An Introduction

Lutz Westermann, GAMS Software GmbH
Company Background

Roots: World Bank, 1976

GAMS Development Corporation (Washington)

Tool Provider: General Algebraic Modeling System

Went commercial in 1987

GAMS Software GmbH (Cologne, Braunschweig) 1996
**Agenda**

**What is GAMS?**
- Algebraic Modeling Languages – A Success Story
- GAMS – Highlights and Design Principles
- Striving for Innovation and Compatibility
  - New Modeling and Solution Concepts
  - Software Quality Assurance

**A Simple Example**
1976 - A World Bank Slide

IDEAL TECHNOLOGY

REAL WORLD PROBLEM

ANALYST

GENERAL ALGEBRAIC MODELING SYSTEM

DATA ↔ MODEL ↔ SOLUTION

Operating Systems
Computer Languages
Solution Packages

RESULT: - Limited drain of resources
- Same representation of models for humans and machines
- Model representation is also model documentation

The Vision
GENERAL ALGEBRAIC MODELING SYSTEM

2012 INFORMS Impact Prize

Originators of Algebraic Modeling Languages

36 Years later
Algebraic Modeling Languages (AML)

1. High-level computer programming languages
   - Formulation of mathematical optimization problems
   - Notation similar to algebraic notation

2. Do not solve problems directly, but offer links to state-of-the-art algorithms ("solver-links")

Source: http://en.wikipedia.org/wiki/Algebraic_modeling_language
Impact of Algebraic Modeling Languages

1. Simplified model development, changes, and transfer
2. Added value to existing applications
3. Increased productivity, quality, reliability and maintainability
4. Made a scarce resource (good modelers) more productive
5. Important vehicle to make mathematical optimization available to a broader audience, e.g. domain specific experts
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. ...

Why is GAMS a tool for him?
General Algebraic Modeling System (GAMS)

Broad User Community and Network

GAMS used in more than 120 countries

25+ Years
GAMS Development
GENERAL ALGEBRAIC MODELING SYSTEM

Broad User Community and Network

More than 10,000 licenses

50% academic users, 50% commercial users

25+ Years GAMS Development

6,000+ monthly downloads of the free system
## 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>

### 25+ Years
GAMS Development
GENERAL ALGEBRAIC MODELING SYSTEM

Strong Development Environment

GAMS IDE

- Project management
- Editor / Syntax coloring / Spell checks
- Listing file / Tree view / Syntax-error navigation
- Model Debugging / Profiling
- Solver selection / Option selection
- Data viewer (GDX)
  - Export
  - Charting
- GAMS Processes Control
- Model Libraries
Free Model Libraries

- GAMS Model Library
- GAMS Test Library
- GAMS Data Utilities Models
- GAMS EMP Library
- Practical Financial Optimization Models

More than 1250 models!
Design Principles

1. Simple modeling language with a balanced mix of declarative and procedural elements

2. Open architecture and interfaces to other systems, independent layers
Simple Declarative Language

1. Few basic language elements: sets, parameters, variables, equations, models
2. Language similar to mathematical notation
3. Easy to learn
4. Model is executable description of the problem
5. Lot’s of code optimization under the hood
Mix of Declarative and Procedural Elements

Procedural elements like loops, for, if, macros and functions:

- Allow to build complex problem algorithms within GAMS
- Interaction with other systems:
  - Job control
  - Data exchange
Independence of Model and Operating System

Platforms supported by GAMS:

Models can be moved between platforms with ease!
Independence of Model and Solver

One environment for a wide range of model types and solvers

All major commercial LP/MIP solver

Open Source Solver (COIN)

Also solver for NLP, MINLP, global, and stochastic optimization

Switching between solvers with one line of code!
Independence of Model and Data

- Declarative Modeling
- ASCII: Initial model development

- GDX: Data layer ("contract") between GAMS and applications
  - Platform independent
  - No license required
  - Direct GDX interfaces and general API
  - ...

Diagram:
- Application → GDX → GAMS
- Application → GDX → SOLVER
Independence of Model and User Interface

**API’s**

- **Low Level**
- **Object Oriented**: .Net, Java, Python
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model
Simple Encapsulation of a GAMS Model

Very simple interface:

- Properties to communicate input data and results
- Properties to change options like the solver to use
- Run() method to run the model
Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

MS Excel

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Coal</td>
<td>-17.62</td>
<td>-35.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WasteHeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Steam</td>
<td>5.5</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Exhaust</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Heat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Electricity</td>
<td>-0.25</td>
<td>-0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution - Generated Heat & Electricity
Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

**Figure 1**: US dollar short rate scenarios

**Figure 2**: Short vs. long rates
Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

```r
v <- rgdx(fnSol, list(name = "tour", form = "sparse"))
nxt <- v$val[, 2]
# compute the sequence of cities, based on nxt
solSeq <- NA * c(1:n + 1)
k <- 1
for (j in c(1:n)) {
  solSeq[j] <- k <- nxt[k]
}
solSeq[n + 1] <-
if (k != 1) stop;
loc <- cmdscale(x =
rx <- range(x =
ry <- range(y =
tspres <- loc[
seq(s) -
```
Striving for **Innovation and Compatibility**

Models 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
- Software Quality Assurance
New Modeling and Solution Concepts

Examples:
- Disjunctive Programs
- Bilevel Programs
- Extended Nonlinear Programs
- Stochastic Programming
- ...

Issues:
- Breakouts of traditional Mathematical Programming classes
- No conventional syntax
- Limited support with common model representation
- Incomplete/experimental solution approaches
- Lack of reliable/any software

GAMS is conservative when it comes to syntax extensions
The “GAMS” – Approach

Extended Mathematical Programming

- **Experimental framework** for automated mathematical programming **reformulations**
- Keep the language simple: Do **not overload** existing GAMS notation
- Use **existing language features** to specify additional model features, structure, and semantics
- Express **extended model information** in **symbolic (source) form** and apply existing modeling/solution technology
- Package new tools with the production system
EMP/SP

• Simple interface to add uncertainty to existing deterministic models
• (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, ...
• Available solution methods:
  – Automatic generation of Deterministic Equivalent (can be solved with any solver)
  – Specialized commercial algorithms (DECIS, LINDO)
Perspectives:
- What is the impact of new features?
- What is the impact of updated or new solvers?
- Is the new distribution backward compatible?
- ...
Quality Test Models Library

- Tests to verify proper behavior of the system
- More than 600 quality test models, each containing numerous pass/fail tests
- Automatically executed every night for all solver combinations: \( \rightarrow \) 12,000 runs / platform (all tests)
- Automatically generated test summaries with different level of information
- Assurance about the basic functionality of the software

Latest GAMS System Builds and Test Results

<table>
<thead>
<tr>
<th>nightly a</th>
<th>System</th>
<th>Libraries</th>
<th>Build</th>
<th>Rev</th>
<th>Status and Time (UTC)</th>
<th>Initial Tests</th>
<th>Full Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friday</td>
<td>inx</td>
<td>Download</td>
<td>24.3.0</td>
<td>45152</td>
<td>Test done 22Mar2014 12:42:54</td>
<td>805 runs 0 failures (q=0,s=0)</td>
<td>Report 11780 runs 0 failures (q=0,s=0)</td>
</tr>
<tr>
<td>Saturday</td>
<td>leg</td>
<td>Download</td>
<td>24.3.0</td>
<td>45152</td>
<td>Test done 22Mar2014 22:03:49</td>
<td>804 runs 0 failures (q=0,s=0)</td>
<td>Report 2198 runs 0 failures (q=0,s=0)</td>
</tr>
<tr>
<td>Friday</td>
<td>vs8</td>
<td>Download</td>
<td>24.3.0</td>
<td>45152</td>
<td>Test started 22Mar2014 01:12:52</td>
<td>947 runs 0 failures (q=0,s=0)</td>
<td>Report results pending</td>
</tr>
<tr>
<td>Friday</td>
<td>wej</td>
<td>Download</td>
<td>24.3.0</td>
<td>45152</td>
<td>Test started 22Mar2014 03:47:14</td>
<td>944 runs 1 failures (q=1,s=0)</td>
<td>Report results pending</td>
</tr>
</tbody>
</table>

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!
Why GAMS?

- Experience of 25+ years
- Broad user community from different areas
- Lots of model templates
- Strong development interface

- Consistent implementation of design principles
  - Simple, but powerful modeling language
  - Independent layers
  - Open architecture: Designed to interact with other applications

- Open for new developments
- Protecting investments of users
Agenda

What is GAMS?

A Simple Example
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 how easy it is to change model type and, consequently, solver technology
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

- **SP**
  - Allows uncertainty,
    - e.g. uncertain demand
A Simple **Transportation Problem**

**Canning Plants** (supply) \(\rightarrow\) shipment \(\rightarrow\) **Markets** (demand)

(Number of cases)

- **Seattle** (350)
- **Topeka** (275)
- **Chicago** (300)
- **New York** (325)
- **San Diego** (600)

Freight: $90 case / thousand miles
A Transportation Model

Minimize subject to

Transportation cost
Demand satisfaction at markets
Supply constraints
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)

\[
\begin{align*}
\text{min} & \quad \sum_i \sum_j c_{ij} \cdot x_{ij} \\
\text{subject to} & \quad \sum_j x_{ij} \leq a_i \quad \forall i \quad \text{(Shipments from each plant \(\leq\) supply capacity)} \\
& \quad \sum_i x_{ij} \geq b_j \quad \forall j \quad \text{(Shipments to each market \(\geq\) demand)} \\
& \quad x_{ij} \geq 0 \quad \forall i, j \quad \text{(Do not ship from market to plant)} \\
& \quad i, j \in \mathbb{N}
\end{align*}
\]
GAMS Algebra (LP Model)

Variables
- \( x(i,j) \): shipment quantities in cases
- \( z \): total transportation costs in thousands of dollars

Positive Variable \( x \); 

Equations
- \( \text{cost} \): define objective function
- \( \text{supply}(i) \): observe supply limit at plant \( i \)
- \( \text{demand}(j) \): satisfy demand at market \( j \);

\[
\begin{align*}
\text{cost} & \quad \Rightarrow \quad z = \sum_{(i,j)} c(i,j) x(i,j) \\
\text{supply}(i) & \quad \Rightarrow \quad \sum_{j} x(i,j) = a(i) \\
\text{demand}(j) & \quad \Rightarrow \quad \sum_{i} x(i,j) = b(j)
\end{align*}
\]

Model \( \text{modelLP} \) /cost, supply, demand/;

Solve \( \text{modelLP} \) using \( \text{lp} \) minimizing \( z \);
Sets
- i canning plants / seattle, san-diego /
- j markets / new-york, chicago, topeka /

Parameters
- a(i) capacity of plant i in cases
  - seattle 350
  - san-diego 600
- b(j) demand at market j in cases
  - new-york 325
  - chicago 300
  - topeka 275
- Table d(i,j) distance in thousands of miles
<table>
<thead>
<tr>
<th></th>
<th>new-york</th>
<th>chicago</th>
<th>topeka</th>
</tr>
</thead>
<tbody>
<tr>
<td>seattle</td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
</tr>
<tr>
<td>san-diego</td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
</tr>
</tbody>
</table>

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

Parameter c(i,j) transport cost in thousands of dollars per case ;
- c(i,j) = f * d(i,j) / 1000 ;
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 

Parameter c(i,j) transport cost in thousands of dollars per case 

$call gams data.gms gdx=data
$if errorlevel 1 $abort Error preparing Data
$gdxin data.gdx
$load i j a b d c

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 

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

$load i<d.dim1 j<d.dim2 dab
Scalar f freight in dollars per case per thousand:
Parameter c(i,j) transport cost in thousands of:
c(i,j) = f * d(i,j) / 1000:

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

Positive Variable x:

Equations
cost define objective function
supply(i) observe supply limit at plant i

$.call gdxxr-w data.xlsx par=d rng=A1 par=a rng=G2 dim=1 rdim=1 par=b rng=B6 dim=1
$.if errorlevel 1 $abort Error preparing Data
$.gdxin data.gdx
$.load icd.dim1 jxjdim2 d a b

Excel Sheet

<table>
<thead>
<tr>
<th></th>
<th>New-York</th>
<th>Chicago</th>
<th>Topeka</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
</tr>
<tr>
<td>3</td>
<td>San-Diego</td>
<td>2.5</td>
<td>1.8</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>New-York</td>
<td>325</td>
<td>300</td>
</tr>
</tbody>
</table>
Solution to LP model

Canning Plants (supply) shipments Markets (demand)

Seattle (350) Topeka (275) Chicago (300) New York (325)
San Diego (600)

Freight: $90 case / thousand miles Total cost: $153,675
Minimum shipment of 100 cases

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

Cost ($)

\[
\begin{align*}
\text{ship} &= 0 \\
\text{ship} &= 1
\end{align*}
\]

\( x \) (Number of cases)

add constraints:

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

\( \text{ship}_{i,j} \in \{0, 1\} \)
Scalar \( \text{min} \) minimum shipment volume (number of cases) /100/  
\( \text{bigm} \) big-M relaxation ;  

\( \text{bigm} = \min \{ \text{smax} (i,a(i)), \text{smax} (j,b(j)) \} \);  

Binary variables \( \text{ship}(i,j) \);  

Equations  
\( \text{minship}(i,j) \) minimum shipment  
\( \text{maxship}(i,j) \) maximum shipment :  

\( \text{minship}(i,j) \) \[ x(i,j) = \text{min} \ast \text{ship}(i,j) ; \]  
\( \text{maxship}(i,j) \) \[ x(i,j) = \text{bigm} \ast \text{ship}(i,j) ; \]  

Model modelMIP /modelLP, minship, maxship/ ;  

Solve modelMIP using mip minimizing \( z \) ;
Solution to MIP model

Canning Plants (supply) shipments Markets (demand)

Seattle (350)  
San Diego (600)  
Topeka (275)  
Chicago (300)  
New York (325)  

Freight: $90 case / thousand miles  
Total cost: $153,675
Cost Savings (non-linear)

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

The cost per case decreases with a increasing shipment volume

Cost Savings (non-linear)
Scalar beta beta factor / 0.95 /;

cost ..    z  =e=  sum((i,j), c(i,j)*x(i,j)**beta) ;
supply(i) .. sum(j, x(i,j)) =l=  a(i) ;
demand(j) .. sum(i, x(i,j)) =g=  b(j) ;
minship(i,j) .. x(i,j) =g=  mins*ship(i,j) ;
maxship(i,j) .. x(i,j) =l=  bigM*ship(i,j) ;

Model transport /all/ ;
Solve transport using minlp minimizing z ;
Display x.l, x.m ;

1: 1 Modified Insert
Solution to MINLP model

Canning Plants (supply) shipments Markets (demand)

Seattle (350)

Topeka (275)

Chicago (300)

New York (325)

San Diego (600)

Freight: ~$90 case / thousand miles  Total cost: $115,438
Uncertain Demand

- Uncertain demand factor $bf$

- Decisions to make:
  - How many units should be shipped “here and now” (without knowing the outcome of the uncertain demand)?
    $\rightarrow$ First-stage decision
  - What can be done after the outcome becomes known and we did not ship enough?
    $\rightarrow$ Second-stage or recourse decision
  - Recourse decisions can be seen as
    - penalties for bad first-stage decisions
    - variables to keep the problem feasible
**GAMS Algebra (SP Model)**

```plaintext
file emp / 'emp.info' /; put emp ' problem &gams.i/
$onput
randvar bf discrete 0.3 0.9  
  0.5 1.0  
  0.2 1.1
stage 2 bf demand u
$offput
putclose emp;
Set scen scenarios / s1*s3 /;
Parameter
  s_bf(scen) demand factor realization by scenario
  s_x(scen,i,j) shipment per scenario
  s_u(scen,j) cases bought per scenario;
Set dict / scen .scenario,'
  bf .randvar .s_bf
  x .level .s_x
  u .level .s_u /
Solve transport using emp minimizing z scenario dict;
Display s_bf, s_x, s_u;
```
Solution to SP model

Canning Plants (supply) \[\rightarrow\] shipments \[\rightarrow\] Markets (demand)

Seattle (350)
San Diego (600)
Topeka (~275)
Chicago (~300)
New York (~325)

Freight: $90 case / thousand miles
Total cost: $158,588
Stochastic Programming in GAMS

- The Extended Mathematical Programming (EMP) framework is 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 or create Deterministic Equivalent (new free solver DE)

Outlook: Running GAMS in an Application

<table>
<thead>
<tr>
<th>Distance</th>
<th>New-York</th>
<th>Chicago</th>
<th>Topeka</th>
<th>Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>2.5</td>
<td>1.7</td>
<td>1.8</td>
<td>350</td>
</tr>
<tr>
<td>San-Diego</td>
<td>2.5</td>
<td>1.8</td>
<td>1.4</td>
<td>600</td>
</tr>
<tr>
<td>Demand</td>
<td>325</td>
<td>300</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>Freight cost</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SHIPMENT</th>
<th>New-York</th>
<th>Chicago</th>
<th>Topeka</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>50</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>San-Diego</td>
<td>275</td>
<td>0</td>
<td>275</td>
</tr>
</tbody>
</table>

GAMS Directory: `c:\program files\GAMS23.2\`
Working Directory: `c:\tmp\`
Solver: CPLEX

Solve LP
Solve MIP

Clear Solution

Solver: CPLEX
Equations: 6 Variables: 7
Model Status: 1 Optimal
Solver Status: 1 Normal Completion
Iterations: 4 Solve Time: 0.00
Objective Value: 153.675

Hands-On
Thank You

Europe
GAMS Software GmbH
P.O. Box 40 59
50216 Frechen, Germany
Phone: +49 221 949 9170
Fax: +49 221 949 9171
info@gams.de

USA
GAMS Development Corp. 1217 Potomac Street, NW Washington, DC 20007
USA
Phone: +1 202 342 0180
Fax: +1 202 342 0181
sales@gams.com