Pre-Conference Workshops

Michael Bussieck
Steve Dirkse
Fred Fiand
Lutz Westermann
Outline

Part I: An Introduction to GAMS
Part II: Stochastic programming in GAMS
Part III: The GAMS Object-Oriented API's
Part IV: Code embedding in GAMS
An Introduction to GAMS: Agenda

- GAMS at a Glance
- Foundation and Design Principles
- GAMS – A simple Example
- Wrap-Up
Company


- Went commercial in 1987

- GAMS Development Corp. (USA), GAMS Software GmbH (Germany)

- 2016: New management team

- Software Tool Provider
GAMS at a Glance

➢ Pioneered Algebraic Modeling Languages

➢ Robust, scalable state-of-the-art algebraic modeling technology for complex, large-scale optimization

➢ Open architecture and uniform interface to all major commercial and academic solvers (30+ integrated)

➢ Evolution through more than 25 years of R&D and user feedback, maturity through experience and rigorous testing
# GAMS Users and Application Areas

- **13,500+ licenses**
- **Users:** 50% academic, 50% commercial or governmental
- **Used in more than 120 countries**

## Broad Range of Application Areas

<table>
<thead>
<tr>
<th>Agricultural Economics</th>
<th>Applied General Equilibrium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chemical Engineering</td>
<td>Economic Development</td>
</tr>
<tr>
<td>Econometrics</td>
<td>Energy</td>
</tr>
<tr>
<td>Environmental Economics</td>
<td>Engineering</td>
</tr>
<tr>
<td>Finance</td>
<td>Forestry</td>
</tr>
<tr>
<td>International Trade</td>
<td>Logistics</td>
</tr>
<tr>
<td>Macro Economics</td>
<td>Military</td>
</tr>
<tr>
<td>Management Science/OR</td>
<td>Mathematics</td>
</tr>
<tr>
<td>Micro Economics</td>
<td>Physics</td>
</tr>
</tbody>
</table>
Strong Development Environment

- Project management
- Editor / Syntax coloring / Spell checks
- Model Debugging / Profiling
- Solver selection / Option selection
- Data viewer
- GAMS Processes Control
Uniform System Documentation

GAMS Documentation Center

The GAMS Documentation Center provides you with the technical information on getting started, using and maintaining our GAMS (General Algebraic Modeling System) products.

- **Release Notes** - What's new in GAMS and all changes made in the last releases.
- **Installation Notes** - GAMS Installation guides on different platforms.
- **Licensing** - GAMS Licensing
- **Tutorials and Examples** - Step-by-step guides included.

Model Libraries

The Model Libraries contains a large number of GAMS Models including:

- GAMS Model Library - GAMS models representing interesting problems such as shipment by firms, investment planning, cropping patterns in agriculture, macroeconomics stabilization, applied general equilibrium, integer programming, and many more.
- GAMS Test Library - GAMS models developed for testing and distributing solvers with the GAMS system.
- GAMS Data Library - GAMS models demonstrating various uses such as spreadsheets and database interface.
- GAMS EMP Library - GAMS Extended Mathematical Programming System.
- GAMS API Library - GAMS Models used as scripts to compile languages interfacing to GAMS.
- FIN Library - GAMS practical financial optimization models designed for financial engineers.
- NOA Library - GAMS nonlinear optimization applications model.

This documentation guides GAMS User through several topics in GAMS system.

- **GAMS Language** - This part introduces the components of the GAMS language in an ordered way, interspersed with detailed examples that are often drawn from the model library. All models from the model library are enclosed in square parentheses (for example, [TRANSPORT]).
  - **Introduction** - an introductory to GAMS User's Guide.
  - **GAMS Programs** - The structure of the GAMS language and its components.
  - **Set Definition** - The declaration and initialization of sets, subsets, and domain checking.
  - **Dynamic Sets** - The membership assignment, the usage of dollar controls, and set operations.
  - **Sets as Sequences: Ordered Sets** - Special features used to deal with a set as if it were a sequence.
  - **Data Manipulations with Parameters** - The declaration and assignment of GAMS parameters.
  - **Data Entry: Parameters, Scalars and Tables** - Three basic forms of GAMS data types: Parameters, Scalars and Tables.
**Simple Integration** of GAMS Models

**Object Oriented API’s**

- Use GAMS for modeling and optimization tasks
- Connects GAMS to other environments
  - Programming languages (.Net, C++, Java, Python)
  - Applications (through Smart Links)
  - (New) Embedded Code (Python)
- Communication through Memory or Files
General Algebraic Modeling System

Free Model Libraries

More than 1400 models!
Where to Find Help?

- Documentation Center: https://www.gams.com/latest/docs/
- Mailing Lists, Newsletters, and Forum:
  - https://www.gams.com/community/newsletters-mailing-list/
  - //forum.gamsworld.org/
- YouTube Channel: https://www.youtube.com/user/GAMSLessons
- GAMS Support: support@gams.com
Foundation of GAMS

- Powerful algebraic modeling language
- Open architecture with interfaces to other systems
- Independent layers
**Powerful Declarative Language**

Similar to mathematical notation

Easy to learn - few basic language elements: sets, parameters, variables, equations, models

Model is executable (algebraic) description of the problem

Lots of code optimization under the hood
Mix of Declarative and Imperative Elements

Control Flow Statements (e.g. loops, for, if,...), macros and functions

Advantages:
- Build complex problem algorithms within GAMS
- Simplified interaction with other systems:
  - Data exchange
  - GAMS process control
Foundation of GAMS

- Powerful algebraic modeling language
- Open architecture with interfaces to other systems
- Independent layers
Open Architecture

Designed to interact with other Systems

- Solver Interfaces: Proprietary or Open Source (COIN)
- Data Exchange and GAMS Control
- ASCII or Binary (GDX)
- OO-API (.Net, C++, Java, Python, file or memory)
- Smart Links to other applications (e.g. MS Excel, Databases, Matlab, R,...)
- Code Embedding
Foundation of GAMS

- Powerful algebraic modeling language
- Open architecture with interfaces to other systems
- Independent layers
Separation of **Model and Platform**

**Supported Platforms**

- **Move models between platforms with ease!**

---

Model

- Platform
- Solver
- Data
- Interface
Separation of Model and Solver

One environment for a wide range of solvers

- All major commercial LP/MIP solver
- Open Source Solver (COIN)
- Also solver for NLP, MINLP, global, and stochastic optimization

More than 30 Solvers integrated!
Separation of **Model and Solver**

**Uniform interface to all major solvers**
- Switching between solvers with one statement
- Unified Documentation
- Licensing (GAMS as a „license broker“)

**Av. number of commercial solvers per license**
- Academic clients: 2.9
- Commercial: 2.2
Separation of Model and Data

Sets
i canning plants
j markets ;

Parameters
a(i) capacity of plant i in cases
b(j) demand at market j in cases
d(i,j) distance in thousands of miles
c(i,j) transport cost in thousands of dollars per case :
Scalar f ;

Variables
x(i,j) shipment quantities in cases
z total transportation costs in thousands 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 ;
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/ ;

- Declarative Modeling
- Sparse Data Structures
- Various ways to exchange data
- ASCII
- Binary
Separation of **Model and User Interface**

No preference for a particular user interface

- Open architecture and interfaces to other systems
- OO-API’s for seamless integration
- Smart Links
- Mode of Operation
- Interactive or Embedded / Batch
- Local or Remote
Application – Cloud Computing

xyz Energy Company

Scenario Analysis in the Cloud
➢ Solve 1,000+ scenarios (MIPs, one hour) every week overnight
➢ Issues:
   ➢ Costs (Licensing)
   ➢ Automation / Security
Application – Cloud Computing

xyz – Energy Company

Implementation:

➢ Amazon Cloud: 1,000+ parallel machines (instances), Python, GAMS + OO Python API
➢ Automated setup, including
  ➢ Starting instances
  ➢ Prepare / Submit / Run GAMS jobs
  ➢ Collect results
  ➢ Stop instances
Application – Cloud Computing

Commercial Aspects

“Hardware” Amazon Cloud (1,000 instances):
Hardware Costs / run: $70!
(1,000 instances/run * $0.07 instance / hour)

Software Licensing:
- Gurobi and IBM offer per-usage license
- Client with strong preference for annual license fee, not a per-usage license
Application – Cloud Computing

45 Provided Model Instances

- Statistics:
  - 163,608 – 1,959,550 rows
  - 84,930 – 983,587 var. (32,240-258,796 dis.)
  - 447,537 – 6,068,729 NZ

- Tests with CPLEX, SCIP, and CBC
- 60 minutes, gap max. 1%
- Manual option tuning for SCIP
  (thanks to Gerald Gamrath & Ambros Gleisner)
Application – Cloud Computing

Results

➢ CPLEX: All instances solved to optimality
➢ SCIP:
  ➢ Could solve all 45 instances
  ➢ But: After 60 min. 2 instances with gap > 20%
➢ CBC:
  ➢ Did also well
  ➢ But: After 60 min. no solution for some instances (< 10%)
Application – Cloud Computing

Proposed Strategy

➢ Run all instances simultaneously with SCIP and CBC
   ➔ „hardware“ costs: $0,07 per instance hour

➢ After 60 minutes take the best solution

➢ If necessary solve „difficult“ model instances with CPLEX (outside the cloud)
Agenda

- GAMS at a Glance
- Foundation and Design Principle
- GAMS – A simple Example
- Wrap-Up
A Simple **Transportation Problem**

What does this example show?

- It gives a first glimpse of how a problem can be formulated in GAMS
- It shows some basics of data exchange with GAMS
- It shows how easy it is to change model type and, consequently, solver technology
Minimize transportation cost subject to demand satisfaction at markets and supply constraints.
Model types in this example

**LP**
Determine minimum transportation cost.
Result: city to city shipment volumes.

**MIP**
Allows discrete decisions,
e.g. if we ship, then we ship at least 100 cases.

**MINLP**
Allows non-linearity,
e.g. a smooth decrease in unit cost when shipping volumes grows.
### Mathematical Model Formulation

**Indices:**
- \( i \) (Canning plants)
- \( j \) (Markets)

**Decision variables:**
- \( x_{ij} \) (Number of cases to ship)

**Data:**
- \( c_{ij} \) (Transport cost per case)
- \( a_i \) (Capacity in cases)
- \( b_j \) (Demand in cases)

\[
\min \sum_i \sum_j c_{ij} \cdot x_{ij} \quad \text{(Minimize total transportation cost)}
\]

subject to
- \( \sum_j x_{ij} \leq a_i \quad \forall i \) (Shipments from each plant \( \leq \) supply capacity)
- \( \sum_i x_{ij} \geq b_j \quad \forall j \) (Shipments to each market \( \geq \) demand)

- \( x_{ij} \geq 0 \quad \forall i, j \) (Do not ship from market to plant)

- \( i, j \in \mathbb{N} \)
GAMS Algebra (declarative Model)

Sets
  i  canning plants
  j  markets ;

Parameters
  a(i)  capacity of plant i in cases
  b(j)  demand at market j in cases
  d(i,j)  distance in thousands of miles
  c(i,j)  transport cost in thousands of dollars per case ;
Scalar  f  freight in dollars per case per thousand miles;

Variables
  x(i,j)  shipment quantities in cases
  z  total transportation costs in thousands 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 ;
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/ ;

Model is executable description of the problem
Model types in this example

- **LP**
  - Determine minimum transportation cost.
  - Result: city to city shipment volumes.

- **MIP**
  - Allows discrete decisions, e.g. if we ship, then we ship at least 100 cases.

- **MINLP**
  - Allows non-linearity, e.g. a smooth decrease in unit cost when shipping volumes grows.
MIP Model: Minimum Shipment of 100 cases

- Shipment volume: $x$ (continuous variable)
- Discrete decision: $\text{ship}$ (binary variable)

add constraints:

\[ x_{i,j} \geq 100 \cdot \text{ship}_{i,j} \quad \forall i, j \quad \text{(if ship=1, then ship at least 100)} \]
\[ x_{i,j} \leq \text{bigM} \cdot \text{ship}_{i,j} \quad \forall i, j \quad \text{(if ship=0, then do not ship at all)} \]

\[ \text{ship}_{i,j} \in \{0,1\} \]
MIP Model: GAMS Syntax

```
* MIP
scalar minS minimum shipment / 100 /
    bigM big M;
bigM = min(smax(i,a(i)), smax(j,b(j)));

binary variable ship(i,j) '1 if we ship from i to j, otherwise 0';

equation minship(i,j) minimum shipment
    maxship(i,j) maximum shipment;

minship(i,j).. x(i,j) =g= minS * ship(i,j);
maxship(i,j).. x(i,j) =l= bigM * ship(i,j);

Model transportMIP / transportLP, minship, maxship / ;
option optcr = 0;

Solve transportMIP using MIP minimizing z ;

rep(i,j,'MIP') = x.l(i,j);
display rep;
```
MIP Model: Results

**** REPORT SUMMARY :
0 NONOPT
0 INFEASIBLE
0 UNBOUNDED

GAMS 24.8.5 r61358 Released May 10, 2017 WEX-WEI x86 64bit/MS Windows 08/07

General Algebraic Modeling System

Execution

--- 66 PARAMETER rep report parameter

LP MIP

seattle .new-york 50.000
seattle .chicago 300.000 300.000
san-diego .new-york 275.000 325.000
san-diego .topeka 275.000 275.000

EXECUTION TIME = 0.000 SECONDS

USER: Franz Nelissen
GAMS Software GmbH
Model types in this example

- **LP**
  - Determine minimum transportation cost.
  - Result: city to city shipment volumes.

- **MIP**
  - Allows discrete decisions,
    - e.g. if we ship, then we ship at least 100 cases.

- **MINLP**
  - Allows non-linearity,
    - e.g. a smooth decrease in unit cost when shipping volumes grows
MINLP: Cost Savings

The cost per case decreases with an increasing shipment volume.

Replace:

$$\min \sum_i \sum_j c_{ij} \cdot x_{ij}$$  (Minimize total transportation cost)

With

$$\min \sum_i \sum_j c_{ij} \cdot x_{ij}^{beta}$$  (Minimize total transportation cost)
MINLP Model: GAMS Syntax

* MINLP
Scalar beta / 0.95 /
Equation costnlp define non-linear objective function;
costnlp.. z =e= sum((i,j), c(i,j)*x(i,j)**beta) ;

Model transportMINLP / transportMIP - cost + costnlp /;

Solve transportMINLP using MINLP minimizing z ;

rep(i,j,'MINLP') = x.l(i,j);
display rep;
MINLP Model: Results

```
0 INFEASIBLE
0 UNBOUNDED
0 ERRORS

GAMS 24.8.5  r61358 Released May 10, 2017 WEX-WEI x86 64bit/MS Windows 08/07
General Algebraic Modeling System

Execution

---- 85 PARAMETER rep report parameter

<table>
<thead>
<tr>
<th></th>
<th>LP</th>
<th>MIP</th>
<th>MINLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>seattle</td>
<td>.new-york</td>
<td>50.000</td>
<td>300.000</td>
</tr>
<tr>
<td>seattle</td>
<td>.chicago</td>
<td>300.000</td>
<td>300.000</td>
</tr>
<tr>
<td>san-diego</td>
<td>.new-york</td>
<td>275.000</td>
<td>275.000</td>
</tr>
<tr>
<td>san-diego</td>
<td>.topeka</td>
<td>275.000</td>
<td>275.000</td>
</tr>
</tbody>
</table>

LOWER  LEVEL  UPPER

---- VAR z

- INF  153.675  + INF

LOWER  LEVEL  UPPER

---- VAR z

- INF  115.438  + INF
```
Agenda

GAMS at a Glance

Foundation and Design Principle

GAMS – A simple Example

Wrap-Up
Thank You

Meet us at the GAMS booth!
Outline

Part I: An Introduction to GAMS

Part II: Stochastic programming in GAMS

Part III: The GAMS Object-Oriented API's

Part IV: Code embedding in GAMS
Stochastic Programming - Introduction

Stochastic programs are mathematical programs that involve uncertain data.

Motivation:
Real world problems frequently include some uncertain parameters. Often these uncertain parameters follow a probability distribution that is known or can be estimated.

Goal:
Find some policy that is feasible for all (or almost all) the possible data instances and that maximizes the expectation of some function of the decision variables and the random variables.

Example:
In a two-stage stochastic programming problem with recourse the decision maker has to make a decision now and then minimize the expected costs of the consequences of that decision.
Simple Example: Newsboy (NB) Problem

• Data:
  • A newsboy faces a certain demand for newspapers
    \( d = 63 \)
  • He can buy newspapers for fixed costs per unit
    \( c = 30 \)
  • He can sell newspapers for a fixed price
    \( v = 60 \)
  • For leftovers he has to pay holding costs per unit
    \( h = 10 \)
  • He has to satisfy his customers demand or has to pay a penalty
    \( p = 5 \)

• Decisions:
  • How many newspapers should he buy: \( X \)
  • How many newspapers should he sell: \( S \)

\[
\begin{align*}
X & = 63 \\
S & = 63
\end{align*}
\]

• Derived Outcomes:
  • How many newspapers need to be disposed: \( I \)
  • How many customers are lost: \( L \)

\[
\begin{align*}
I & = 0 \\
L & = 0
\end{align*}
\]
**Simple NB Problem – GAMS Formulation**

**Variable**  
Z Profit;

**Positive Variables**  
X Units bought
I Inventory
L Lost sales
S Units sold;

**Equations**  
Row1, Row2, Profit;

* demand = UnitsSold + LostSales  
Row1.. d =e= S + L;

* Inventory = UnitsBought - UnitsSold  
Row2.. I =e= X - S;

* Profit, to be maximized;  
Profit.. Z =e= v*S - c*X - h*I - p*L;

**Model** nb / all /;

**Solve** nb max z use lp;

>`nbsimple.gms`
NB Problem – Add Uncertainty

• Uncertain demand $d$

• Decisions to make:
  • How much newspaper should he buy “here and now” (without knowing the outcome of the uncertain demand)?
    → First-stage decision
  • How many newspapers are sold?
  • How many customers are lost after the outcome becomes known?
  • How many unsold newspapers go to the inventory?
    → Second-stage or recourse decisions
  • Recourse decisions can be seen as
    • penalties for bad first-stage decisions
    • variables to keep the problem feasible
Stochastic NB Problem – GAMS Extension

Idea:
Use deterministic model formulation plus some annotation to define uncertainty.

```
randvar d discrete 0.7  45
  0.2  40
  0.1  50
```

- Make demand d uncertain
- Define (non-default) stage 2 variables and equations
**Stochastic NB Problem – GAMS Extension**

```gams
file emp / "%emp.info%" /; put emp '* problem %gams.i%'/;
$onput
randvar d discrete 0.7 45
          0.2 40
          0.1 50
stage 2 I L S d
stage 2 Row1 Row2
$offput
putclose emp;
```

Syntax to write an **EMP** info file on the fly, e.g. ` [...]\225a\empinfo.dat`

---

**EMP, what?**

→ **Excursus**
The EMP Framework

EMP stands for Extended Mathematical Programming

EMP Information  Original Model

Translation

Reformulated Model

Viewable

Solve using established Algorithms

Solution

Mapping Solution into original space
Dictionary with output-handling information

- The expected value of the solution can be accessed via the regular .L (and .M) fields
- Additional information can be stored in a parameter by scenario, e.g.:
  - **level:** Levels of variables or equations
  - **randvar:** Realization of a random variable
  - **opt:** Probability of each scenario
- This needs to be stored in a separate dictionary:

```plaintext
Set scen Scenarios / s1*s3 /;
Parameter
  s_d(scen) Demand realization by scenario
  s_x(scen) Units bought by scenario
  s_s(scen) Units sold by scenario
  s_o(scen,*) scenario probability / #scen.prob 0 /;

Set dict / scen .scenario.''
  d .randvar .s_d
  s .level .s_s
  x .level .s_x
  '' .opt .s_o /;

solve nb max z use emp scenario dict;
```
3 parts of a GAMS EMP stochastic model

1. The deterministic core model

2. EMP annotations in EMP info file

3. The dictionary with output-handling information
Extensions to the Simple NB Problem

• Multiple stages:

\[
\text{stage stageNo rv } | \text{ equ } | \text{ var } \{ \text{rv } | \text{ equ } | \text{ var} \}
\]

- StageNo defines the stage number
- The default StageNo for the objective variable and objective equation is the highest stage mentioned
- The default StageNo for all the other random variables, equations and variables not mentioned is 1

• Several probability distributions for random variables:
  
  - Discrete distributions:
    
    \[
    \text{randVar rv discrete prob val } \{ \text{prob val} \}
    \]
  
  - Continuous distributions: normal, binomial, exponential, ...
    
    \[
    \text{randVar rv distr par } \{ \text{par} \}
    \]
    
    sample rv \{rv\} sampleSize

• Joint Random variables:

\[
\rightarrow \text{nbdiscindep.gms}
\]
\[
\rightarrow \text{nbcontindep.gms}
\]
Independent vs. Joint Random Variables

Demand
- Prob: 0.2
  - d: 40
- Prob: 0.7
  - d: 45
- Prob: 0.1
  - d: 50

Price
- Prob: 0.2
  - p: 55
- Prob: 0.7
  - p: 60
- Prob: 0.1
  - p: 65

Demand / Price
- Prob: 0.04
  - d: 40 / p: 55
- Prob: 0.14
  - d: 40 / p: 60
- Prob: 0.02
  - d: 40 / p: 65
- Prob: 0.14
  - d: 45 / p: 55
- Prob: 0.49
  - d: 45 / p: 60
- Prob: 0.07
  - d: 45 / p: 65
- Prob: 0.02
  - d: 50 / p: 55
- Prob: 0.02
  - d: 50 / p: 55
- Prob: 0.07
  - d: 50 / p: 60
- Prob: 0.01
  - d: 50 / p: 65

Demand / Price
- Prob: 0.2
  - d: 40 / p: 55
- Prob: 0.7
  - d: 45 / p: 60
- Prob: 0.1
  - d: 50 / p: 65

vs.
 Extensions to the Simple NB Problem

• Multiple stages:

\[
\text{stage stageNo rv | equ | var \{rv | equ | var\}}
\]

- \text{StageNo} defines the stage number
- The default \text{StageNo} for the objective variable and objective equation is the highest stage mentioned
- The default \text{StageNo} for all the other random variables, equations and variables not mentioned is 1

• Several probability distributions for random variables:

- Discrete distributions:

  \[
  \text{randVar rv discrete prob val \{prob val\}}
  \]

- Continuous distributions: normal, binomial, exponential, ...

  \[
  \text{randVar rv distr par \{par\}}
  \]

  \[
  \text{sample rv \{rv\} sampleSize}
  \]

• Joint Random variables:

  \[
  \text{jRandVar rv rv \{rv\} prob val val \{val\}}
  \]

  \[
  \{prob val val \{val\}}
  \]
Chance Constraints with EMP

OBJ.. Z =e= X1 + X2;
E1.. om1*X1 + X2 =g= 7;
E2.. om2*X1 + 3*X2 =g= 12;
Model sc / all /;
solve sc min z use lp;

chance E1 0.6
chance E2 0.6
Chance Constraints with EMP

3 out of 4 must be true
\[ 0.75 \geq 0.6 \]

- \[ 1 \times X_1 + X_2 \geq 7; \]
- \[ 2 \times X_1 + X_2 \geq 7; \]
- \[ 3 \times X_1 + X_2 \geq 7; \]
- \[ 4 \times X_1 + X_2 \geq 7; \]

2 out of 3 must be true
\[ 0.66 \geq 0.6 \]

- \[ 1 \times X_1 + 3 \times X_2 \geq 12; \]
- \[ 2 \times X_1 + 3 \times X_2 \geq 12; \]
- \[ 3 \times X_1 + 3 \times X_2 \geq 12; \]
Chance Constraints [chance]

• Defines individual or joint chance constraints (CC):

  \[
  \text{chance equ} \{\text{equ}\} [\text{holds}] \text{ minRatio} [\text{weight}|\text{varName}] 
  \]

• Individual CC: A single constraint \text{equ} has to hold for a certain ratio (0 ≤ \text{minRatio} ≤ 1) of the possible outcomes

• Joint CC: A set of constraints \text{equ} has to hold for a certain ratio (0 ≤ \text{minRatio} ≤ 1) of the possible outcomes

• If \text{weight} is defined, the violation of a CC gets penalized in the objective (weight violationRatio)

• If \text{varName} is defined the violation get multiplied by this existing variable

⇒ simplechance.gms
SP in GAMS - Summary & Outlook

- The Extended Mathematical Programming (EMP) framework can be used to replace parameters in the model by random variables

- Support for Multi-stage recourse problems and chance constraint models

- Easy to add uncertainty to existing deterministic models, to either use specialized algorithms (e.g. solvers Lindo, DECIS) or create Deterministic Equivalent (free solver DE)

- Besides the expected value, EMP also supports optimization of other risk measures (e.g. VaR)

- GAMS/Scenred2 interfaces GAMS with the well-known scenario reduction software Scenred (https://www.gams.com/latest/docs/T_SCENRED2.html)

- More information: https://www.gams.com/latest/docs/UG_EMP_SP.html
Thank You!
Meet us at the GAMS booth!
Extended Example: Newsboy (NB) Problem

- **Data:**
  - A newsboy faces a certain demand for newspapers
    \[ d = 63 \]
  - He can buy newspapers for fixed costs per unit
    \[ c = 30 \]
  - He can sell newspapers for a fixed price
    \[ v = 60 \]
  - For leftovers he has to pay holding costs per unit
    \[ h = 10 \]
  - He has to satisfy his customers demand or has to pay a penalty
    \[ p = 5 \]
  - He can return units for a refund (stage 3)
    \[ r = 9 \]

- **Stage 1: Decisions:**
  - How many newspapers should he buy: \( X \)

- **Stage 2: Decisions & Derived Outcomes**
  - How many newspapers should he sell: \( S \)
  - How many newspapers go to his inventory: \( I \)
  - How many customers are lost: \( L \)

- **Stage 3: Decisions & Derived Outcomes**
  - How many units returned for refund: \( Y \)
  - How many units kept for holding cost \( h \) again: \( E \)
Stages [stage]

• Defines the stage of random variables \((rv)\), equations \((equ)\) and variables \((var)\):

\[
\text{stage stageNo rv | equ | var \{rv | equ | var\}}
\]

• \text{StageNo} defines the stage number
• The default \text{StageNo} for the objective variable and objective equation is the highest stage mentioned
• The default \text{StageNo} for all the other random variables, equations and variables not mentioned is 1
Random Variables

Discrete Distribution

Normal Distribution

Poisson Distribution

Exponential Distribution
Random Variables (RV) [randVar]

Defines both discrete and parametric random variables:

randVar rv discrete prob val {prob val}

The distribution of discrete random variables is defined by pairs of the probability prob of an outcome and the corresponding realization val.

randVar rv distr par {par}

The name of the parametric distribution is defined by distr, par defines a parameter of the distribution.

For parametric distributions a sample can be created.
Joint RVs [jRandVar]

• Defines discrete random variables and their joint distribution:

```
jRandVar rv rv {rv} prob val val {val}
{prob val val {val}}
```

• At least two discrete random variables \( rv \) are defined and the outcome of those is coupled

• The probability of the outcomes is defined by \( \text{prob} \) and the corresponding realization for each random variable by \( \text{val} \)

\[\rightarrow\text{nbdiscjoint.gms}\]
Correlation between RVs [correlation]

• Defines a correlation between a pair of random variables:

  \[
  \text{correlation } \text{rv } \text{rv } \text{val}
  \]

• \text{rv} is a random variable which needs to be specified using the \text{randvar} keyword and \text{val} defines the desired correlation (-1 \leq \text{val} \leq 1).

  ➔ nbcontjoint.gms
Pre-Conference Workshops

Michael Bussieck
Steve Dirkse
Fred Fiand
Lutz Westermann
Outline

Part I: An Introduction to GAMS

Part II: Stochastic programming in GAMS

Part III: The GAMS (Object-Oriented) API's

Part IV: Code embedding in GAMS
Excel and GAMS

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Distance</td>
<td>New-York</td>
<td>Chicago</td>
<td>Topeka</td>
<td>Supply</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
<td>350</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>San-Diego</td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Demand</td>
<td>325</td>
<td>300</td>
<td>275</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Freight cost</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SHIPMENT</td>
<td>New-York</td>
<td>Chicago</td>
<td>Topeka</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Seattle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>San-Diego</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GAMS Directory: C:\GAMS\win64,24.8.5\nWorking Directory: c:\tmp\n Solver: CPLEX

SOLVE LP
SOLVE MIP

Clear Solution

Hands-On TransXLS
Excel and GAMS

- VBA GAMS API to call GAMS from Excel
- Exchange of input data and results using either GDXXRW or GDX API
Embedding GAMS in 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
Low level APIs ➔ **Object Oriented API**

- Low level APIs
  - GDX, OPT, GAMSX, GMO, ...
  - High performance and flexibility
  - Automatically generated imperative APIs for several languages (C, Delphi, Java, Python, C#, ...)

- **Object Oriented GAMS API**
  - Additional layer on top of the low level APIs
  - Object Oriented
  - Written by hand to meet the specific requirements of different Object Oriented languages
Features of the object oriented API

• No modeling capability! Model is still written in GAMS

• Control GAMS execution \(\rightarrow\) \textit{GAMSJob}

• Prepare input data and retrieve results in a convenient way \(\rightarrow\) \textit{GAMSDatabase}

• Scenario Solving: Feature to solve multiple very similar models in a dynamic and efficient way \(\rightarrow\) \textit{GAMSModelInstance}

• Seamless integration of GAMS into other programming environments

• .NET, C++, Java and Python APIs are part of the current GAMS release available at \texttt{www.gams.com}. Many examples available:
  • Sequence of Transport examples (Tutorial)
  • Cutstock, Warehouse, Benders Decomposition, ...
using System;
using GAMS;

namespace TransportSeq
{
    class Transport1
    {
        static void Main(string[] args)
        {
            GAMSWorkspace ws = new GAMSWorkspace();
            GAMSJob t1 = ws.AddJobFromGamsLib("transport");

            t1.Run();
            foreach (GAMSVariabileRecord rec in t1.OutDB.GetVariable("x"))
            {
                Console.WriteLine("x(" + rec.Key(0) + "," + rec.Key(1) + "):");
                Console.WriteLine("    level=" + rec.Level);
                Console.WriteLine("    marginal=" + rec.Marginal);
            }
        }
    }
}
Seamless Integration

- GAMS concept: Separation of tasks

- Use GAMS for modeling and optimization tasks

- Programming languages like C# (.NET), C++, Java and Python are well-suited for developing applications
  - Frameworks available for a wide range of specific task:
    - GUI
    - Web development
    - ...

- The object oriented GAMS API provides a convenient link to run GAMS in such environments
Seamless Integration

- Example: Small Transport Desktop application written in C#
- Convenient data preparation
- Representation of the results in a predefined way
- Modeling details are hidden from the user
**Scenario Solving**

Solving Transport in a loop with different scenarios for the demand:

```
Loop(s,
    d(i,j) = dd(s,i,j);
    solve transport using lp minimizing z;
    objrep(s) = transport.objval;
);
```
Scenario Solving – GUSS

```gams
set dict / s.scenario.
   d.param .dd
   z.level .objrep /
solve transport using lp minimizing z;
```

- Save model generation and solver setup time
- Hot start (keep the model hot inside the solver and use solver’s best update and restart mechanism)

- Apriori knowledge of all scenario data
- Model rim unchanged from scenario to scenario
foreach (string s in scen)
{
    f.FirstRecord().Value = v[s];
    modelInstance.Solve();
    objrep[s] = z.FirstRecord().Level;
}

• Save model generation and solver setup time
• Hot start (keep the model hot inside the solver and use solver’s best update and restart mechanism)
• Data exchange between solves possible

• Model rim unchanged from scenario to scenario
GAMSJob
• Manages the execution of a GAMS program given by GAMS model source

GAMSCheckpoint
• Captures the state of a GAMSJob

GAMSModelInstance
• A single mathematical model generated by a GAMS solve statement

GAMSModifier
• Marks elements of a GAMSModelInstance to be modifiable
• *bmult* is one parameter of the model which gets modified before we solve the instance:

```csharp
GAMSParameter bmult = mi.SyncDB.AddParameter("bmult", 0, "demand multiplier");
bmult.AddRecord().Value = 1.0;
mi.Instantiate("transport us lp min z", opt, new GAMSModifier(bmult));
double[] bmultlist = new double[] { 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3 };

foreach (double b in bmultlist)
{
    bmult.FirstRecord().Value = b;
    mi.Solve();
    ...
    Console.WriteLine(" Obj: " + mi.SyncDB.GetVariable("z").FindRecord().Level);
}
```
• Updating bounds of a variable:

```
GAMSVariable x = mi.SyncDB.AddVariable("x", 2, VarType.Positive, ";
GAMSPParameter xup = mi.SyncDB.AddParameter("xup", 2, "upper bound on x");
mi.Instantiate("transport us lp min z", modifiers: new
GAMSMOrder(x,UpdateAction.Upper,xup));

foreach (GAMSSetRecord i in t7.OutDB.GetSet("i"))
    foreach (GAMSSetRecord j in t7.OutDB.GetSet("j"))
    {
        xup.Clear();
        xup.AddRecord(i.Keys[0],j.Keys[0]).Value = 0;
        mi.Solve();
        <...
        Console.WriteLine(" Obj: " + mi.SyncDB.GetVariable("z").FindRecord().Level);
    }
```
GAMSModelInstances in **Parallel**

- Multiple GAMSModelInstances running in parallel with one common data source (work):
GAMSModelInstances in **Parallel**

- Threads consume data from source dynamically instead of getting a fixed amount of data at thread initialization time.

- Implicit load balancing by architecture:
  - Number of solves in a thread depend on its speed
  - Keeps all threads busy as long as possible

- Typical applications:
  - Scenario analysis
  - Decomposition algorithms (Benders, CG, ...)

- Communication between threads for “dynamic” algorithms
Summary

- Object Oriented API provides an additional abstraction layer of the low level GAMS APIs
- Powerful and convenient link to other programming languages
- Versions for .NET, C++, Java, and Python are available and part of the current distribution
- Many examples are available:
  - Sequence of Transport examples (Tutorial)
  - Cutstock, Warehouse, Benders Decomposition, ...
Thank You

Meet us at the GAMS booth!
Pre-Conference Workshops

Michael Bussieck
Steve Dirkse
Fred Fiand
Lutz Westermann
Outline

Part I: An Introduction to GAMS

Part II: Stochastic programming in GAMS

Part III: The GAMS (Object-Oriented) API's

Part IV: Code embedding in GAMS
24.9.1 Major release *(August 30, 2017)*

Acknowledgments

We would like to thank all of our users who have reported problems and made suggestions for improving this release. In particular, we thank Etienne Ayotte-Sauvé, Wolfgang Britz, Florian Habermacher, Florian Häberlein, Maximilian Held, Ignacio Herrero, Hanspeter Höschle, Erwin Kalvelagen, Toni Lastusilta, John Ross, and Tom.

GAMS System

**GAMS**

- **New feature, the Embedded Code Facility**: This extends the connectivity of GAMS to other programming languages. It allows the use of Python code during compile and execution time. GAMS symbols are shared with the external code, so no communication via disk is necessary.

  The embedded code feature is available on Linux, MacOS X, and Windows. For these platforms, a Python 3.6 installation is included with the GAMS distribution. If the user wants to work with a different Python 3.6, installed separately, for models with embedded code the new command line option `pySetup` needs to be set to 0.

  **Note**

  This feature is currently in beta status. Any feedback to support@gams.com is appreciated.

- **New command line option procDirPath**: Specifies the directory where the process directory should be created.
Motivation – Avoid Unreadable/Slow Code

- GAMS code for parallel assignment and equation definition is compact, elegant, and efficient
- GAMS uses relational data tables as a base data structure
- Traditional data structure are not available in GAMS
  - No arrays, lists, dictionaries, trees, graphs, ...
- GAMS can represent such traditional data structures but ...
  - GAMS code becomes quickly unreadable
    \[ t(tt) = t(tt-1); \] // advances set element t to t+1
  - Performance with naïve representation is very inefficient
    \[ t(tt) = \text{ord}(tt)=tCnt; \] // advances set element t to t+1
  - Writing code that executes efficiently requires deep understanding of underlying GAMS internal data structures and often results in even more unreadable code
Motivation –
Data Input/Transformation at Compile Time

- GAMS data input (ASCII) follows strict syntax
- Practical GAMS models get data (via ASCII input files) that is often not in a proper shape
  - Hence GAMS code is often augmented with execution of scripts and programs to get data files into a GAMS readable shape
  - GAMS even ships a selection of POSIX text utilities (sed, grep, awk, ...) on Windows to support a somewhat standardized way of transforming text files into GAMS readable format
  - Scripts spawned by GAMS cannot (easily) access data that is already available in GAMS
- GAMS has no string editing facilities to e.g.
  - modify labels
  - change content of compile time variables
    - “Solution”: $xxx new and weird compile time constructs, e.g. $setNames, $splitOption, ...
Motivation – Other

- Connecting libraries for special algorithms (e.g. graph algorithms like connected components, matrix operations like Cholesky factorization) to GAMS is not easy
- Current “solution” has issues
  - $unload/$call/$load or execute_unload/execute/execute_load
    - Performance: disk IO + process creation
    - Knowledge of data API (GDX or OO-API)
    - Remapping of relational data (plus concept of UELs) into other data structures
  - Add compile time directives to perform a single special task (e.g. $splitOption)
  - Introduce unreadable option or put_utility syntax to perform a single special task (e.g. option a<b;)
- Object Oriented API/Framework versus Embedded Code
  - OO-API: Framework in control
  - Embedded Code: GAMS in control
Embedded Code

- Support the use of external code during GAMS compile and execution time
- Provide support for off-line debugging of embedded code
- Share GAMS symbols (sets, parameters, variables, and equations) structure and content with the external code in memory
- Communication of the data between GAMS and the embedded code inspired by the existing interface to GDX in many ways:
  - Records access by both labels and label indices
  - Data in GAMS can be merged with or replaced by data from embedded code
  - Data from embedded code can be send to GAMS database filtered or domain checked
- Provide automatically generated, additional source code for common tasks

→ Allows the user to concentrate on the task at hand and not the mechanics
Example

- Calculate $|i| |j|$ 4-quantiles of $|k|$ numbers:

  ```gams
  Set i / ... /, j / ... /, k / ... /
  Table p(i,j,k)
      k1       k2 ...
  i1.j1 18.002966  84.483404 ...
  i1.j2  7.644259   50.520856 ...
  ...
  
  Set q; Parameter quantiles(i,j,q);
  ```

  ```python
  import pandas as pd
  df = pd.DataFrame(list(gams.get('p')))
  q = df.groupby([0,1]).quantile([0,.25,.5,.75,1])
  gams.set('quantiles', [ (*k[:-1],str(k[-1]),v[3]) for k,v in q.iterrows() ])
  ```

  ```gams
  $offEmbeddedCode
  q<quantiles
  display q, quantiles;
  ```
Split Example – Data

country / system.empty /
city / system.empty /
mccCountry(cc,country)
mccCity (cc,city);
Split Example – Embedded Code

```python
$onEmbeddedCode Python:
country = set()
city = set()
mccCountry = []
mccCity = []
for cc in gams.get("cc"):
    r = str.split(cc, " - ", 1)
country.add(r[0])
city.add(r[1])
mccCountry.append((cc,r[0]))
mccCity.append((cc,r[1]))
gams.set("country",list(country))
gams.set("city",list(city))
gams.set("mccCountry",mccCountry)
gams.set("mccCity",mccCity)
$offEmbeddedCode
country city mccCountry mccCity
```
split example – output

display country, city;

---- 27 set country

france, usa, spain, germany

---- 27 set city

seville, washington dc, new york, paris
munich, madrid, toulouse, berlin
bonn, lille, houston, bilbao
cordoba
**Sorting Example**

Set  
i / i1*i10 /;
Parameter  
a(i) Random Data
  aIndex(i) Sorted index of a;

\[
a(i) = \text{uniformInt}(1, 10*\text{card}(i));
\]

**embeddedCode Python:**

```python
a = list(gams.get("a"))
tmp = [r[0] for r in sorted(enumerate(a),
                          key=lambda x:x[1][-1])]
aIndex = len(a)*[-1]
for idx in range(len(tmp)):
    aIndex[tmp[idx]] = (a[tmp[idx]][0], idx+1)

gams.set("aIndex", aIndex)
```

`GAMS`
Sorting Example – Output

Display a, aIndex;

----  44 PARAMETER a   Random Data

1 18.000, 2 85.000, 3 56.000,
4 31.000, 5 30.000, 6 23.000,
7 35.000, 8 86.000, 9  7.000,
10 51.000

----  44 PARAMETER aIndex   Sorted Index of a

1  2.000, 2  9.000, 3  8.000,
4  5.000, 5  4.000, 6  3.000,
7  6.000, 8 10.000, 9  1.000,
10 7.000
Exchange via Files

```python
$f = open('i.txt', 'w')
for i in range(int(gams.arguments)):
    f.write('i'+str(i+1)+'\n')
f.close()
```

```plaintext
Set i /
$include i.txt /
Display i;

---- 21 SET i

i1, i2, i3, i4, i5, i6, i7, i8, i9, i10
```
Exchange via Environment Variables

Set i / i1*i5 /;
Parameter b(i) / i1 2, i2 7, i3 59, i4 2, i5 47 /;
Set k "from 0 to max(b)" / k0*k? /;

$onEmbeddedCode Python:
    import os
    kmax = int(max([b[1] for b in list(gams.get("b"))]))
    gams.printLog('max value in b is ' + str(kmax))
    os.environ["MAXB"] = str(kmax)
$offEmbeddedCode

$if x%sysEnv.MAXB%==x $abort MAXB is not set
Set k "from 0 to max(b)" / k0*k%sysEnv.MAXB% /;
Scalar card_k;
card_k = card(k);
Display card_k;

---- 15 PARAMETER card_k = 60.000
Multiple Independent Python Sessions

```plaintext
$ if not %sysEnv.GMSPYTHONMULTINST%==1
$abort.noError Set command line option pyMultInst=1

Set       i    / i1*i3 /

Parameter h(i)
  ord_i    / 0 /

loop(i,
  ord_i = ord(i);
  embeddedCode Python:
    i = int(list(gams.get("ord_i"))[0])
    gams.printLog(str(i))
    pauseEmbeddedCode
    h(i) = embeddedHandle;
);

loop(i,
  continueEmbeddedCode h(i):
    gams.printLog(str(i))
    endEmbeddedCode
);
```

Hands-On
embeddedMultInstance
Multiple Independent Python Sessions –

Log

--- Initialize embedded library embpycclib.dll
--- Execute embedded library embpycclib.dll
--- 1
--- Initialize embedded library embpycclib.dll
--- Execute embedded library embpycclib.dll
--- 2
--- Initialize embedded library embpycclib.dll
--- Execute embedded library embpycclib.dll
--- 3
--- Execute embedded library embpycclib.dll
--- 1
--- Execute embedded library embpycclib.dll
--- 2
--- Execute embedded library embpycclib.dll
--- 3
Performance Considerations

Set
\[ i / i1*i50 /, \ p(i,i); \textbf{Alias} (i,ii); \]
Parameter \( c(i,i); \ c(i,ii) = \text{uniform}(-50,50); \)

Set
\[ \text{iter} / 1*100 /; \]
Scalar \( t\text{cost}, \min TC\text{ost} / +\text{inf} /; \)
loop(iter, 
  embeddedCode Python:
  import random
  i = list(gams.get("i"))
  p = list(i)
  random.shuffle(p)
  for idx in range(len(i)):
    p[idx] = (i[idx], p[idx])
  gams.set("p", p)
  end
EmbeddedCode p
  t\text{cost} = \text{sum}(p, c(p));
  \textbf{if} (t\text{cost} < \min TC\text{ost}, \min TC\text{ost} = t\text{cost});
);
Display \min TC\text{ost};

EXECUTION TIME \quad = \quad 16.375\ SECONDS
Performance Considerations

Set $i / i1*i50 /$, $p(i,i)$; Alias $(i,ii)$;

Parameter $c(i,i)$; $c(i,ii) = \text{uniform}(-50,50)$;

embeddedCode Python:
  import random
pauseEmbeddedCode

Set $\text{iter} / 1*1000 /$;
Scalar $\text{tcost, minTCost} / +\text{inf} /$

loop ($\text{iter}$, continueEmbeddedCode:
  i = list(gams.get("i"))
p = list(i)
random.shuffle(p)
for idx in range(len(i)):
  p[idx] = (i[idx], p[idx])
gams.set("p", p)
pauseEmbeddedCode p
tcost = sum(p, c(p));
  if (tcost < minTCost, minTCost = tcost);
); continueEmbeddedCode:
pass
endEmbeddedCode
Display minTCost;

EXECUTION TIME = 1.797 SECONDS
Performance Considerations

Set \( i / i1*i50 /, p(i,i); \text{Alias} (i,ii); \)
Parameter \( c(i,i); c(i,ii) = \text{uniform}(-50,50); \)

embeddedCode Python:

```python
import random
i = list(gams.get("i"))
```

pauseEmbeddedCode

Set \( \text{iter} / 1*1000 /; \)
Scalar \( \text{tcost}, \text{minTCost} / +\text{inf} /; \)
loop(\text{iter,} \)

continueEmbeddedCode Python:

```python
p = list(i)
random.shuffle(p)
for idx in range(len(i)):
    p[idx] = (i[idx], p[idx])
gams.set("p", p)
```

pauseEmbeddedCode p
tcost = sum(p, c(p));

```python
if (tcost < minTCost, minTCost = tcost);
```

); continueEmbeddedCode:

pass
endEmbeddedCode
Display \( \text{minTCost}; \)

EXECUTION TIME = 1.593 SECONDS
Performance Considerations

Set i / i1*i50 /, p(i,i); Alias (i,ii);
Parameter c(i,i); c(i,ii) = uniform(-50,50);

embeddedCode Python:
    import random
    i = list(gams.get("i",keyType=KeyType.INT))
pauseEmbeddedCode

Set iter / 1*1000 /;
Scalar tcost, minTCost / +inf /;
loop(iter,
    continueEmbeddedCode Python:
        p = list(i)
        random.shuffle(p)
        for idx in range(len(i)):
            p[idx] = (i[idx], p[idx])
        gams.set("p", p)
    pauseEmbeddedCode p
tcost = sum(p, c(p));
    if (tcost < minTCost, minTCost = tcost);
);
continueEmbeddedCode:
    pass
endEmbeddedCode
Display minTCost;

EXECUTION TIME = 1.437 SECONDS
Syntax: GAMS

Compile Time:

$onEmbeddedCode[S|V] Python: [arguments]
  Python code
{Python code}
$offEmbeddedCode {output symbol}

$onEmbeddedCode[S] Python: [arguments]
  • Starts a section with Python code
  • Parameter substitution is activated
  • The optional arguments can be accessed in the Python code

$onEmbeddedCodeV Python: [arguments]
  • Same as $onEmbeddedCode but parameter substitution is disabled
    (the Python code is passed on verbatim)

$offEmbeddedCode {output symbol}
  • Ends a section with Python code
  • The optional output symbol(s) get updated in the GAMS database
Syntax: **GAMS**

Execution Time:

```gams
EmbeddedCode[S|V] Python: [arguments]
  Python code
  {Python code}
endEmbeddedCode {output symbol}
```

- **EmbeddedCode[S]** Python: [arguments]
  - Starts a section with Python code
  - Parameter substitution is activated
  - The optional arguments can be accessed in the Python code

- **EmbeddedCodeV** Python: [arguments]
  - Same as EmbeddedCode but parameter substitution is disabled (the Python code is passed on verbatim)

- ```gams```endEmbeddedCode {output symbol}
  - Ends a section with Python code
  - The optional output symbol(s) get updated in the GAMS database
Syntax: **GAMS**

Execution Time:

```gams
pauseEmbeddedCode {output symbols}
continueEmbeddedCode[S|V] [handle]: [arguments]
• pauseEmbeddedCode {output symbol}
  • Pauses a section with Python code
  • The optional output symbol(s) get updated in the GAMS database
• continueEmbeddedCode[S] [handle]: [arguments]
  • Continues a previously paused section with Python code
  • Parameter substitution is activated
  • The optional arguments can be accessed in the Python code
  • The optional handle (pointing to a specific paused embedded code section) could be retrieved by the function embeddedHandle. If omitted, the last section paused will be continued.
• continueEmbeddedCodeV [handle]: [arguments]
  • Same as continueEmbeddedCode but parameter substitution is disabled (the Python code is passed on verbatim)
```
The Python Class `ECGamsDatabase` serves as interface between GAMS and Python. An instance of this class is automatically created when an embedded code section is entered and can be accessed using the identifier `gams`. Several methods can be used to interact with GAMS:

- `gams.get(symbolName, [...])`
  - Retrieves iterable object representing the symbol `symbolName`
  - Several optional parameters allow to modify format of the data
- `gams.set(symbolName, data[, merge][, domCheck])`
  - Sets data for the GAMS symbol `symbolName`
  - Data takes a Python list of items representing records of the symbol
  - Optional parameter `merge` specifies if data in a GAMS symbol is merged or replaced
  - Optional parameter `domCheck` specifies if Domain Checking is applied
Syntax: **Python**

- `gams.getUel(idx)`
  - Returns the label corresponding to the label index `idx`
- `gams.mergeUel(label)`
  - Adds `label` to the GAMS universe if it was unknown and returns the corresponding label index
  - Note: New labels cannot be added at execution time
- `gams.getUelCount()`
  - Returns the number of labels
- `gams.printLog(msg)`
  - Print `msg` to log
- `gams.arguments`
  - Contains the arguments that were passed to the Python interpreter at start-up of the embedded code section
- `gams.epsAsZero`
  - Flag to read GAMS EPS as 0 [True] or as a small number (4.94066E-300) [False]
- `gams._debug`
  - Debug flag for additional output
Some **Examples** of Python Embedded Code

- Splitting of labels (compile time)
- Permutation
- Sorting
- Calculation of quantiles
- Power set
- Matching
- Parsing of specially structured ASCII input
- TSP subtour elimination
- Benders Decomposition using **Message Passing Interface (MPI)** on **High-Performance Computing (HPC)** infrastructure
Plot example

Hands-On meanvarplot
Next steps ...

• More examples
  • High Performance Libraries for specific tasks
    • FORTRAN (Factorization of matrix)
    • C/C++ (Expansion Planning Power Systems)
  • Support of other popular frameworks (compiled and interpreted)
    • C/C++
    • C#/.NET, Java, R, ...
    • Connect of powerful libraries e.g. boost::graph, ...
• Provide a configurable build system that supports building the required libraries (for compiled languages) at GAMS compile time
• Provide a documented API to allow integration of individual user embedded code libraries
• Asynchronous/parallel use of embedded code

This feature is currently in beta status. Any feedback to support@gams.com is highly appreciated.
Thank You

Meet us at the GAMS booth!