



Keep the Model Hot

A Scenario Solver for GAMS

Michael R. Bussieck
GAMS Development Corp.

MBussieck@gams.com
<http://www.gams.com>





GAMS at a Glance

The screenshot displays the GAMS software interface with several components:

- Code Editor:** Contains GAMS code for creating an example GDX file for charting. The code includes comments and parameters for data generation.
- Data Table:** A table listing model elements with columns for Entry, Symbol, Type, Dim, and Nr Elem. The selected entry is '12 StockData'.
- Line Chart:** Titled 'StockData', showing the stock prices of IBM, DELL, HP, and SUN over time. The y-axis ranges from 102 to 104, and the x-axis shows years from 1998 to 2005.
- 3D Surface Plot:** Titled 'Surface', showing a 3D surface plot with a prominent peak. The x-axis is labeled with 's2' through 's49', and the y-axis ranges from -0.2 to 0.6.
- Log Window:** Shows the execution log for 'chartdat.gms', including start and stop times and file paths.

General Algebraic Modeling System:
 Algebraic Modeling Language,
 Integrated Solver, Model
 Libraries, Connectivity- &
 Productivity Tools

Design Principles:

- **Balanced mix of declarative and procedural elements**
- Open architecture and interfaces to other systems
- Different layers with separation of:
 - model and data
 - model and solution methods
 - model and operating system
 - model and interface

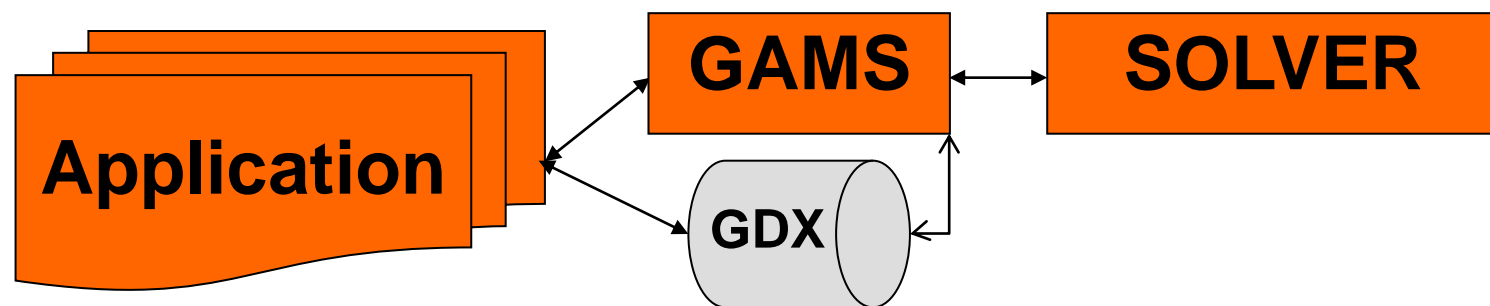


Outline

- Prelude
- GAMS Execution System
- Scenario Solver
- GMO + Python & Co



Calling GAMS from your Application



Creating Input for GAMS Model

→ Data handling using **GDX** API

Callout to GAMS

→ GAMS option settings using **Option** API

→ Starting GAMS using **GAMS** API

Reading Solution from GAMS Model

→ Data handling using **GDX** API



Automated Generation of APIs

‘The GAMS Wrapper’

- API is defined using the GAMS language
- A tool written in GAMS is used to regenerate APIs for all languages
- Executed on request and nightly

```

* Properties reading/writing data that originally came from control vectors
gmoModelType .int. (r,w).getModelType ,w.modeltype )
gmoM .int. r .GetRows
gmoN .int. r .GetCols
gmoScaleOpt .int. (r,w).ScaleOpt
gmoSense .int. (r,direction ,w.setObjSense )
gmoObjVar .int. (r,w).GetObjVar ,w.SetObjVar )
gmoOptFile .int. (r,w).OptFile
gmoPriorOpt .int. (r,w).prioropt
gmoNLConst .int. (r,w).nlconst
gmoNZ .int. (r ,.GetNonZeros,w.NZ
gmoNLNZ .int. r .GetNLNonZeros
gmoNLM .int. r .GetNLRows
gmoNLN .int. r .GetNLCols
gmoObjRow .int. r .GetObjRow
gmoDictionary .int. (r,w).Dictionary
gmoHaveBasis .int. (r,w).havebasis

gmoNameOptFile .oSS.( r .NameOptFile ,w.SetNameOptFile )
gmoNameSolFile .oSS.( r .NameSolFile ,w.SetNameSolFile )
gmoNameZLib .oSS.( r .NameZLib ,w.SetNameZLib )
gmoNameMatFile .oSS.( r .NameMatFile ,w.SetNameMatFile )
gmoNameDict .oSS.( r .NameDict ,w.SetNameDict )
gmoNameParams .oSS.( r .NameParams ,w.SetNameParams )
gmoNameInput .oSS. r .NameInput

Model Type
Number of equations
Number of variables
Scaling Flag
Direction of optimisation
Objective variable index
Optfile Number
Priority Flag
length of NL constant pool
Number of non zeros in constraints
Number of nonlinear non zeros in constral
Number of nonlinear rows
Number of nonlinear columns
Objective row index
Dictionary file written
Do we have basis

Option file name
Solution file name
External Function Library Name
Matrix file name
Dictionary file name
Params file name
Input file name

set if(en,tp,es,ta) function and procedures /
gmoLoadDataLegacy .(0,result.int,1,msg.oSS)
gmoInitData .(0,result.int,1,rows.int, 2,cols.int)
gmoCompleteData .(0,result.int,1,instname.CSS)
gmoQMaker .(0,result.int,1,density.D)
gmoSetObjQ .(0,result.int,1,colIdx.PLIA,2,rowIdx.PLIA,3,coef.PDA)
    
```

- A change in the definition of the API immediately makes it into all language interfaces
- No manual and therefore error-prone efforts required



Automated Generation of APIs

‘The GAMS Wrapper’

- Automated nightly testing
- API version checks
- Reusable for multiple GAMS component libraries
 - GAMS
 - GDX
 - Option
 - ...

```

* Properties reading/writing data that originally came from control vectors
gmsModelType .int. ( r ,getModelType ,w.modeltype ) Model Type
gmsM .int. r .GetRows ) Number of equations
gmsN .int. r .GetCols ) Number of variables
gmsScaleOpt .int. ( r,w ).ScaleOpt ) Scaling Flag
gmsSense .int. ( r,direction ,w.setObjSense ) Direction of optimisation
gmsObjVar .int. ( r ,GetObjVar ,w.SetObjVar ) Objective variable index
gmsOptFile .int. ( r,w ).OptFile ) Optfile Number
gmsPriorOpt .int. ( r,w ).prioropt ) Priority Flag
gmsNLConst .int. ( r,w ).nlconst ) length of NL constant pool
gmsNZ .int. ( r ,GetNonZeros,w.NZ ) Number of non zeros in constraints
gmsNLNZ .int. r .GetNLNonZeros ) Number of nonlinear non zeros in constral
gmsNLM .int. r .GetNLRows ) Number of nonlinear rows
gmsNLN .int. r .GetNLCols ) Number of nonlinear columns
gmsObjRow .int. r .GetObjRow ) Objective row index
gmsDictionary .int. ( r,w ).Dictionary ) Dictionary file written
gmsHaveBasis .int. ( r,w ).havebasis ) Do we have basis

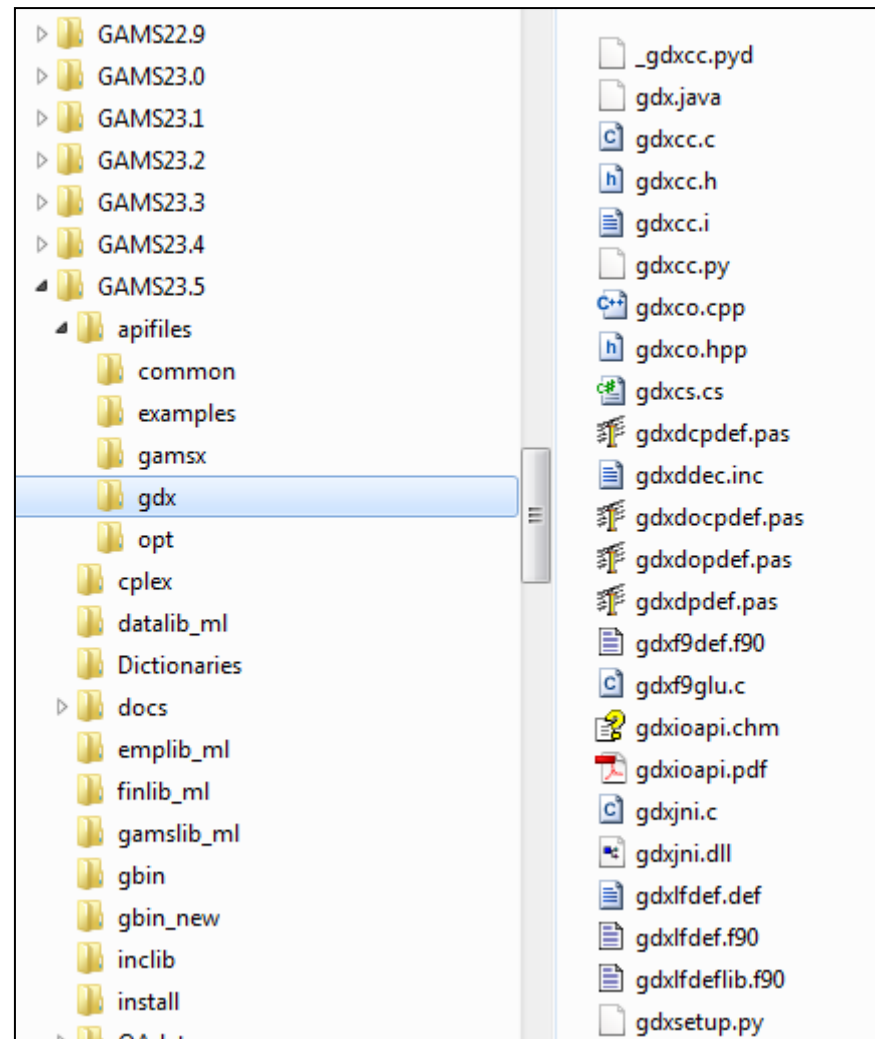
gmsNameOptFile .oSS. ( r .NameOptFile ,w.SetNameOptFile ) Option file name
gmsNameSolFile .oSS. ( r .NameSolFile ,w.SetNameSolFile ) Solution file name
gmsNameXLib .oSS. ( r .NameDll ,w.SetNameDLL ) External Function Library Name
gmsNameMatFile .oSS. ( r .NameMatFile ,w.SetNameMatFile ) Matrix file name
gmsNameDict .oSS. ( r .NameDict ,w.SetNameDict ) Dictionary file name
gmsNameParams .oSS. ( r .NameParams ,w.SetNameParams ) Params file name
gmsNameInput .oSS. r .NameInput ) Input file name

/
set if(en,tp,es,ta) function and procedures /
gmsLoadDataLegacy .(0,result.int,1,msg.oSS) Read GMS instance - Legacy Mode
gmsInitData .(0,result.int,1,rows.int, 2,cols.int) Initialises GMS data
gmsCompleteData .(0,result.int,1,instname.CSS) Complete GMS data instance needs lots of
gmsQMaker .(0,result.int,1,density.D) Create QP Info
gmsSetObjQ .(0,result.int,1,colIdx.PLIA,2,rowIdx.PLIA,3,coef.PDA) Get Q matrix for objective
  
```



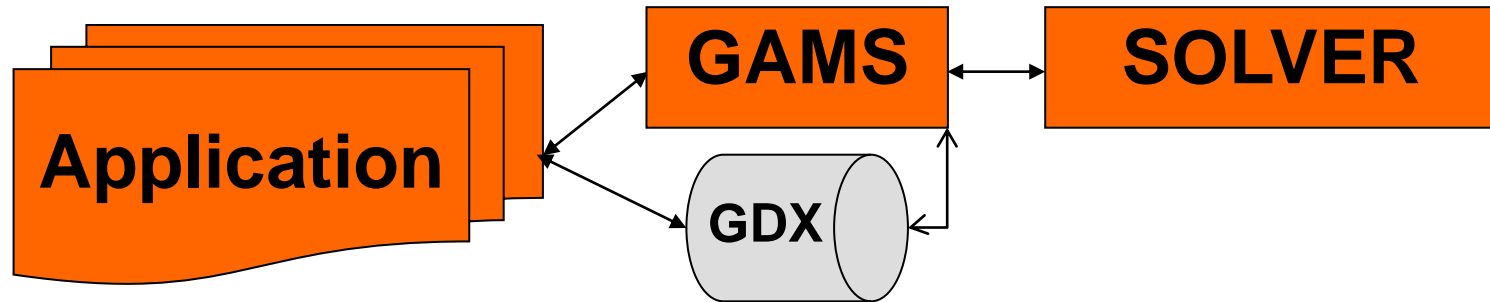
Distributed GAMS APIs

- Component Libraries
 - GAMS
 - GDX
 - Option
- Supported languages
 - C, C++, C#
 - Delphi
 - Fortran
 - Java
 - VBA, VB.Net
 - **Python**
- Examples/Documentation





GAMS Execution System



- Mix of declarative and procedural language elements
 - Multiple models
 - Loops/if-then-else
 - Well suited for decomposition approaches (SP)
 - Many examples in the GAMS Model library
 - Cutting stock
 - TSP
 - ...



GAMS Scenario Solver

```
Loop (s,  
      d(i,j) = dd(s,i,j);  
      f = ff(s);  
      solve mymodel min z using lp;  
      rep(s) = mymodel.objval;  
);
```

Setting	Solve time (secs)
SolveLink=0 (<i>default</i>)	40.297
SolveLink=%SolveLink.LoadLibrary%	03.625



GAMS Scenario Solver

```
cost.. z=e=sum((i,j), f*d(i,j)/1000*x(i,j));  
set dict / s.scenario.''  
      d.param      .dd  
      f.param      .ff  
      x.level      .xx /  
solve mymodel min z using lp scenario dict;
```

- Update model data instead of matrix coefficients/rhs
- Hot start (keep the model hot inside the solver and use solver's best update mechanism)
- Save model generation and solver setup time
- Model rim unchanged from scenario to scenario
- Apriori knowledge of all scenario data



Scenario Solver – Cont.

- Dynamic model – rolling horizon



- Example:
 - Combined Heat and Power Planning with Heat Storage. All data known apriori but heat storage level.
 - Can't use GAMS Scenario Solver
 - Implement Scenario Solver in Python
 - Identify some parameters as “modifiable” parameters
 - Implement rolling horizon in Python



GMO – GAMS Modeling Object

- Powerful & convenient API – a few calls do the job
- In-core communication between GAMS and the solver, making potentially large model scratch files unnecessary
- Support shared-library implementation of solver links
- Support multiple models
- Support meta-solvers (e.g. DICOPT, SBB, Examiner)
- Implement once, run everywhere (multiple platforms & multiple languages)
- Comprehensive – one-stop shop for all linking needs




GMO Talk by Steve Dirkse, TC40, 13:30-15:00



Python & Co with GMO

- Populated GMO object (e.g. by GAMS)
- GMO API to allow modification and alteration of bounds, rhs, “modifiable” parameters (NL expression evaluation)
- GMO/GEV (GAMS Environment Object) based solver links
- Runtime system (Python, Java, ...)

 Alternative way to implement decomposition, and other algorithmic ideas based on MP models.

- Examples:
 - TSP (subtour elimination constraint generation)
 - Markowitz portfolio optimization



Summary

- Automated API generation for various GAMS components (GAMS, GDX, OPT) in various languages including Python
- GAMS Scenario Solver approach for solving very similar models.
- GMO + Python & Co represent an alternative to GAMS execution system.
- Outlook: Get some of the improvements of the GMO + Python & Co approach back into GAMS
- (Python) API for GMO not published yet (available on request). Still unclear where this experiment will lead us.



Contacting GAMS

Europe

GAMS Software GmbH
Eupener Str. 135-137
50933 Cologne
Germany

Phone: +49 221 949 9170

Fax: +49 221 949 9171

<http://www.gams.de>

info@gams.de

USA

GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007
USA

Phone: +1 202 342 0180

Fax: +1 202 342 0181

<http://www.gams.com>

sales@gams.com

support@gams.com