GAMS – An Introduction

Get ready to learn the basics of GAMS

Frederik Fiand & Lutz Westermann
GAMS Software GmbH
Agenda

GAMS at a Glance

GAMS - Hands On Examples

Outlook 1 - Deployment of GAMS models

Outlook 2 - Solving Multiple Models Efficiently
GAMS at a Glance
1976 - A World Bank Slides

The Vision

GAMS came into being!

RESULT:
- Limited drain of resources
- Same representation of models for humans and machines
- Model representation is also model documentation
The aim of this system is to provide one representation of a model which is easily understood by both humans and machines.


Self-documenting model. A GAMS model is a machine-executable documentation of an optimization problem.

Company

- Roots: World Bank, 1976
- Went commercial in 1987

Locations
- GAMS Development Corporation (Washington)
- GAMS Software GmbH (Germany)

Product
- The General Algebraic Modeling System
What did this give us?

- Simplified model development & maintenance
- Increased productivity tremendously
- Made mathematical optimization available to a broader audience (domain experts)

2012 INFORMS Impact Prize
Broad User Community and Network

- 14,000+ licenses
- Users: 50% academic, 50% commercial
- GAMS used in more than 120 countries
- Uniform interface to ~40 solvers

30+ Years GAMS Development
<table>
<thead>
<tr>
<th>Broad Range of Application Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agricultural Economics</td>
</tr>
<tr>
<td>Chemical Engineering</td>
</tr>
<tr>
<td>Econometrics</td>
</tr>
<tr>
<td>Environmental Economics</td>
</tr>
<tr>
<td>Finance</td>
</tr>
<tr>
<td>International Trade</td>
</tr>
<tr>
<td>Macro Economics</td>
</tr>
<tr>
<td>Management Science/OR</td>
</tr>
<tr>
<td>Micro Economics</td>
</tr>
</tbody>
</table>
Declarative and Procedural Language Elements

**Declarative elements**

- Similar to mathematical notation
- Easy to learn - few basic language elements: sets, parameters, variables, equations, models
- Model is executable (algebraic) description of the problem

**Procedural elements**

- Control Flow Statements (e.g. loops, for, if,...),
- Build complex problem algorithms within GAMS
- Simplified interaction with other systems
  - Data exchange
  - APIs
Cross Platform GUI – GAMS Studio

- Open source Qt project (Mac/Linux/Win)
  - Published on GitHub under GPL
- Released in beta status
- Group Explorer
- Editor / Syntax coloring / Spell checks

- Tree view / Syntax-error navigation
- Option Editor
- Integrated Help
- Model Debugging & Profiling
- Solver selection & setup
- Data viewer
- GAMS Processes Control
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
  - ...
Independence of **Model and User Interface**

**API’s**

- *Low Level*
- *Object Oriented*: .Net, Java, Python, C++
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model
Free Model Libraries

More than 1400 models!
Why GAMS?

- Experience of 30+ 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
GAMS – Hands On Examples
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
LP
• Determine minimum transportation cost
• Result: city to city shipment volumes

MIP
• Discrete decisions
• E.g.: Ship at least 100 cases

MINLP
• Non-linearity
• E.g.: Decrease in unit cost with growing volumes

SP
• Uncertainty
• E.g.: Uncertain demand
A Simple Transportation Problem

Canning Plants (supply)  shipments  Markets (demand)

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

Freight: $90 case / thousand miles
A Simple **Transportation Problem**

Minimize Transportation cost
subject to Demand satisfaction at markets
Supply constraints

Supply (cases)
- Seattle: 350
- San Diego: 600

Distance (thousand miles)
- Seattle to Topeka: 1.8
- Seattle to Chicago: 1.7
- Seattle to New York: 2.5
- San Diego to Topeka: 1.4
- San Diego to Chicago: 1.4
- San Diego to New York: 2.5

Demand (cases)
- Topeka: 275
- Chicago: 300
- New York: 325

Freight: $90 case / thousand miles
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)

Objective: Minimize total transportation cost

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

Subject to
1. $\sum_j x_{ij} \leq a_i \quad \forall i$ (Shipments from each plant $\leq$ supply capacity)
2. $\sum_i x_{ij} \geq b_j \quad \forall j$ (Shipments to each market $\geq$ demand)
3. $x_{ij} \geq 0 \quad \forall i, j$ (Do not ship from market to plant)
4. $i, j \in \mathbb{N}$

Indices: $i$ (Canning plants)  
$\quad j$ (Markets)  
Decision variables: $x_{ij}$ (Number of cases to ship)  
Data:  
$\quad c_{ij}$ (Transport cost per case)  
$\quad a_i$ (Capacity in cases)  
$\quad b_i$ (Demand in cases)

Objective: Minimize total transportation cost

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

Subject to
1. $\sum_j x_{ij} \leq a_i \quad \forall i$ (Shipments from each plant $\leq$ supply capacity)
2. $\sum_i x_{ij} \geq b_j \quad \forall j$ (Shipments to each market $\geq$ demand)
3. $x_{ij} \geq 0 \quad \forall i, j$ (Do not ship from market to plant)
4. $i, j \in \mathbb{N}$
GAMS Syntax (LP Model)

Variables
\[ x(i,j) \text{ shipment quantities in cases} \]
\[ z \text{ total transportation costs in thousands of dollars} \]

Positive Variable \( x \)

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

\[ \text{cost} \Rightarrow z = \sum(i,j) c(i,j) x(i,j) \]
\[ \text{supply}(i) \Rightarrow \sum(j) x(i,j) = a(i) \]
\[ \text{demand}(j) \Rightarrow \sum(i) x(i,j) = b(j) \]

Model transportLP /all/;

Solve transportLP using lp minimizing z;
GAMS Syntax (Data Input)

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

$onecho > instructions.txt
par=d rng=Sheet1!A1 rdim=1 cdim=1
par=b rng=Sheet1!B6 rdim=0 cdim=1
par=a rng=Sheet1!G2 rdim=1 cdim=0
$offecho
$call gdxrwx data.xlsx c=data.gdx @instructions.txt
$if errorlevel 1 $off Error preparing data
$gdxin data.gdx
$load i<d.dim1 j<d.dim2 d a b
$gdxin
Scalar f  freight in dollars per case per thousand miles /90/ ;
c(i,j)  =  f * d(i,j) / 1000 ;
Solution to LP model

Canning Plants (supply) — shipments (Number of cases) — Markets (demand)

Seattle (350) — 0 cases
San Diego (600) — 275 cases
Topeka (275) — 0 cases
Chicago (300) — 300 cases
New York (325) — 50 cases

Freight: $90 case / thousand miles
Total cost: $153,675
LP
- Determine minimum transportation cost
- Result: city to city shipment volumes

MIP
- Discrete decisions
- E.g.: Ship at least 100 cases

MINLP
- Non-linearity
- E.g.: Decrease in unit cost with growing volumes

SP
- Uncertainty
- E.g.: Uncertain demand
MIP Model: Minimum Shipment of 100 cases

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

\[
\begin{align*}
  x_{i,j} &\geq 100 \cdot \text{ship}_{i,j} \quad \forall i,j \\
  x_{i,j} &\leq \text{bigM} \cdot \text{ship}_{i,j} \quad \forall i,j
\end{align*}
\]

- (if ship=1, then ship at least 100)
- (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
MIP Model: Solution

Canning Plants (supply)  shipments (Number of cases)  Markets (demand)

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

Freight: $90 case / thousand miles  Total cost: $153,675
LP  
- Determine minimum transportation cost  
- Result: city to city shipment volumes

MIP  
- Discrete decisions  
- E.g.: Ship at least 100 cases

MINLP  
- Non-linearity  
- E.g.: Decrease in unit cost with growing volumes

SP  
- Uncertainty  
- E.g.: Uncertain demand
MINLP: Cost Savings

The cost per case decreases with a increasing shipment volume

Replace:
\[
\begin{align*}
\min & \quad \sum_i \sum_j c_{ij} \cdot x_{ij} \\
\text{With} & \quad \min \sum_i \sum_j c_{ij} \cdot x_{ij}^{\beta} \\
\end{align*}
\]

(Minimize total transportation cost)
MINLP Model: GAMS Syntax

```
102  * MINLP
103  Scalar  beta / 0.95 /
104  Equation  costnlp define non-linear objective function;
105  costnlp..  z  =e=  sum((i,j), c(i,j)*x(i,j)**beta) ;
106
107  Model  transportMINLP / transportMIP - cost + costnlp /;
108
109  Solve  transportMINLP using MINLP minimizing z ;
110
111  rep(i,j,'MINLP') = x.l(i,j);
112  display rep;
113
114
```

Hands-On
MINLP Model: Results
MINLP Model: Solution

Canning Plants (supply) shipments (Number of cases) Markets (demand)

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

Freight: $90 case / thousand miles  Total cost: $115,438
• Determine minimum transportation cost
• Result: city to city shipment volumes

LP

• Discrete decisions
• E.g.: Ship at least 100 cases

MIP

• Non-linearity
• E.g.: Decrease in unit cost with growing volumes

MINLP

• Uncertainty
• E.g.: Uncertain demand

SP
**Stochastic Programming in GAMS**

**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)
Transport Example – Uncertain Demand

Decisions to make

- First-stage decision: How many units should be shipped “here and now” (without knowing the outcome)
- Second-stage (recourse) decision:
  - How can the model react if we do not ship enough?
  - Penalties for “bad” first-stage decisions, e.g. buy additional cases \( u(j) \) at the demand location:

\[
\text{costsp} \quad : \quad z = e^= \sum_{i,j} c(i,j) x(i,j) + \sum_j 0.3 u(j);
\]

\[
\text{demandsp}(j) \quad : \quad \sum_{i} x(i,j) = g^= \text{bf} b(j) - u(j);
\]
Uncertain Demand: GAMS Algebra

```
* Stochastic Program with uncertain demand
Positive variable u(j) unsatisfied demand;
Scalar    bf demand factor / 1 /;
Equation  costsp    define objective function for SP
          demandsp(j) demand satisfaction in SP;

costsp..    z =e= sum((i,j), c(i,j)*x(i,j)) + sum(j, 0.3*u(j));
demandsp(j). sum(i, x(i,j)) =e= bf*b(j) - u(j);

Model transportSP / costsp, demandsp, supply /;
File emp / 'emp.info' /; put emp:
$put
randvar bf discrete 0.3  0.9
      0.5  1.0
      0.2  1.1
$endput
stage 2 bf u demandsp
$offput
Putclose emp;
Set scen scenarios / s1*s4 /;
Parameter
  s_bf(scen) demand factor for realization by scenario
  s_x(scen,i,j) shipment per scenario
  s_u(scen,j) unsatisfied demand per scenario (bought cases);

Set dict / scen . scenario . ''
  bf . randvar . s_bf
  x . level . s_x
  u . level . s_u /

option emp=lindo;
Solve transportSP min z use emp scenario dict;
```
Uncertain Demand: Results
Stochastic Program: Solution

Canning Plants (supply) \[\xrightarrow{\text{shipments}}\] 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).

• More information: https://www.gams.com/latest/docs/UG_EMP_SP.html
Outlook: Deployment of GAMS models

- APIs – Application Programming Interfaces to GAMS
- Using R/Shiny to deploy and visualize GAMS models in a Web Interface
- Using GAMS Jupyter Notebooks to tell “optimization stories”
Calling GAMS from your Application

Creating Input for GAMS Model
→ Data handling using GDX API

Callout to GAMS
→ GAMS option settings using Option API
→ Starting GAMS using GAMS API

Reading Solution from GAMS Model
→ Data handling using GDX API
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, C++, C#, Delphi, Java, Python, VBA, ...)

• 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
Transport Application GUI Example

- Scenario solves of the transportation problem

- Features:
  - Preparation of input data
  - Loading data from Access file
  - Solving multiple scenarios of a model
  - Displaying results

- Four implementation steps:
  1. Graphical User Interface
  2. Preparation of GAMS model
  3. Implementation of scenario solving using GAMSJob
  4. GAMSModelInstance for performance improvements

Hands-On
From GAMS Model to Visual Web User Interface

Currently under Development
Setting the **Model Input Data**

- Data exchange via local files or database connection
- Visualization and modification of input data with intuitive controls
- From a GAMS model to the first interface within minutes
- Comprehensive configurability
Communication with GAMS

- Start and Stop model execution
- Access to GAMS Log file and Listing file
Inspecting the **Results**

- Extensive visualization options of the model output
- Interactive analysis of the results
- On-the-fly switch between different options to inspect the results
Scenario Management

✓ Solve multiple scenarios or load saved data for comparison
Enhanced Model Deployment in GAMS using R/Shiny

- Application connects Web User Interface with a GAMS model
- User Interface allows
  - Data exchange via local files or database
  - Modification of the input data
  - Extensive visualization options
  - Comparison of different scenarios
  - Multi-user support based on Docker technology
  - User authentication
- Tool with intuitive interface for planners (configuration vs. programming)
- This “product” is currently under development. If you are interested in getting involved, please contact support@gams.com

Model Deployment in GAMS
Wednesday, September 12, 4:10 pm - 5:50 pm
(Session WD-16)
Welcome to Jupyter @ GAMS!

Enter your credentials in order to sign in or contact GAMS Support for further information.

Getting Started
- Introduction
- Milco Example
- PickStock Example
- A GAMS Tutorial by Richard E. Rosenthal

Further Help
- Jupyter Notebook Users Manual (from Bryn Mawr College)
- GAMS World Forum
- Contact GAMS
GAMS Jupyter Example

In [17]: %gams
Parameter fund(date) 'Index fund report parameter'; fund(d) = sum(s, price(d, s)*w1(s));
Parameter error(date) 'Absolute error'; error(d) = abs(index(d)-fund(d));

Plotting of the results

In [18]: %gams pull -d fund error
fig, ax = plt.subplots()
index.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Dow Jones")
fund.plot(y="value", ax=ax, xticks=[0, trainingDays, len(date)], yticks=[], label="Index Fund")

Hands-On
Using GAMS Jupyter Notebooks to tell “optimization stories”

• Runs in a browser/on a server  
  → No local installation needed
• Allows to use notebook technology in combination with GAMS
• Notebooks allow to combine GAMS and Python
  • GAMS works great with well structured data and optimization models
  • Python is very rich in features to retrieve, manipulate, and visualize data that comes in all sort of ways
  • ➔ Combining GAMS and Python in a notebook it is relatively easy to tell an optimization story with text, data, graphs, math, and models

• This “product” is currently under development. Give it a try at https://jupyterhub.gams.com/hub/login
Outlook: Solving Multiple Models Efficiently

- Solvelink & the GAMS Grid Facility
- GUSS – The Gather-Update-Solve-Scatter
Motivation

• Solving challenging real-world problems often involves the solution of many optimization problems
  • Decomposition Methods
  • Scenario Analysis
  • Heuristics
  • …

• Such approaches are often chosen, if solving the problem at hand does not work with a single monolithic model, e.g.
  • Due to it’s size and the required resources (e.g. memory)
  • Due to time restrictions (Problem should be solved in minutes but it takes days)
  • …

→ GAMS Grid Facility
→ Gather-Update-Solve-Scatter
Solvelink Option

Controls GAMS function when linking to solve.

```gams
Model transport /all/ ;
Option solvelink = {0 %Solvelink.ChainScript%,
  1 %Solvelink.CallScript%,
  2 %Solvelink.CallModule%,
  3 %Solvelink.AsyncGrid%,
  4 %Solvelink.AsyncSimulate%,
  5 %Solvelink.LoadLibrary%,
  6 %solveLink.aSyncThreads%,
  7 %solveLink.threadsSimulate%};

solve transport using lp minimizing z;
```
Solvelink Option – Sequential Solves

• ChainScript [0]: Solver process, GAMS vacates memory
  + Maximum memory available to solver
  + protection against solver failure (*hostile* link)
  - swap to disk

• Call{Script [1]/Module [2]}: Solver process, GAMS stays live
  + protection against solver failure (*hostile* link)
  + no swap of GAMS database
  - file based model communication

• LoadLibrary [5]: Solver DLL in GAMS process
  + fast memory based model communication
  + update of model object inside the solver (hot start)
  - not supported by all solvers
Solvelink Option – Sequential Solves

---

88 PARAMETER time  
  time for 100 scenarios

ChainScript 17.042,  
  CallModule 2.150,  
  LoadLibrary 0.631
Solvelink Option – Asynchronous Solves

• aSyncGrid [3]: GAMS starts the solution and continues in a Grid computing environment

• aSyncThreads [6]: The problem is passed to the solver in core without use of temporary files, GAMS does not wait for the solver to come back
Solvelink Option – Asynchronous Solves

```plaintext
62 Model transport /all/ ;
64 option lp=cpexd;
65 set s scenarios / s1*s100 /
66 sl solvelink / aSyncGrid, aSyncThreads /;
67 parameter dd(s,i,j) distance by scenario
68 ff(s) freight cost by scenario
69 time(*) time for 100 scenarios
70 sl_val(sl) solvelink value / aSyncGrid 3, aSyncThreads 6 /
71 h(s) scenario handle;
72 scalar tmp;
73 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
75 ff(s) = uniform(0.9,1.1)*f;
76 option limrow=0, limcol=0, solprint=silent;
77 * Async SOLVE
78 loop(sl,
79   tmp = jnow;
80   transport.solvelink=sl_val(sl);
81   loop(s,
82     d(i,j) = dd(s,i,j);
83     f = ff(s);
84     Solve transport using lp minimizing z ;
85     h(s) = transport.handle; // save instance handle
86   ));
87 repeat
88   display@readcollect(h) 'Waiting for next instance to complete';
89   loop(s@handlecollect(h(s)),
90     display@handledelt(h(s)) 'trouble deleting handles';
91     h(s) - 0; // indicate that we have loaded the solution
92   ));
94 until card(h) = 0 or timeelapsed > 180; // wait until all models are loaded
95 time(sl) = (jnow-tmp)*24*60*60;
96);
97 display time;
```

---
97 PARAMETER time time for 100 scenarios
aSyncGrid 2.991, aSyncThreads 0.144
Solvelink Option – Asynchronous Solves

• Helpful, if large ratio of solver time / GAMS time

---
124 PARAMETER time

time for 4 scenarios

ChainScript 21.523
CallModule 21.215
LoadLibrary 20.815
aSyncGrid 6.521
aSyncThreads 5.316
GUSS – Gather-Update-Solve-Scatter
aka Scenario Solver

Generates model once and updates the algebraic model keeping the model “hot” inside the solver.
GUSS – Gather-Update-Solve-Scatter
aka Scenario Solver

```plaintext
69 parameter dd(s,i,j) distance by scenario
70       ff(s)  freight cost by scenario
71 scalar time(*) time for 100 scenarios;
72 scalar tmp;
73
74 dd(s,i,j) = uniform(0.9,1.1)*d(i,j);
75 ff(s) = uniform(0.9,1.1)*f;
76 option limrow=0, limcol=0, solprint=off;
77
78 * GUSS
79 transport.solveLink = 0;
80 tmp = jnow;
81 Set mattrib / system.GUSSModelAttributes /
82 Parameter
83   xxGUSS(s,i,j) collector for level of x
84   srep(s, mattrib) model attibutes like modelstat etc
85 o(*) GUSS options / SkipBaseCase 1 /
86
87 Set dict / s . scenario."
88   o . opt . srep
89   d . param . dd
90   f . param . ff
91   x . level . xxGUSS /
92
93 Solve transport using lp minimizing z scenario dict;
94 time('GUSS') = (jnow-tmp)*24*60*60;
95
96 display time;
```

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.65</td>
<td>ChainScript</td>
</tr>
<tr>
<td>1.80</td>
<td>CallModule</td>
</tr>
<tr>
<td>0.44</td>
<td>LoadLibrary</td>
</tr>
<tr>
<td>3.22</td>
<td>aSyncGrid</td>
</tr>
<tr>
<td>3.18</td>
<td>aSyncThreads</td>
</tr>
<tr>
<td>0.36</td>
<td>GUSS</td>
</tr>
</tbody>
</table>

---- 137 PARAMETER time

time for 100 scenarios
Grid & GUSS – Examples from the model library

- **trnsgrid**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_trnsgrid.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_trnsgrid.html)
  - Simple asynchronous solves in a loop, separate collection loop

- **tgridmix**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_tgridmix.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_tgridmix.html)
  - Asynchronous solves in combined submission & collection loop. Keep number of submitted models <= #threads

- **guassgrid**: [https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_guassgrid.html](https://www.gams.com/latest/gamslib_ml/libhtml/gamslib_guassgrid.html)
  - Asynchronous GUSS-solves in combined submission & collection loop. Keep number of submitted models <= #threads

**GAMS and High Performance Computing**

*Thursday, September 13, 9:00 am - 10:15 am (Session TA-5)*
Thank You

Meet us at the GAMS booth!