Pre-Conference Workshops

Fred Fiand
Franz Nelissen
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

- *Roots: World Bank, 1978 – 1987 Initial product*
- 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

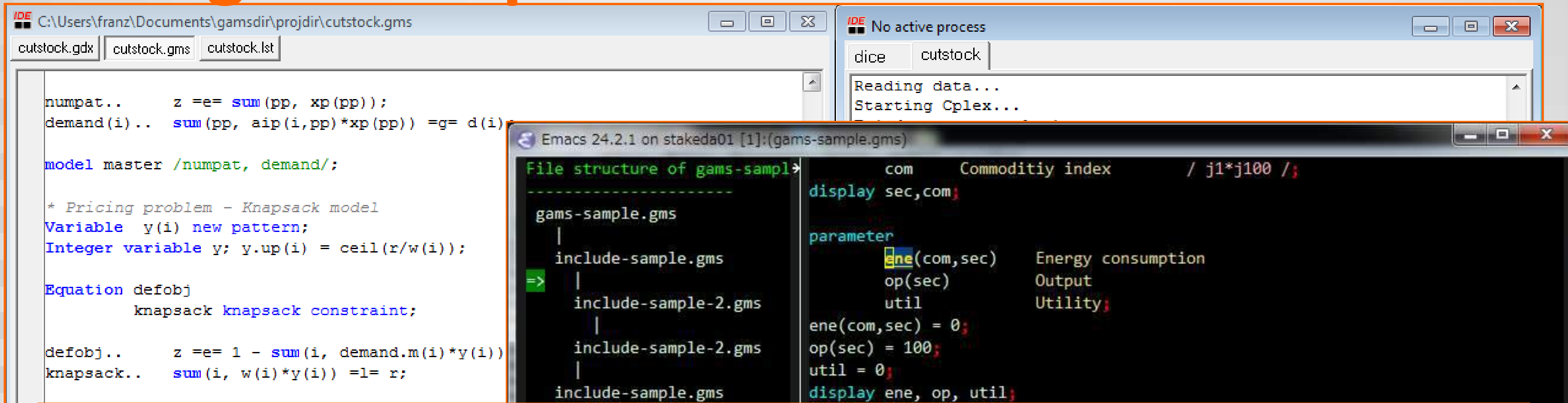
Users: 50% academic, 50% commercial or governmental

Used in more than 120 countries

Broad Range of Application Areas

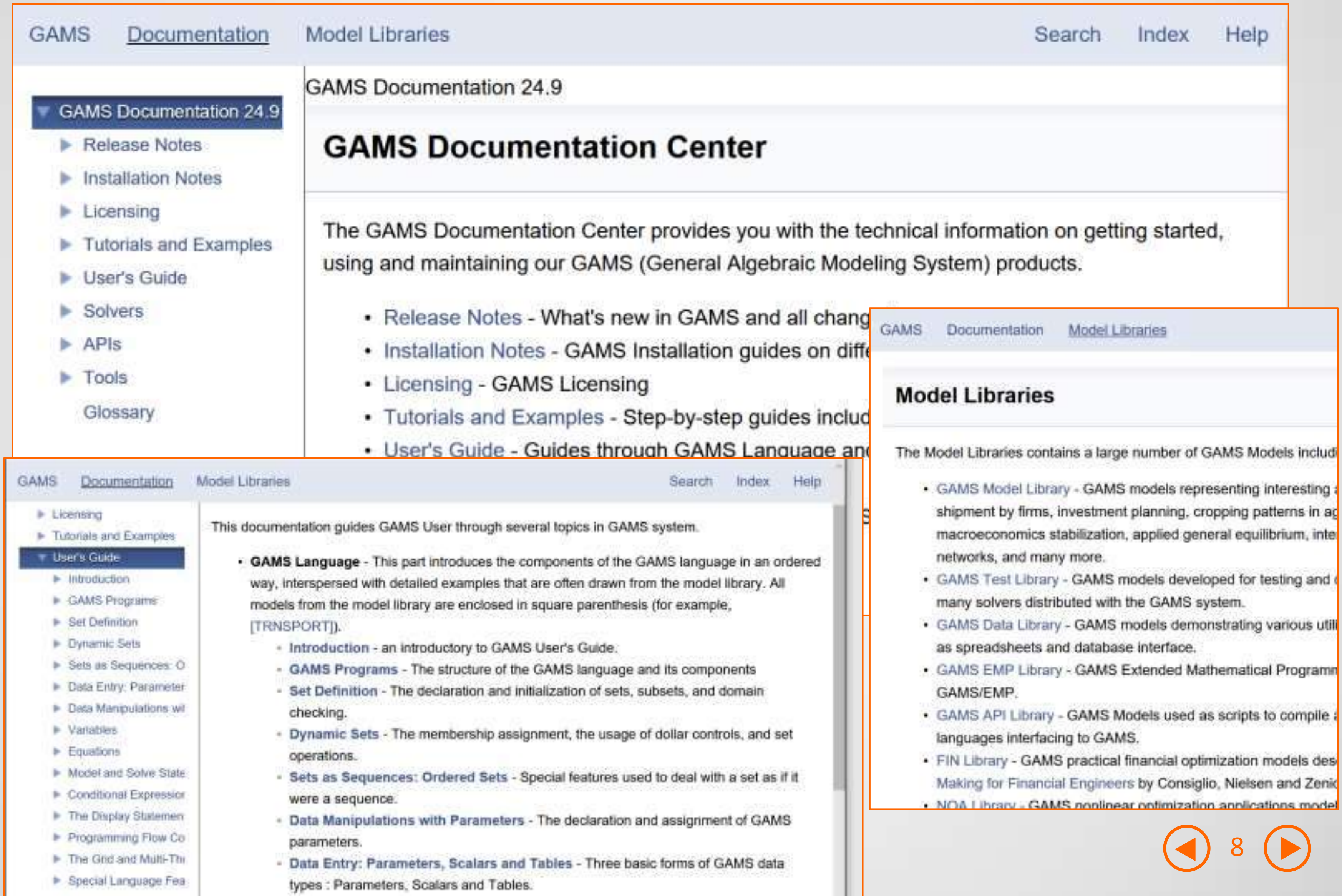
Agricultural Economics	Applied General Equilibrium
Chemical Engineering	Economic Development
Econometrics	Energy
Environmental Economics	Engineering
Finance	Forestry
International Trade	Logistics
Macro Economics	Military
Management Science/OR	Mathematics
Micro Economics	Physics

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

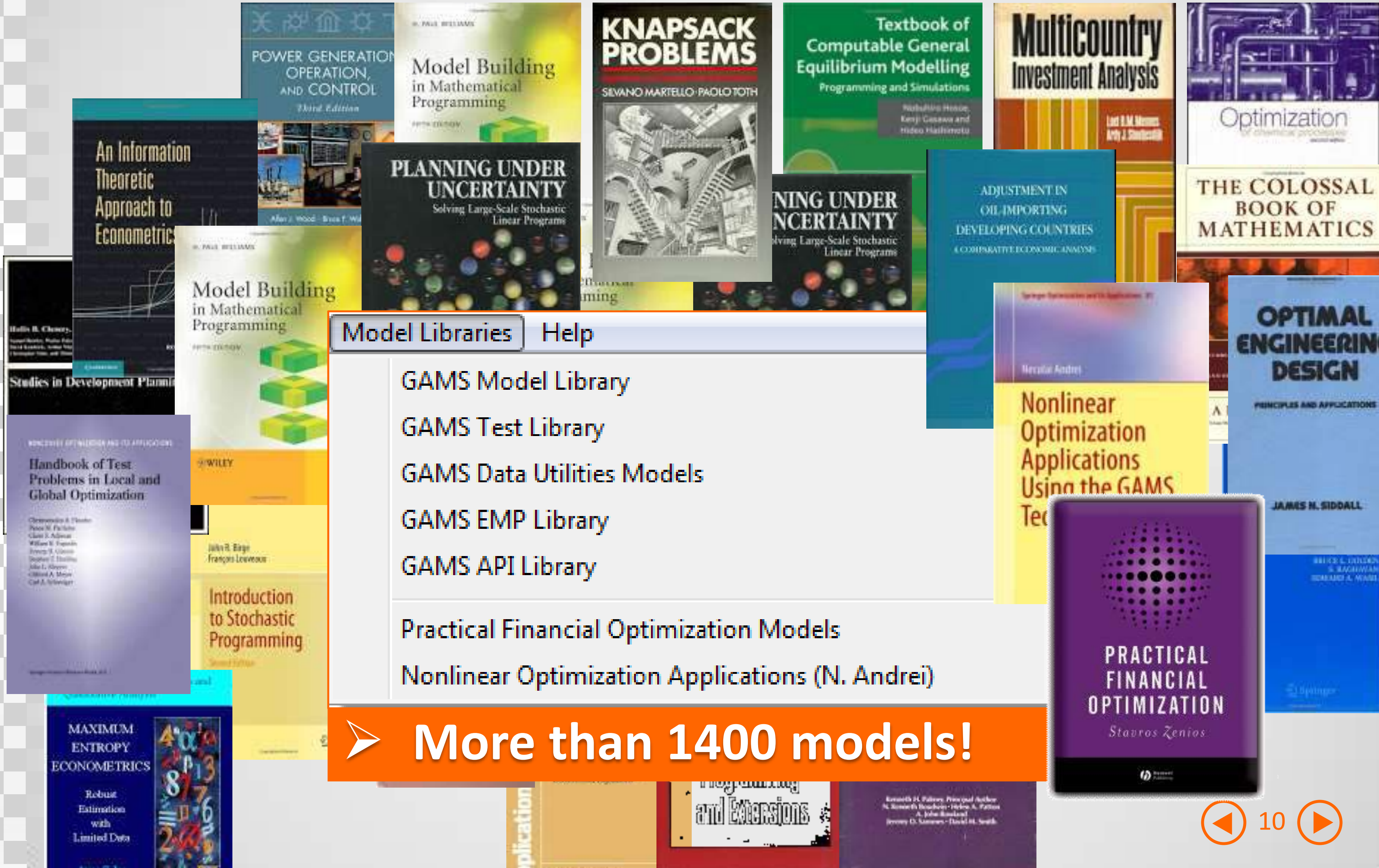


Simple Integration of GAMS Models

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

Free Model Libraries



Model Libraries | Help

- GAMS Model Library
- GAMS Test Library
- GAMS Data Utilities Models
- GAMS EMP Library
- GAMS API Library

- Practical Financial Optimization Models
- Nonlinear Optimization Applications (N. Andrei)

➢ **More than 1400 models!**

COURSES AND WORKSHOPS 2017

June

Online Course

Introduction to Practical Global CGE Modeling with GAMS

Prague, Czech Republic

- Practical General Equilibrium Modeling with GAMS
- Energy and Environmental CGE Modeling with GAMS
- Advanced Techniques in General Equilibrium Modeling with GAMS
- Overlapping Generation General Equilibrium Modeling with GAMS

August

Annapolis, MD, USA

- Single Country General Equilibrium Modelling with GAMS and STAGE
- Global CGE Modelling with GAMS and GLOBE

Frisco, CO, USA

- Basic GAMS Modeling – An Introductory Class
- Advanced GAMS Modeling

September

Essen, Germany

- Trade Policy Analysis with GAMS and MPSGE

November

Weisenheim a.B., Germany

- Modeling and Optimization with GAMS (basic)
- Modeling and Optimization with GAMS (advanced)

Continuous

- Online Practical General Equilibrium Modeling with GAMS
- Online Advanced Techniques in General Equilibrium Modeling with GAMS

Where to Find Help?

- *Documentation Center: https://www.gams.com/latest/docs/*
- *Free Model Libraries: https://www.gams.com/latest/docs/modlibsindex.html*
- 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

Foundation of GAMS

Powerful algebraic modeling language

Open architecture with interfaces to other systems

Independent layers

Model			
Platform	Solver	Data	Interface

Separation of Model and Platform

Supported Platforms

Microsoft Windows

Linux

Welcome to Mac OS X

solaris

AIX

➢ **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

FICO

Gurobi Optimization

IBM

mosek

COIN|OR

➢ **More than 30 Solvers integrated!**

Model

Platform | Solver | Data | Interface

Separation of Model and Solver

Local and distributed / remote execution

- Distributed Algorithm (CPLEX, GUROBI)
- Remote Execution
  - DoCloud (IBM), Gurobi Instant Cloud
  - Solve Engine (Satalia)
  - NEOS (Kestrel)
- Grid Computing Facility

Model			
Platform	Solver	Data	Interface

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

Model

Platform | Solver | Data | Interface

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

Model

Platform | Solver | Data | Interface

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

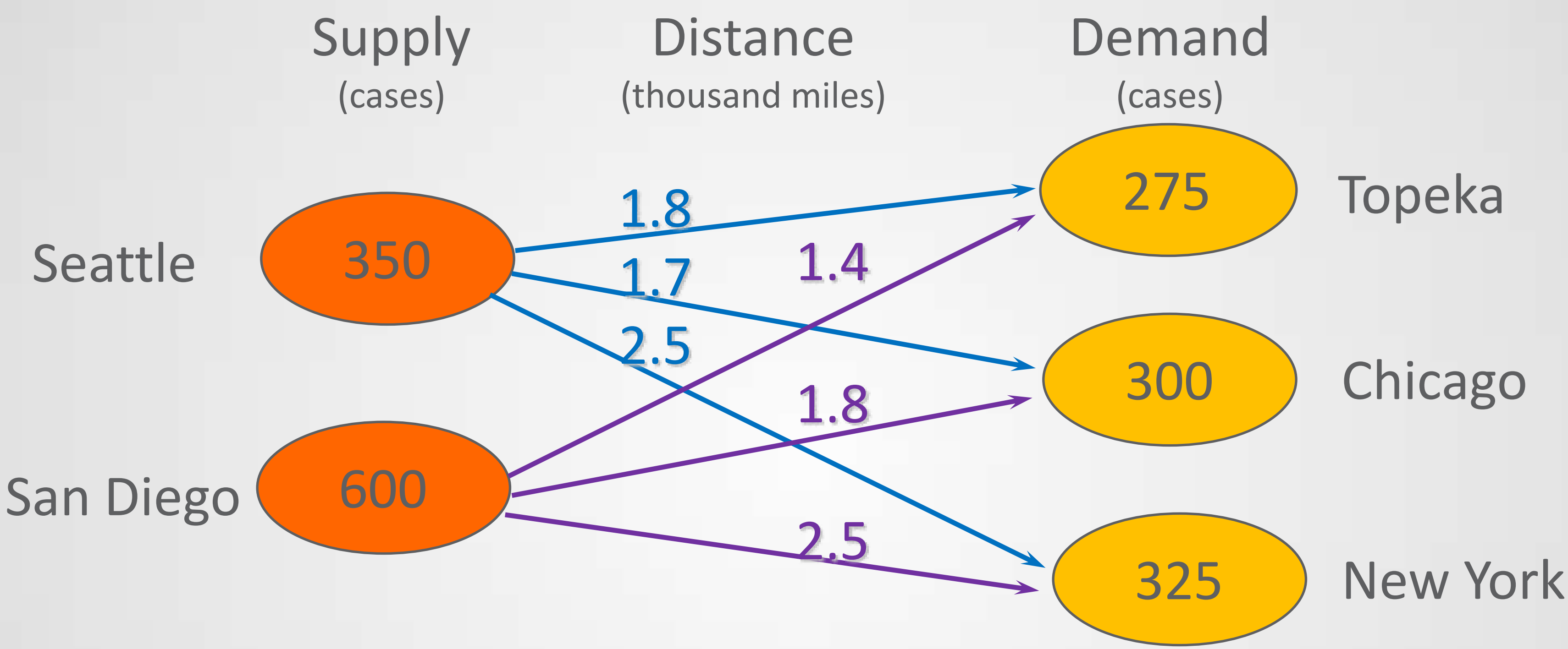
Proposed Strategy

- Run all instances simultanuesly 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



Freight: $90 case / thousand miles

Minimize	Transportation cost
subject to	Demand satisfaction at markets
	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_i	(Demand in cases)

$\min \quad \sum_i \sum_j c_{ij} \cdot x_{ij}$ (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

Agenda

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

What does a modeler have to think about?

1. Problem
2. Mathematics
3. Programming
4. Performance
5. Scalability
6. Connectivity
7. Deployment
8. Maintenance (Life Cycle)
9. ...

GAMS eases the transitions between these domains

- Simplifies Modeling
- Increases Productivity

GAMS - Evolution

https://en.wikipedia.org/wiki/General_Algebraic_Modeling_System#Timeline

Timeline [edit]

- 1976 GAMS idea is presented at the International Symposium on Mathematical Programming (ISMP), Budapest[2]
- 1978 Phase I: GAMS supports linear programming. Supported platforms: Mainframes and Unix Workstations
- 1979 Phase II: GAMS supports nonlinear programming.
- 1987 GAMS becomes a commercial product
- 1988 First PC System (16 bit)
- 1988 Alex Meeraus, the initiator of GAMS and founder of GAMS Development Corporation, is awarded INFORMS Computing Society Prize
- 1990 32 bit Dos Extender
- 1990 GAMS moves to Georgetown, Washington, D.C.
- 1991 Mixed Integer Non-Linear Programs capability (DICOPT)
- 1994 GAMS supports mixed complementarity problems
- 1995 MPSGE language is added for CGE modeling
- 1996 European branch opens in Germany
- 1998 32 bit native Windows
- 1998 Stochastic programming capability (OSL/SE, DECIS)
- 1999 Introduction of the GAMS Integrated development environment (IDE)
- 2000 GAMS World initiative started
- 2001 GAMS Data Exchange (GDX) is introduced
- 2002 GAMS is listed in OR/MS 50th Anniversary list of milestones
- 2003 Conic programming is added
- 2003 Global optimization in GAMS
- 2004 Quality assurance initiative starts
- 2004 Support for Quadratic Constrained programs
- 2005 Support for 64 bit PC Operating systems
- 2006 GAMS supports parallel grid computing
- 2007 GAMS supports open-source solvers from COIN-OR
- 2008 Support for 32 and 64 bit Mac OS X
- 2009 GAMS available on the Amazon Elastic Compute Cloud
- 2009 GAMS supports extended mathematical programs (EMP)
- 2010 GAMS is awarded the company award of the German Society of Operations Research (GOR)
- 2010 GDXMRW interface between GAMS and Matlab
- 2011 Support for Extrinsic Function Libraries
- 2012 The Winners of the 2012 INFORMS Impact Prize included Alexander Meeraus. The prize was awarded to the originators of the five most important algebraic modeling languages.
- 2012 Introduction of Object Oriented API for .NET, Java, and Python
- 2012 The winners of the 2012 Coin OR Cup included Michael Bussieck, Steven Dirkse, & Stefan Vigerske for GAMSlinks
- 2013 Support for distributed MIP (Cplex/Gurobi)
- 2013 Stochastic programming extension of GAMS EMP
- 2013 GDXRRW interface between GAMS and R
- 2014 Local search solver LocalSolver added to solver portfolio
- 2015 LaTeX documentation from GAMS source (Model2TeX)
- 2016 New Management Team

1978

Now

Striving for Innovation and Compatibility

Users must benefit from

- Advancing hardware / New Platforms
- Enhanced / new solver and solution technology
- Improved / upcoming interfaces to other systems
- New Modeling Concepts

Protect Investments of Users

- Life time of a model: 15+ years
- New maintainer, platform, solver, user interface
- Backward Compatibility
- Quality Assurance

New Modeling and Solution Concepts

Examples:

- Bilevel Programs
- Extended Nonlinear Programs
- Stochastic Programming
- Disjunctive Programs

Issues:

- Breakouts of traditional Mathematical Programming classes
- Limited support with common model representation
- Incomplete/experimental solution approaches
- New and interesting solver features driven by implementation choices → May break solver independence of models

Challenge:

- Find a concept that combines the essentials of new features independent of the particular implementation choices.

Striving for Innovation and Compatibility

Users must benefit from

- Advancing hardware / New Platforms
- Enhanced / new solver and solution technology
- Improved / upcoming interfaces to other systems
- New Modeling Concepts

Protect Investments of Users

- Life time of a model: 15+ years
- New maintainer, platform, solver, user interface
- Backward Compatibility
- Quality Assurance

Quality Assurance

- What are the impacts of new features / updated modules / platforms?
- Is the new distribution backward compatible?

- 700+ quality test models (GAMS Test Library)
- Automatically executed every night for all solver combinations (13,000+ runs / platform)

Latest GAMS System Bu ×

Secure https://builds.gams.com

Latest GAMS System Builds and Test Results

Sunday 03Sep17 04:45 (UTC)

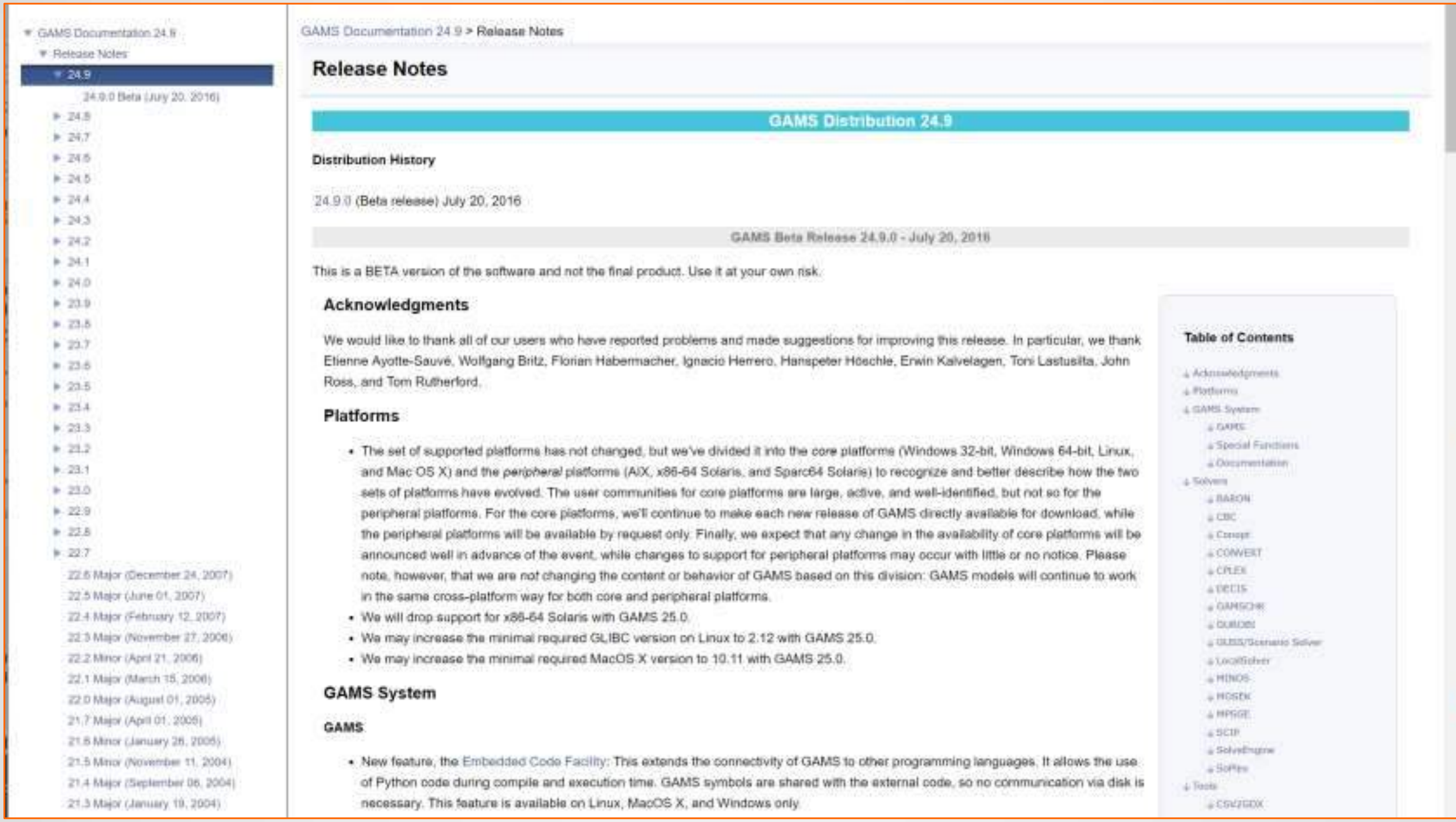
[Latest Builds | Alpha Builds | Beta Builds | Nightly Builds | α Docs | Glossary]

Comments? | Server log

NOTE: The (nightly) alpha builds are internal development versions of the GAMS system. They may have known bugs, unfinished features, beta versions of third-party software, or may not function at all! Not for production use!

alpha nb	System	Libraries	Build	Rev	Status and Time (UTC)		Initial Tests		Full Tests	
Saturday	aix	Download	25.0.0	63850	Test done	02Sep2017 08:04:27	397 runs 1 failures (q=1,s=0)	Report	3422 runs 1 failures (q=1,s=0)	Report
Saturday	deg dmg	Download	25.0.0	63850	Test started	02Sep2017 02:20:43	635 runs 0 failures (q=0,s=0)	Report	results pending	
Saturday	leg	Download	25.0.0	63850	Test started	02Sep2017 02:39:26	650 runs 1 failures (q=1,s=0)	Report	results pending	
Friday	legDbg	Download	25.0.0	63850	Test done	02Sep2017 00:43:25	633 runs 0 failures (q=0,s=0)	Report	*2307 runs 0 failures (q=0,s=0)*	Report
Sunday	sig	Download	25.0.0	63854S	Test started	03Sep2017 02:49:50	509 runs 0 failures (q=0,s=0)	Report	results pending	

Recent Enhancements



GAMS Documentation 24.9
Release Notes
24.9
24.9.0 Beta (July 20, 2016)
24.8
24.7
24.6
24.5
24.4
24.3
24.2
24.1
24.0
23.9
23.8
23.7
23.6
23.5
23.4
23.3
23.2
23.1
23.0
22.9
22.8
22.7
22.6 Major (December 24, 2007)
22.5 Major (June 01, 2007)
22.4 Major (February 12, 2007)
22.3 Major (November 27, 2006)
22.2 Minor (April 21, 2006)
22.1 Major (March 15, 2006)
22.0 Major (August 01, 2005)
21.7 Major (April 01, 2005)
21.6 Minor (January 26, 2005)
21.5 Minor (November 11, 2004)
21.4 Major (September 06, 2004)
21.3 Major (January 19, 2004)

GAMS Documentation 24.9 > Release Notes

Release Notes

GAMS Distribution 24.9

Distribution History

24.9.0 (Beta release) July 20, 2016

GAMS Beta Release 24.9.0 - July 20, 2016

This is a BETA version of the software and not the final product. Use it at your own risk.

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, Ignacio Herrero, Hanspeter Höschle, Erwin Kalvelagen, Toni Lastusilta, John Ross, and Tom Rutherford.

Platforms

- The set of supported platforms has not changed, but we've divided it into the core platforms (Windows 32-bit, Windows 64-bit, Linux, and Mac OS X) and the peripheral platforms (AIX, x86-64 Solaris, and Sparc64 Solaris) to recognize and better describe how the two sets of platforms have evolved. The user communities for core platforms are large, active, and well-identified, but not so for the peripheral platforms. For the core platforms, we'll continue to make each new release of GAMS directly available for download, while the peripheral platforms will be available by request only. Finally, we expect that any change in the availability of core platforms will be announced well in advance of the event, while changes to support for peripheral platforms may occur with little or no notice. Please note, however, that we are not changing the content or behavior of GAMS based on this division: GAMS models will continue to work in the same cross-platform way for both core and peripheral platforms.
- We will drop support for x86-64 Solaris with GAMS 25.0.
- We may increase the minimal required GLIBC version on Linux to 2.12 with GAMS 25.0.
- We may increase the minimal required MacOS X version to 10.11 with GAMS 25.0.

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. This feature is available on Linux, MacOS X, and Windows only.

Table of Contents

Acknowledgments
Platforms
GAMS System
GAMS
Special Functions
Documentation
Solvers
BARON
CBC
Conopt
CONVERT
CPLEX
DECIS
GAMSCHK
GUROBI
GUSS/Scenario Solver
LocalSolver
MINOS
MOSEK
MPSGE
SCIP
SolveEngine
SoPlex
Tools
CSV2GDX

➢ https://www.gams.com/24.9/docs/releasenotes/24.9.html

Why GAMS?

- Algebraic modeling technology for complex, large-scale optimization.
- Uniform interface to all major solvers
- Design principles: Simple, but powerful modeling language, open architecture, independent layers
- More than 25 years of R&D and user feedback
- Open for new developments
- Protecting investments of users

GAMS Talks at OR2017

- Embedded Code in GAMS - Using Python as an Example
  Given by: Lutz Westermann
  Wednesday (Sep. 06), 13:30-15:00 [WC-02] / WGS|102

- High Performance Computing with GAMS
  Given by: Fred Fiand, Michael Bussieck
  Thursday (Sep. 07), 11:00-12:30 [TB-02] / WGS|102

- A distributed Optimization Bot/Agent Application Framework for GAMS Models
  Given by: Franz Nelissen
  Thursday (Sep. 07), 14:45-16:15 [TD-02] / WGS|102

- Exam scheduling at United States Military Academy West Point
  Given by: Frederik Proske, Robin Schuchmann
  Friday (Sep. 08), 09:00-10:30 [FA-02] / WGS|102

Thank You

Meet us at the GAMS booth!

Pre-Conference Workshops

Fred Fiand
Franz Nelissen
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

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 to 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 63
  - How many newspapers should he sell: S 63

- Derived Outcomes:
  - How many newspapers need to be disposed: I 0
  - How many customers are lost: L 0

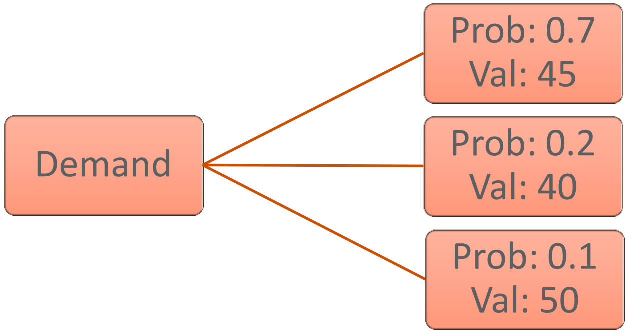
Simple NB Problem – GAMS Formulation

```
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 customers are lost after the outcome becomes known?
  - How many unsold models go to the inventory?
    → ***Second-stage* or *recourse decision***
  - 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.

parameter demand | probability | Value of d

```
randvar d discrete 0.7  45
                   0.2  40
                   0.1  50
stage 2 I L S d
stage 2 Row1 Row2
```

Make demand d uncertain

Define (non-default) stage 2 variables and equations

Stochastic NB Problem – GAMS Extension

```
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, e.g. [...]\225a\empinfo.dat

EMP, what?
→ Excursus

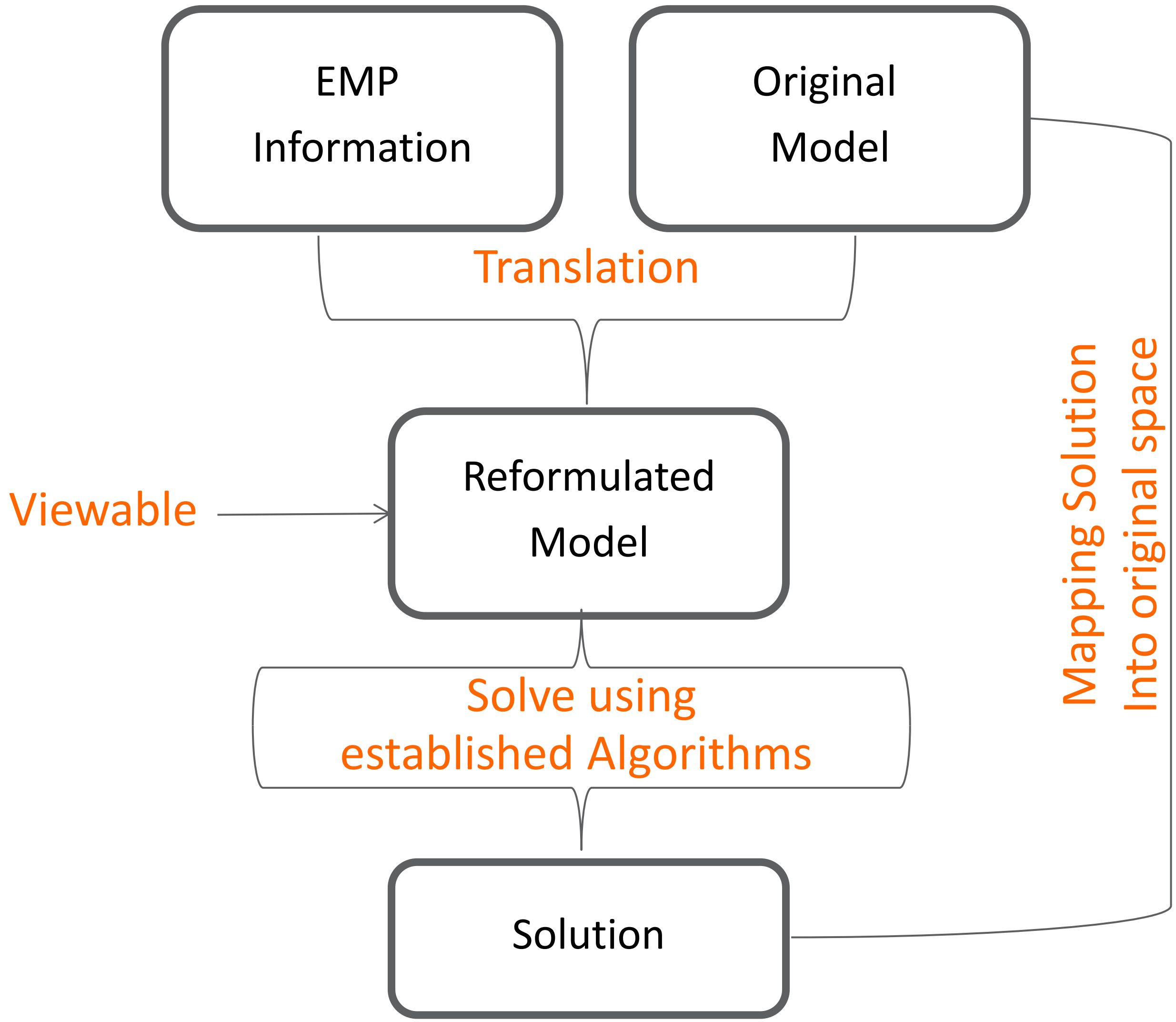
Excursus: EMP, what?

EMP stands for **Extended Mathematical Programming**

Idea:

- Use existing language features to specify additional model features, structure, and semantics
- Express extended model in symbolic (source) form and apply existing modeling/solution technology

The EMP Framework



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:

```
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

→ nbsimple.gms

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`]

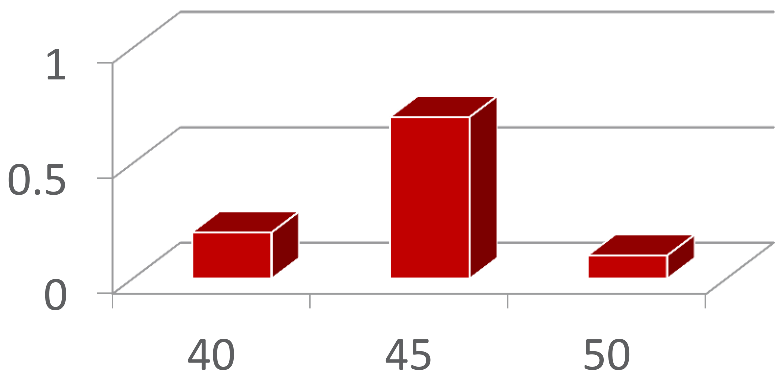
- Defines the stage of random variables (`rv`), equations (`equ`) and variables (`var`):

```
stage stageNo rv | equ | var {rv | equ | var}
```

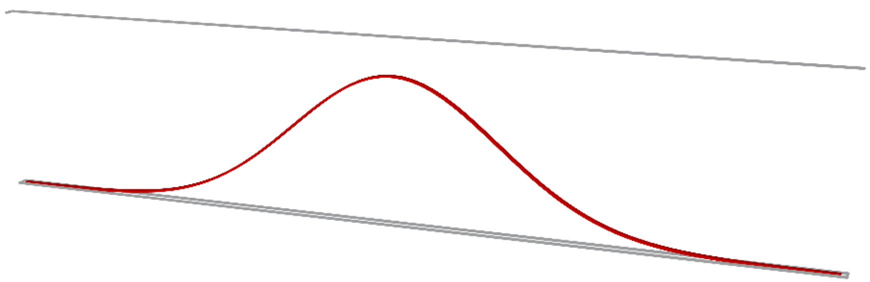
- `StageNo` defines the stage number
- The default `StageNo` for the objective variable and objective equation is the highest stage mentioned
- The default `StageNo` for all the other random variables, equations and variables not mentioned is 1

Random Variables

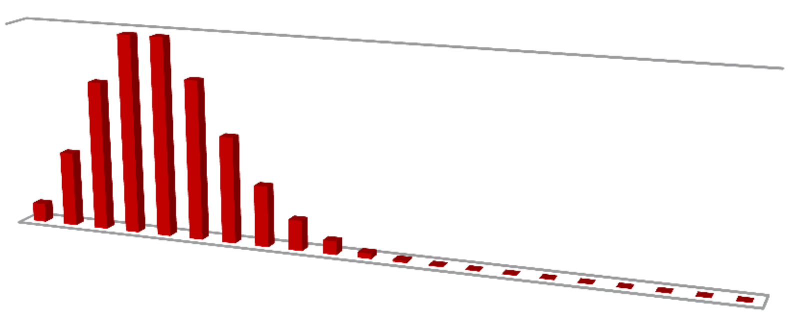
Discrete Distribution



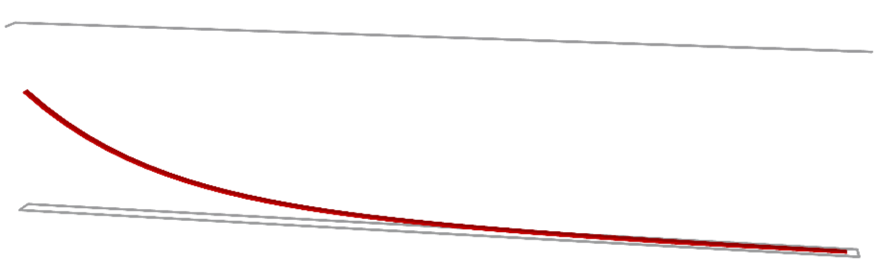
Normal Distribution



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`.

→ nbdiscindep.gms

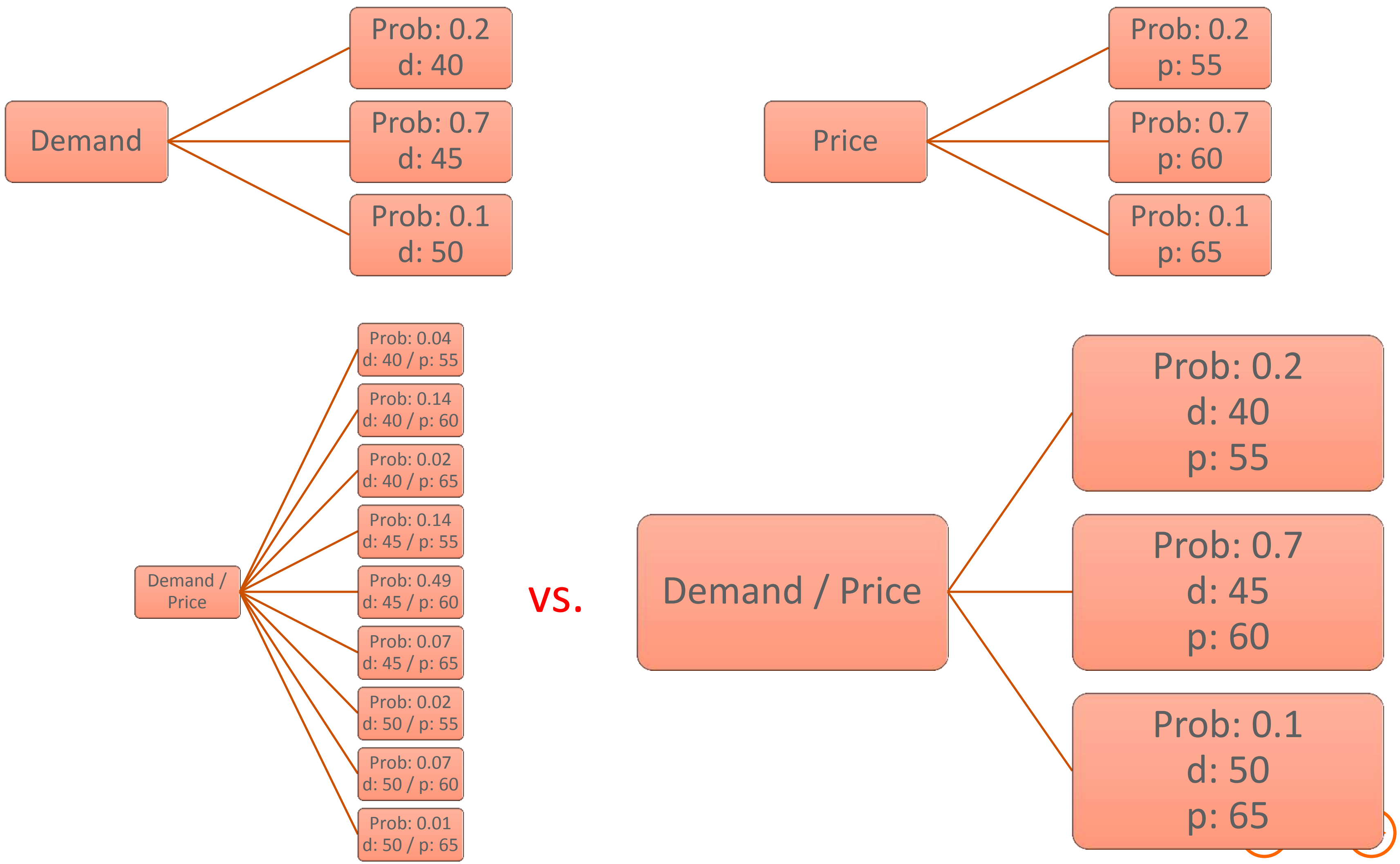
```
randVar rv distr par {par}
```

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.

→ nbcontindep.gms

Independent vs. Joint Random Variables



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 `prob` and the corresponding realization for each random variable by `val`

→ nbdiscjoint.gms

Correlation between RVs [`correlation`]

- Defines a correlation between a pair of random variables:

```
correlation rv rv val
```

- `rv` is a random variable which needs to be specified using the `randvar` keyword and `val` defines the desired correlation (-1 ≤ `val` ≤ 1).

→ nbcontjoint.gms

Chance Constraints

```
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
```

GAMS

Chance Constraints

3 out of 4 must be true

[0.75 ≥ 0.6]

```
• 1*X1 + X2 =g= 7;
• 2*X1 + X2 =g= 7;
• 3*X1 + X2 =g= 7;
• 4*X1 + X2 =g= 7;
```

2 out of 3 must be true

[0.66 ≥ 0.6]

```
• 1*X1 + 3*X2 =g= 12;
• 2*X1 + 3*X2 =g= 12;
• 3*X1 + 3*X2 =g= 12;
```

Chance Constraints [chance]

- Defines individual or joint chance constraints (CC):

```
chance equ {equ} [holds] minRatio [weight|varName]
```

- Individual CC: A single constraint `equ` has to hold for a certain ratio (0 ≤ `minRatio` ≤ 1) of the possible outcomes
- Joint CC: A set of constraints `equ` has to hold for a certain ratio (0 ≤ `minRatio` ≤ 1) of the possible outcomes
- If `weight` is defined, the violation of a CC gets penalized in the objective (weight violationRatio)
- If `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!

OR 2017 BERLIN

GAMS

Pre-Conference Workshops

Fred Fiand
Franz Nelissen
Lutz Westermann

Outline

Part I: An Introduction to GAMS

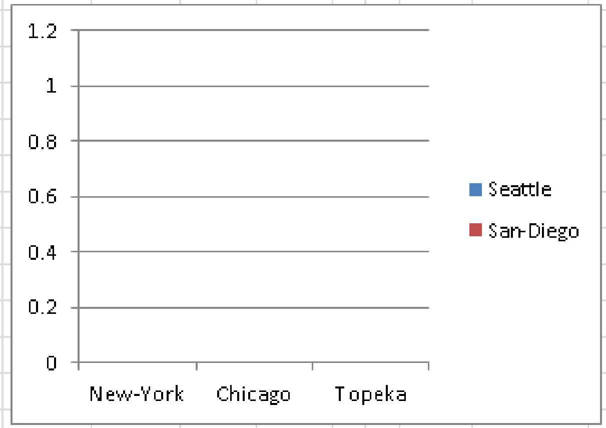
Part II: Stochastic programming in GAMS

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

Part IV: Code embedding in GAMS

Excel and GAMS

	A	B	C	D	E	F
1	**Distance**	New-York	Chicago	Topeka		Supply
2	Seattle	2.5	1.7	1.8		350
3	San-Diego	2.5	1.8	1.4		600
4						
5	Demand	325	300	275		
6						
7	Freight cost	90				
8						
9						
10	**SHIPMENT**	New-York	Chicago	Topeka		
11	Seattle					
12	San-Diego					
13						
14						
15						
16						
17						
18	**GAMS Directory:**	C:\GAMS\win64\24.8.5\				
19	**Working Directory:**	c:\tmp\				
20	**Solver:**	CPLEX				



Clear Solution

SOLVE LP

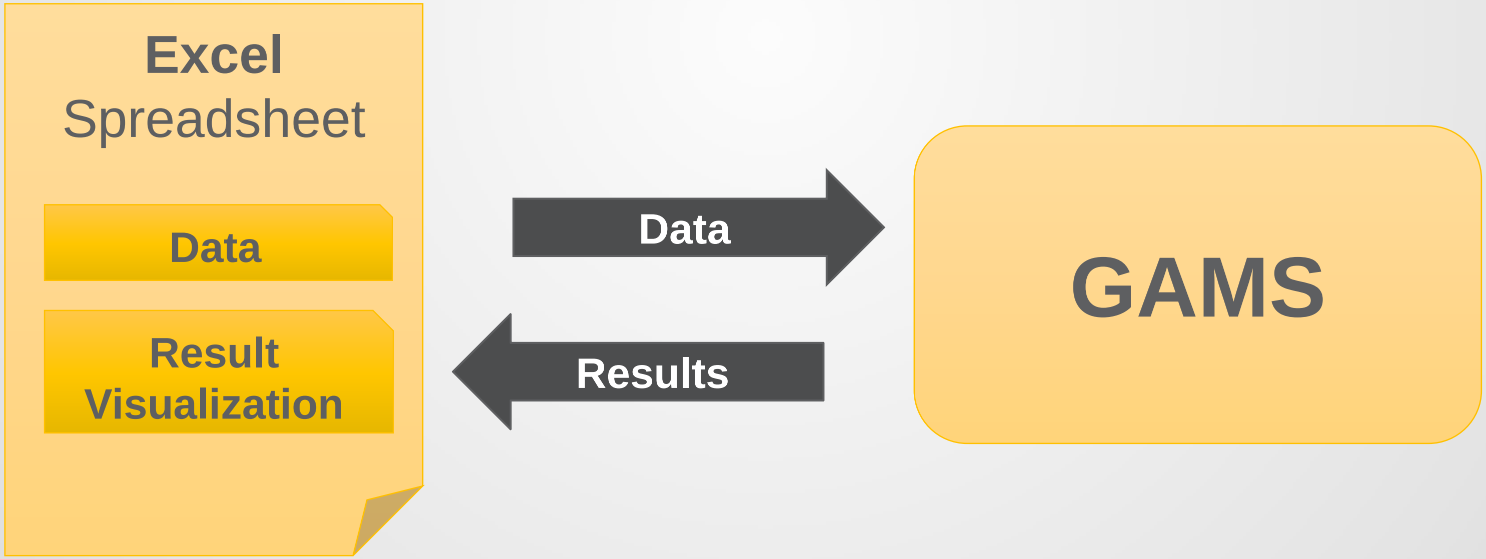
SOLVE MIP

Main | Model | Listing

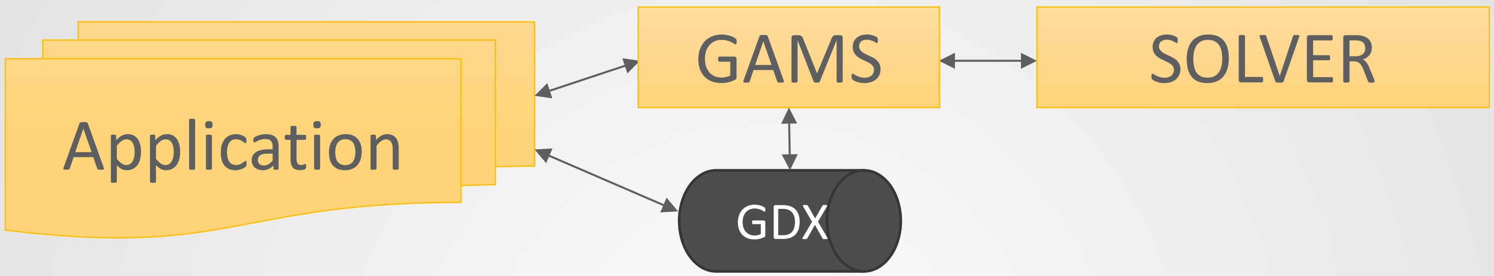
Hands-On
TransXLS

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

- Prepare input data and retrieve results in a convenient way → *GAMSDatabase*
- Control GAMS execution → *GAMSJob*
- Scenario Solving: Feature to solve multiple very similar models in a dynamic and efficient way →*GAMSModelInstance*
- Seamless integration of GAMS into other programming environments
- .NET, C++, Java and Python APIs are part of the current GAMS release available at *www.gams.com*. Many examples available:
  - Sequence of Transport examples (Tutorial)
  - Cutstock, Warehouse, Benders Decomposition, …

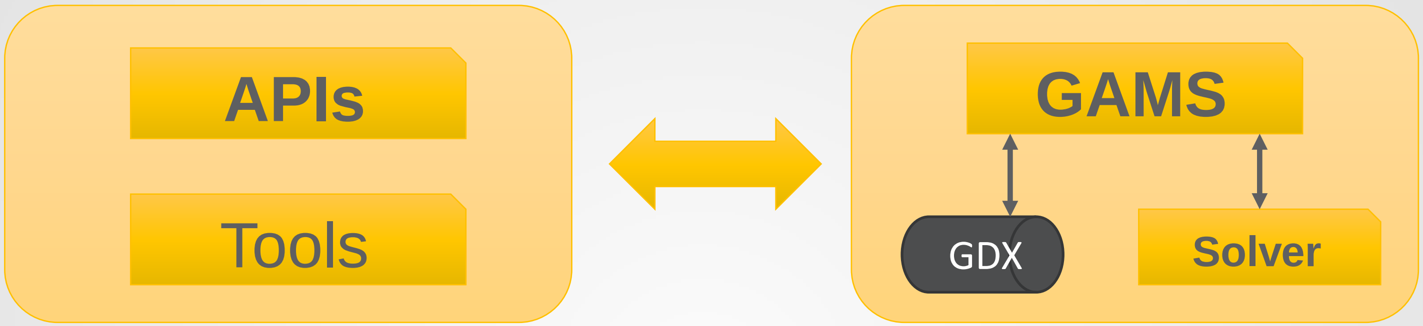
Small Example – C#

```
using System;
using GAMS;

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

      t1.Run();
      foreach (GAMSVariableRecord 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);
      }
    }
  }
}
```

Hands-On
TransportSeq.sln

Seamless Integration

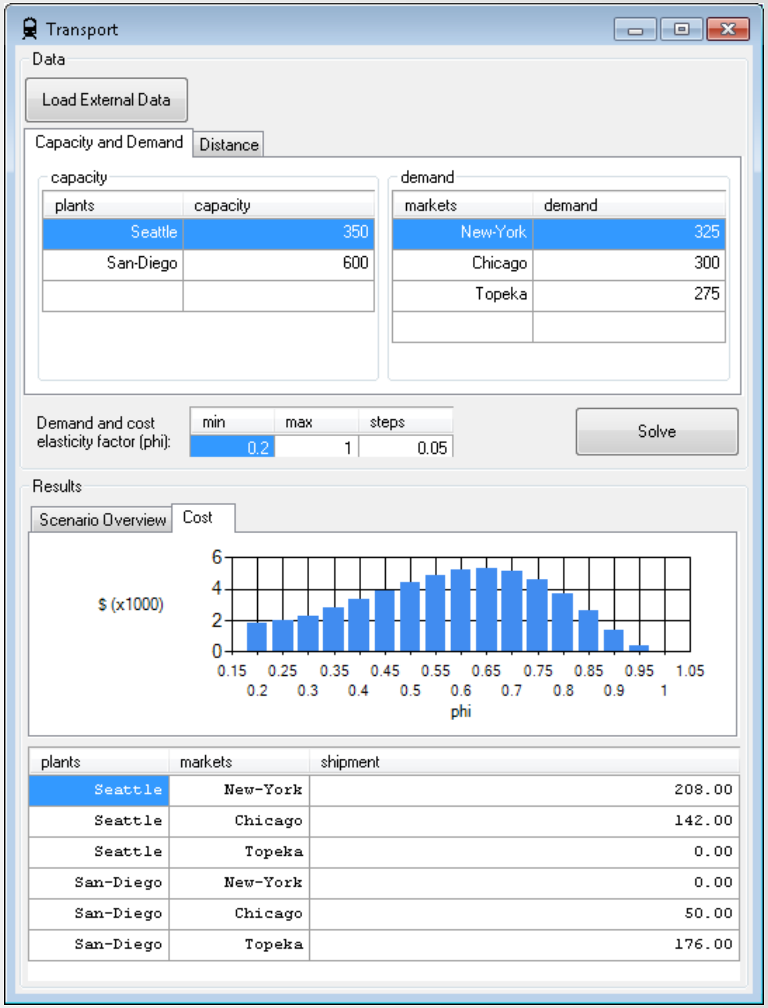
- GAMS concept: Separation of tasks



- 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



Hands-On
TransportGUI.sln

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

```
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

Scenario Solving – GAMSModelInstance

```
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

GAMSModelInstance etc.

`GAMSJob`

- Manages the execution of a GAMS program given by GAMS model source

creates

`GAMSCheckpoint`

- Captures the state of a `GAMSJob`

initializes

`GAMSModelInstance`

- A single mathematical model generated by a GAMS solve statement

modifies

`GAMSModifier`

- Marks elements of a `GAMSModelInstance` to be modifiable

GAMSModelInstance – Example

- *bmult* is one parameter of the model which gets modified before we solve the instance:

```
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);
}
```

Hands-On
Transport7

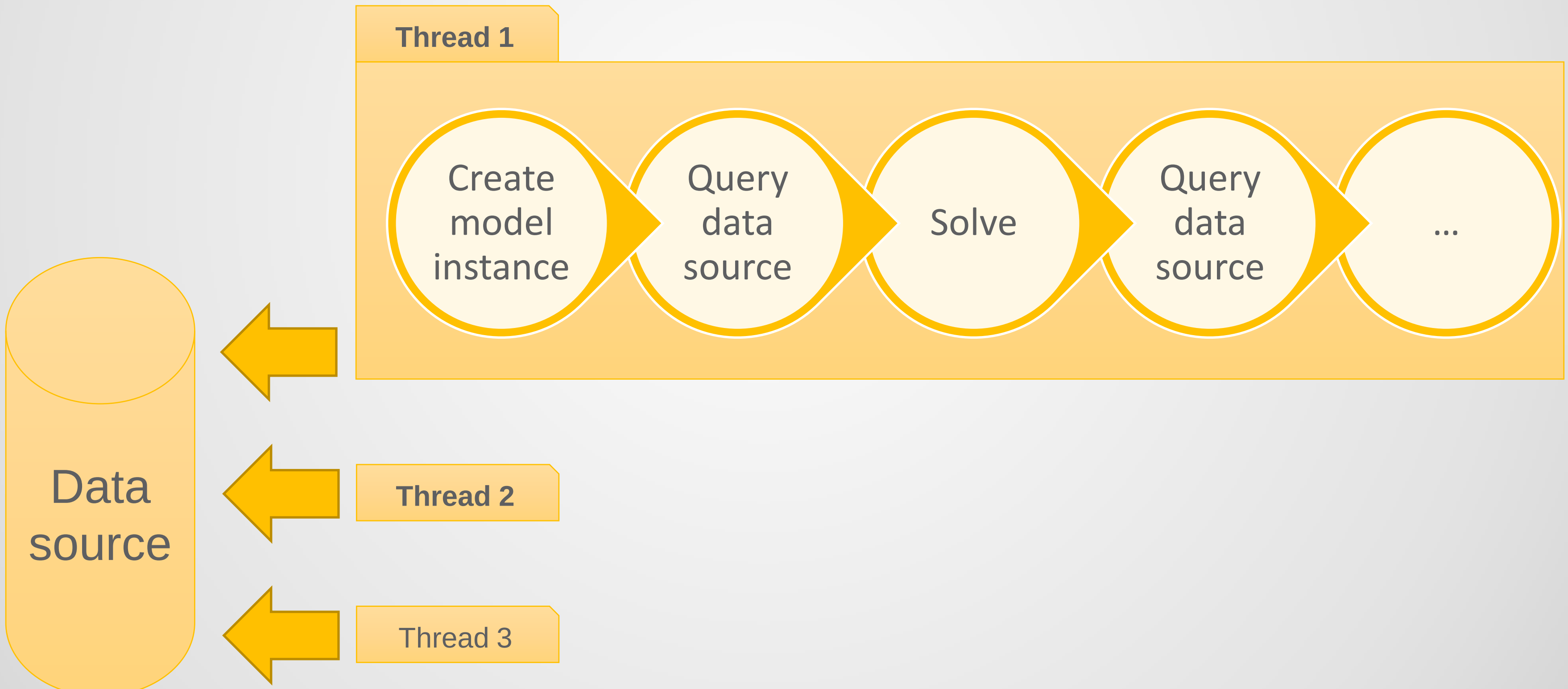
GAMSModelInstance – Example

- Updating bounds of a variable:

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

Hands-On
Transport8

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

Fred Fiand
Franz Nelissen
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) = 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*

Split Example – Data

```
Set cc      / "France - Paris",    "France - Lille",
              "France - Toulouse",
              "Spain - Madrid",    "Spain - Cordoba",
              "Spain - Seville",   "Spain - Bilbao",
              "USA - Washington DC",
              "USA - Houston",     "USA - New York",
              "Germany - Berlin",
              "Germany - Munich", "Germany - Bonn" /
    country / system.empty /
    city    / system.empty /
    mccCountry(cc,country)
    mccCity   (cc,city);
```

Hands-On
embeddedSplit

Split Example – Embedded Code

```
$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) = uniformInt(1,10*card(i));

embeddedCode 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)
endEmbeddedCode aIndex
```

Hands-On
embeddedSort

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

```
$onEmbeddedCode Python: 10
  f = open('i.txt', 'w')
  for i in range(int(gams.arguments)):
    f.write('i'+str(i+1)+'\n')
  f.close()
$offEmbeddedCode

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

```
----      21 SET i

i1 ,    i2 ,    i3 ,    i4 ,    i5 ,    
i6 ,    i7 ,    i8 ,    i9 ,    i10
```

Hands-On
exchangeViaFiles

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
```

Hands-On
exchangeViaEnvVars

Multiple Independent Python Sessions

```
$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
embeddedMultiInstance

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); Alias (i,ii);
Parameter c(i,i); c(i,ii) = uniform(-50,50);

Set    iter                / 1*100 /;
Scalar tcost, minTCost / +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)
  endEmbeddedCode p
  tcost = sum(p, c(p));
  if (tcost < minTCost, minTCost = tcost);
);
Display minTCost;
```

EXECUTION TIME = 16.375 SECONDS

Hands-On
performance

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
pauseEmbeddedCode


Set    iter             / 1*1000 /;
Scalar tcost, minTCost / +inf   /;
loop(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); Alias (i,ii);
Parameter c(i,i); c(i,ii) = uniform(-50,50);

embeddedCode Python:
  import random
  i = list(gams.get("i"))
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 = 1.593 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:

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

Syntax: GAMS

Execution Time:

```
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)

Syntax: Python

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

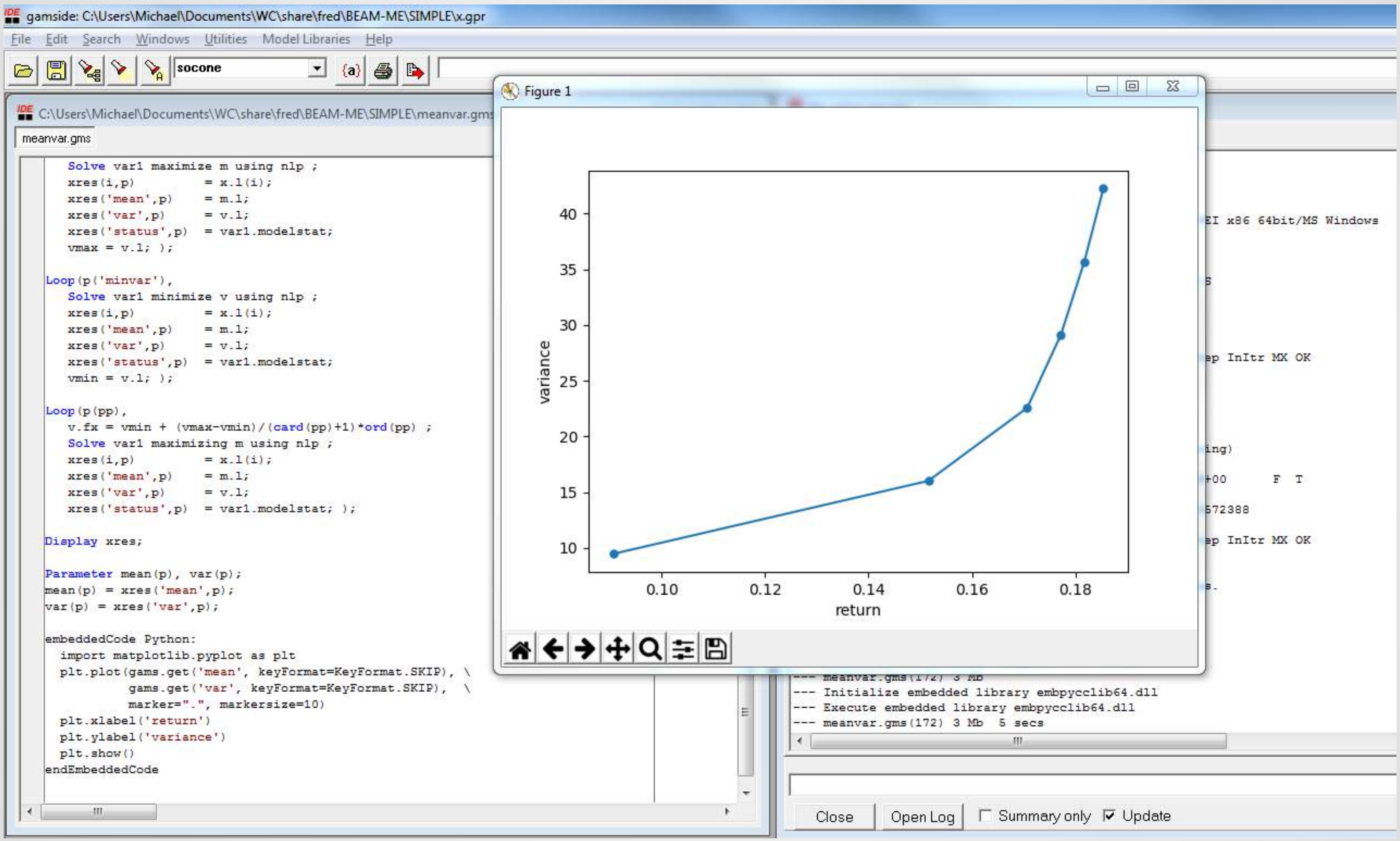
Syntax: Python

- `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!