT – default solver for NLP/QP
- PATH – MCP
- SNOPT – NLP
- MINOS – NLP
- MOSEK – LP, MIP
- BARON – global optimization
- LGO – global optimization
- AOA – default solver
- OSI
Open Solver Interface (OSI)

- All AIMMS solvers linked to AIMMS using AIMMS OSI
- You can link your own solvers to AIMMS by using OSI
- A collection of C++ interfaces allows communication between AIMMS and a solver during problem solving
Diagnostic Tools

- **Debugger**
  - Set breakpoints, step through the execution, view variables

- **Data pages**
  - Default view of the current contents of an identifier selected in Model Explorer

- **Profiler – resolve computational time issues**
  - Shows number of calls to procedures and function, number of evaluations of individual statements, sets, parameters, variables, and constraints

- **Identifier cardinalities**
  - Locate missing or incorrectly specified domain conditions by displaying for each model identifier the theoretical maximum cardinality as well as actual number of values stored

- **Math program inspector**
  - Analyze the causes of infeasibility, unrealistic results, etc.
Prices

- US $1,972 and up (XA, CONOPT & AOA are included)
- FREE evaluation can be ordered
Sample Problem

- Dofasco needs to schedule its two products on the rolling mill
- If 40 hours of production time are available, what quantity of each product should be made in order to maximize profits?

<table>
<thead>
<tr>
<th>Product</th>
<th>Tons per hour</th>
<th>Profit per ton</th>
<th>Max tons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bands</td>
<td>200</td>
<td>$25</td>
<td>6,000</td>
</tr>
<tr>
<td>Coils</td>
<td>140</td>
<td>$30</td>
<td>4,000</td>
</tr>
</tbody>
</table>
Formulation

\[
\begin{align*}
\text{max} & \quad 25x_B + 30x_C \\
\text{s.t.} & \quad (1/200)x_B + (1/140)x_C \leq 40 \\
& \quad 0 \leq x_B \leq 6,000 \\
& \quad 0 \leq x_C \leq 4,000
\end{align*}
\]
AIMMS Model

- **Set:**
  - products (bands, coils)

- **Parameters:**
  - profit per ton (25, 30)
  - productivity (1/200, 1/140)
  - hours (40)
  - product upper limit (6000, 4000)

- **Variables:**
  - product amount \( (x_B, x_C) \)
  - total profit \( (25x_B + 30x_C) \)

- **Constraint:**
  - working time \( (1/200x_B + 1/140x_C \leq 40) \)

**Mathematical Program:**

Maximize Total Profit

s.t. Constraints

\[
\begin{align*}
\text{Maximize} & \quad 25x_B + 30x_C \\
\text{s.t.} & \quad (1/200)x_B + (1/140)x_C \leq 40 \\
& \quad 0 \leq x_B \leq 6,000 \\
& \quad 0 \leq x_C \leq 4,000 \\
\end{align*}
\]
Demos and References

- See demos!
- http://www.aimms.com
- AIMMS, A One-Hour Tutorial for Beginners, Paragon Decision Technology, 2006
NEOS

- **Network Enabled Optimization System** is the project of the Optimization Technology Center founded by Argonne National Laboratory and Northwestern University, Illinois, USA


- **NEOS Guide**, a guide to optimization models, algorithms and software

- **NEOS Server**, a facility for solving optimization problems remotely over the Internet
Input Formats

- **Low-level formats** explicitly describe every constraint and objective (e.g. MPS)
- **C or Fortran programs** represent constraints and objectives by subroutines that evaluate the function values
- **High-level** algebraic formulations describe optimization problems in symbolic formats using modeling languages
Using NEOS

- Go to the list of solvers: http://neos.mcs.anl.gov/neos/solvers/
- Choose an appropriate solver for your problem
- Formulate your optimization problem in the accepted format
- Submit your file(s) through the form on a website
- Receive results