Recent enhancements in GAMS

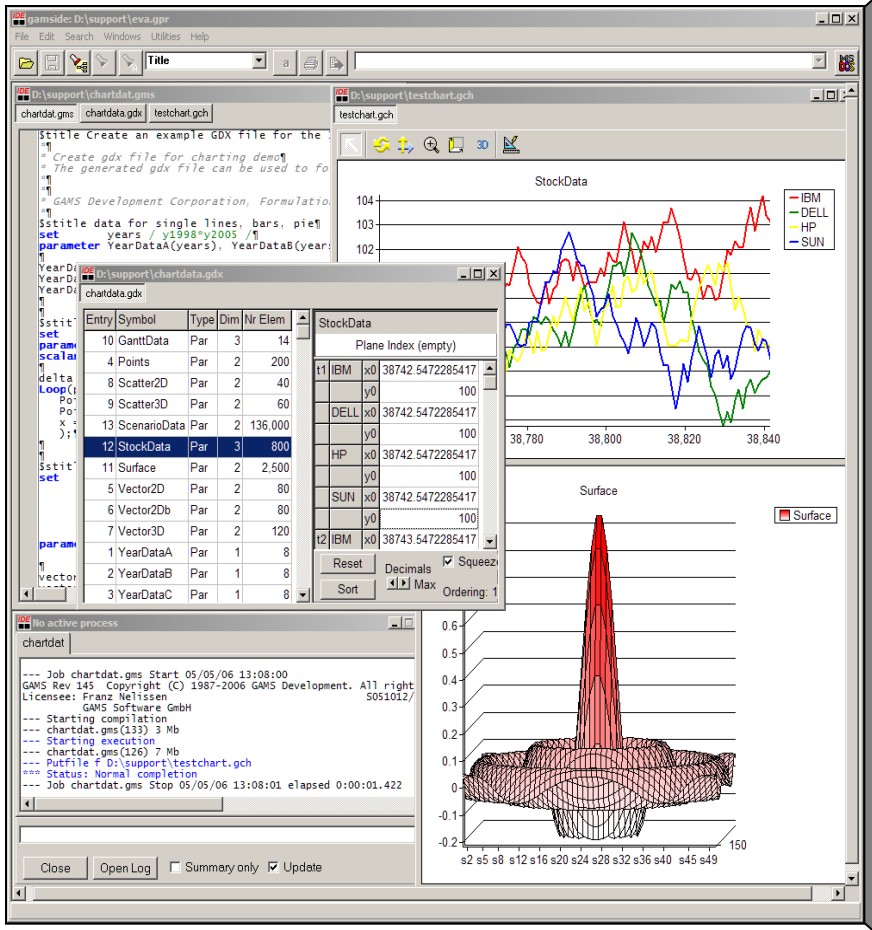
Lutz Westermann lwestermann@gams.com

GAMS Software GmbH

GAMS Development Corporation

www.gams.com

GAMS at a Glance



Algebraic Modeling System

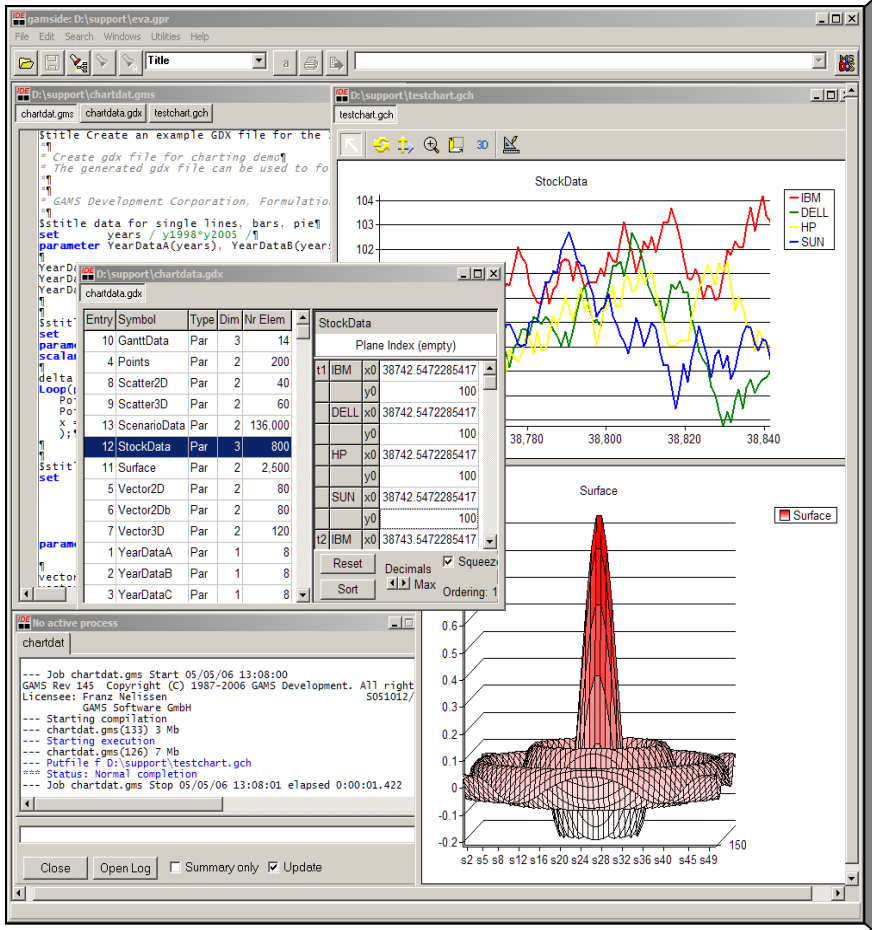
- Facilitates to formulate mathematical optimization problems similar to algebraic notation

  ➔ Simplified model building

- Provides links to appropriate state-of-the-art external algorithms

  ➔ Efficient solution process

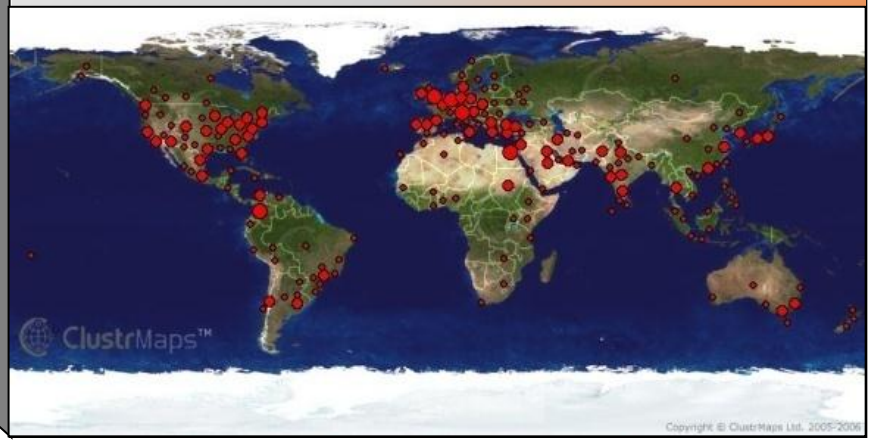
GAMS at a Glance



General **A**lgebraic **M**odeling **S**ystem

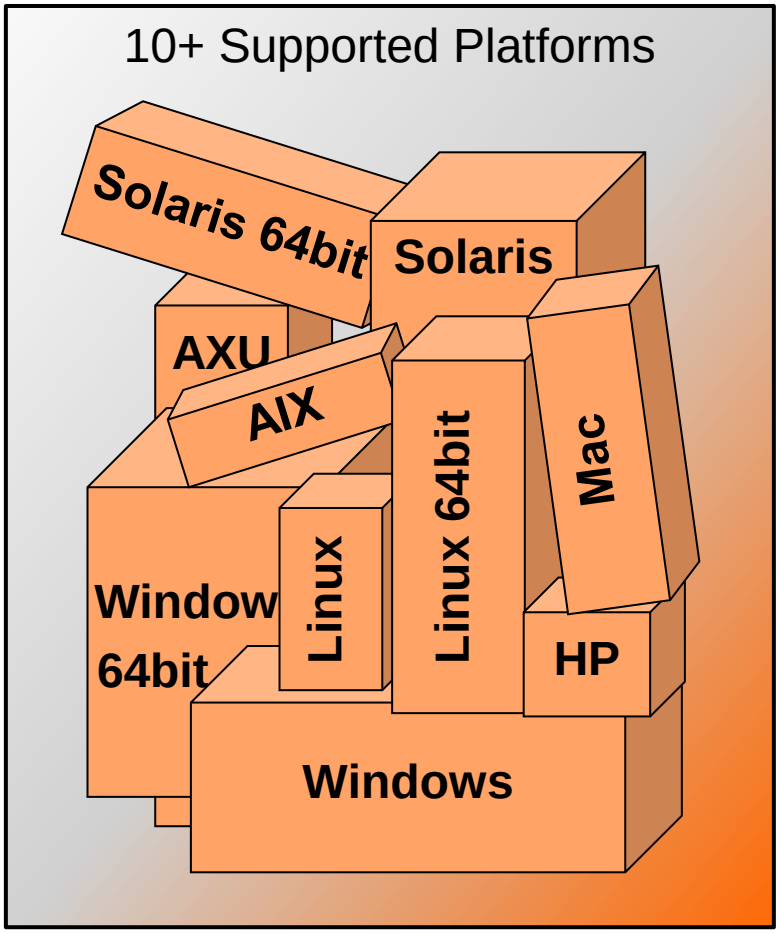
- Roots: World Bank, 1976
- Went commercial in 1987
- GAMS Development Corp.
- GAMS Software GmbH

- Broad academic & commercial user community and network



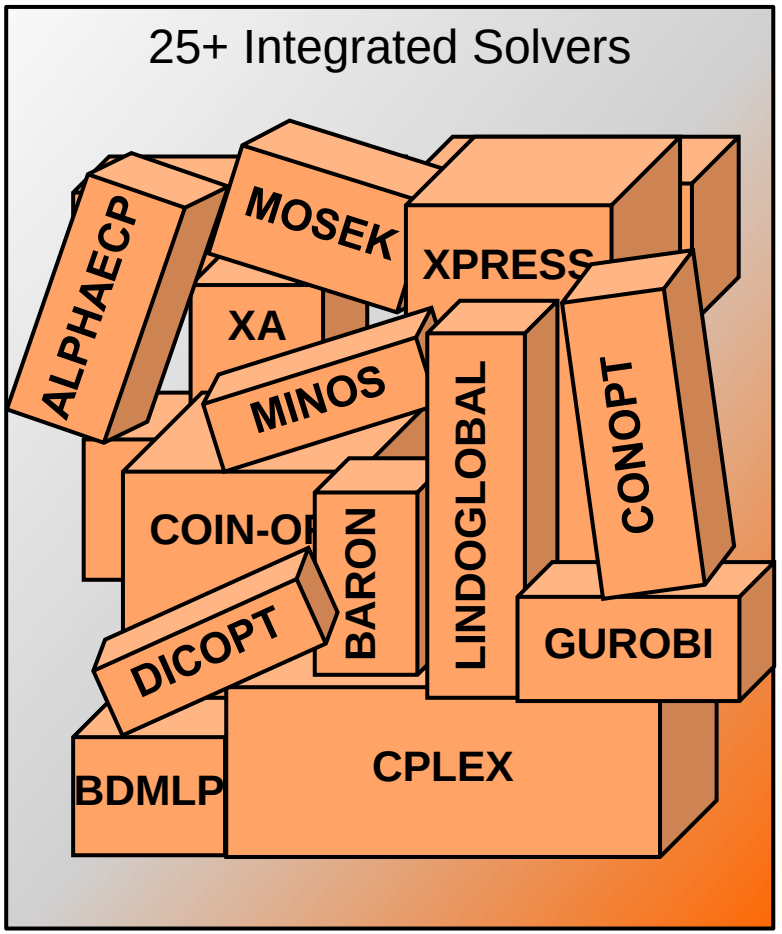
GAMS' Fundamental concepts

- **Platform independence**
- Hassle-free switch of solution methods
- Open architecture and interfaces to other systems
- Balanced mix of declarative and procedural elements



GAMS' Fundamental concepts

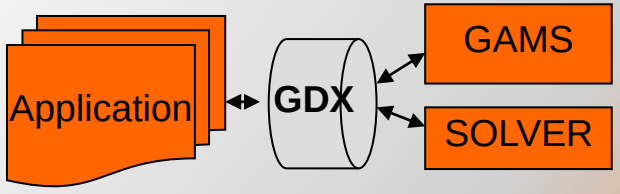
- Platform independence
- **Hassle-free switch of solution methods**
- Open architecture and interfaces to other systems
- Balanced mix of declarative and procedural elements



GAMS' Fundamental concepts

- Platform independence
- Hassle-free switch of solution methods
- **Open architecture and interfaces to other systems**
- Balanced mix of declarative and procedural elements

Binary Data Exchange



- Fast exchange of data
- Syntactical check on data before model starts
- Data Exchange at any stage (Compile and Run-time)
- Platform Independent
- Direct GDX interfaces and general API
- Scenario Management Support
- Full Support of Batch Runs

GAMS' Fundamental concepts

- Platform independence
- Hassle-free switch of solution methods
- Open architecture and interfaces to other systems
- **Balanced mix of declarative and procedural elements**

Declaration of..

- **Sets**
- **Parameters**
- **Variables**
- **Equations**
- **Models**
- **…**

Procedural Elements like…

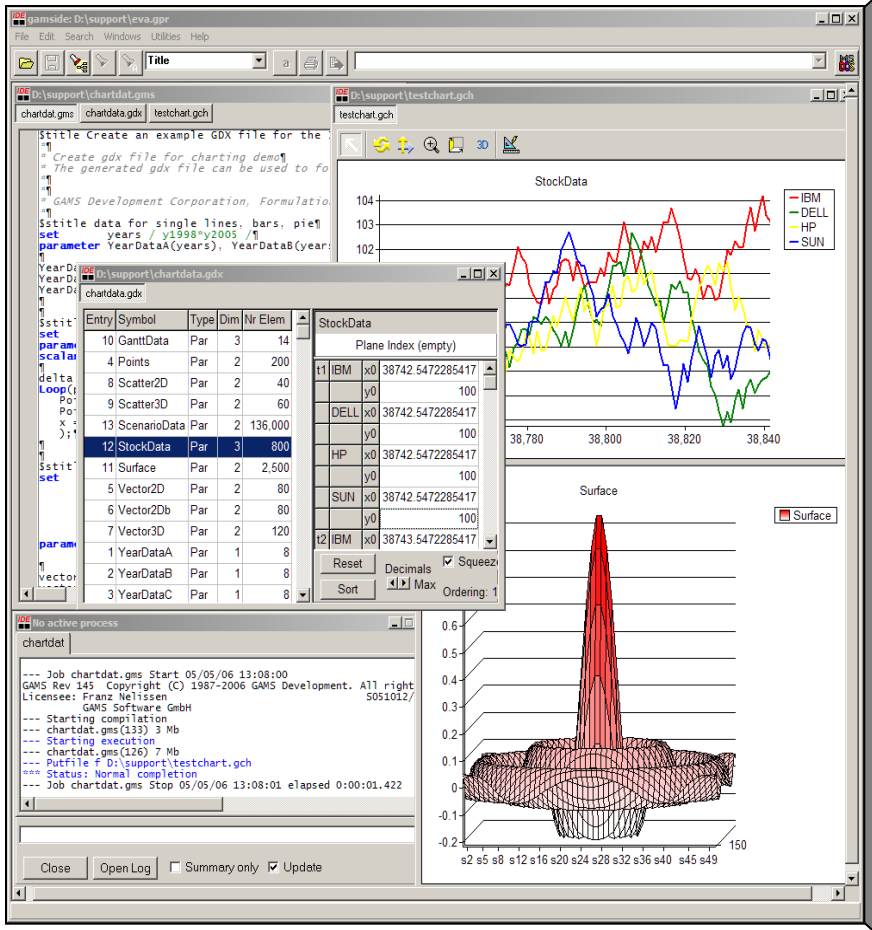
- **loops**
- **if-then-else**
- **…**

GAMS' Fundamental concepts

- **Different layers with separation of**
  - model and data
  - model and solution methods
  - model and operating system
  - model and interface

→ **Models benefit from**
  - advancing hardware
  - enhanced / new solver technology
  - improved / upcoming interfaces to other systems

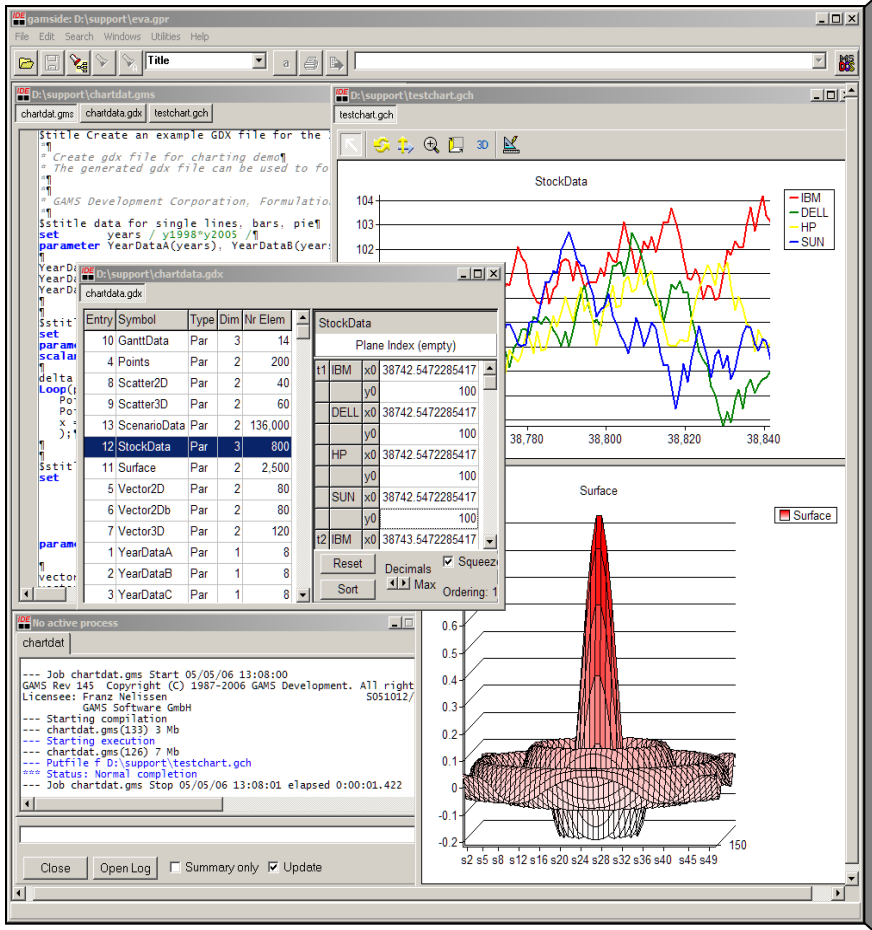
GAMS at a Glance



The GAMS/BASE Module

- Compiler and Execution System
- GAMS IDE (Windows)
- Documentation + Model libraries
- GDX Utilities
- Free Solvers/Solver Links

GAMS at a Glance

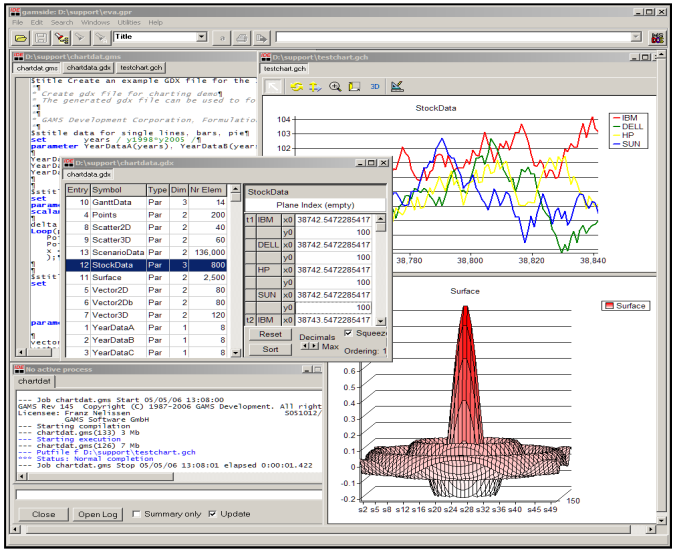


The GAMS/BASE Module

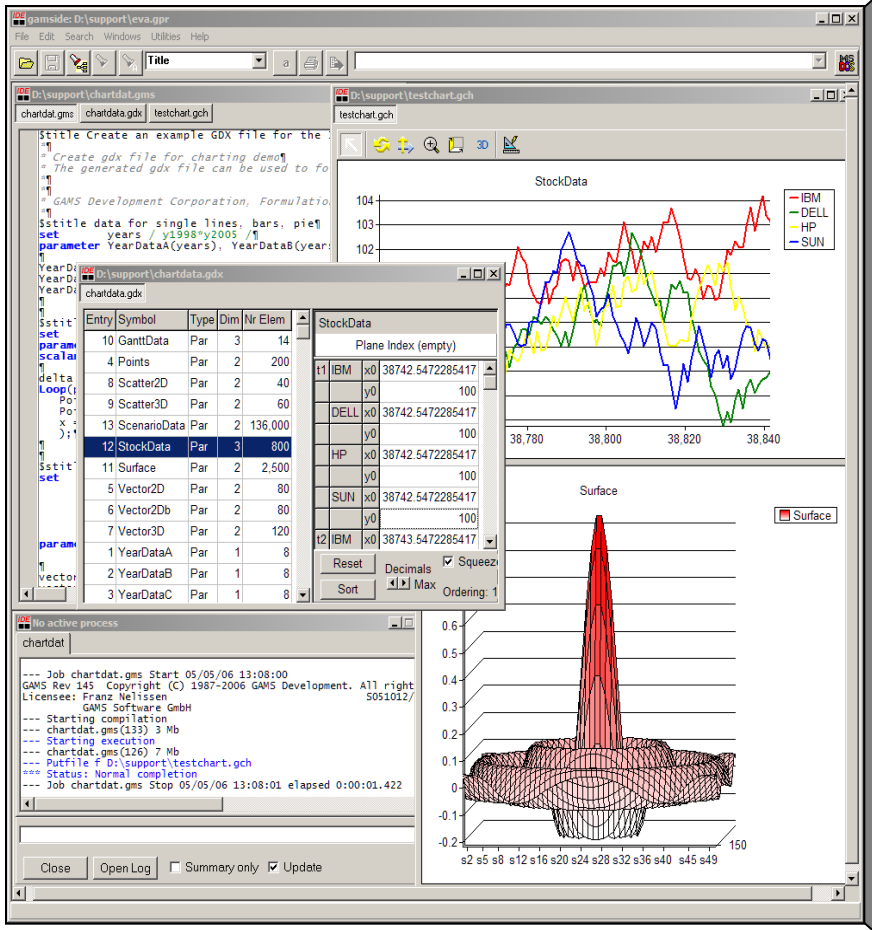
- Compiler and Execution System
- GAMS IDE (Windows)
- Documentation + Model libraries
- GDX Utilities
- Free Solvers/Solver Links

Integrated Development Environment

- Project management
- Editor / Syntax coloring / Spell checking
- Launching and monitoring of (multiple) GAMS processes
- Listing file / Tree view / Syntax-error navigation
- Solver selection / Option selection
- GDX viewer
  - Data cube
  - Data export (e.g. to MS Excel)
  - Charting facilities
- Model libraries
- Documentation



GAMS at a Glance

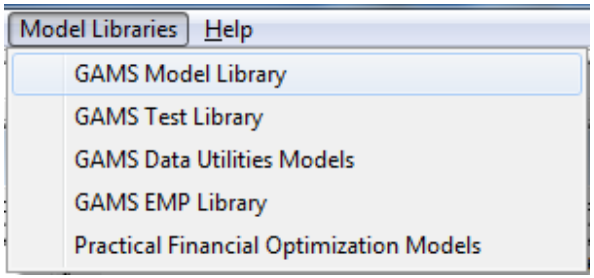


The GAMS/BASE Module

- Compiler and Execution System
- GAMS IDE (Windows)
- Documentation + Model libraries
- GDX Utilities
- Free Solvers/Solver Links

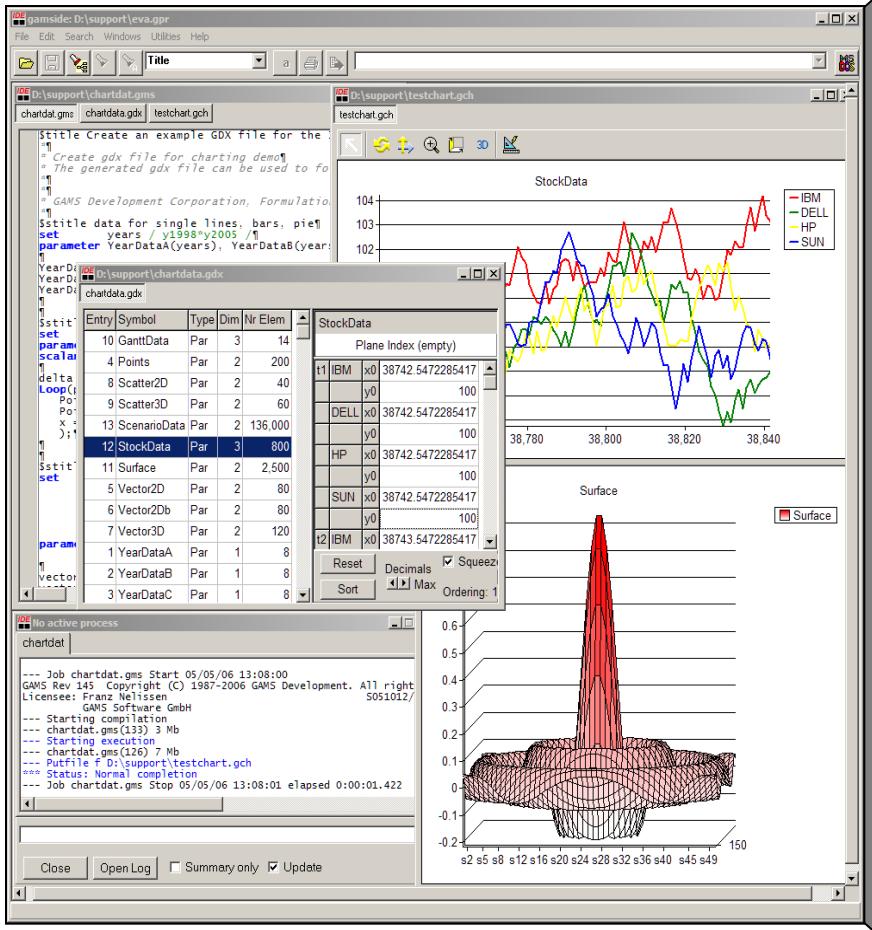
Documentation

- **Distributed Documentation**
  - GAMS Users Guide
  - Expanded GAMS Users Guide (McCarl)
  - Solver Manuals
  - GAMS Utility Manuals



- **Hundreds of models in different libraries**
- **Wikis**
  - Support Wiki http://support.gams-software.com
  - Interfaces Wiki http://interfaces.gams-software.com
- **User groups / Mailing lists**

GAMS at a Glance

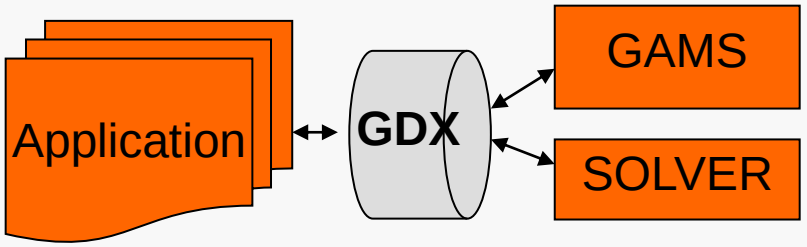


The GAMS/BASE Module

- Compiler and Execution System
- GAMS IDE (Windows)
- Documentation + Model libraries
- GDX Utilities
- Free Solvers/Solver Links

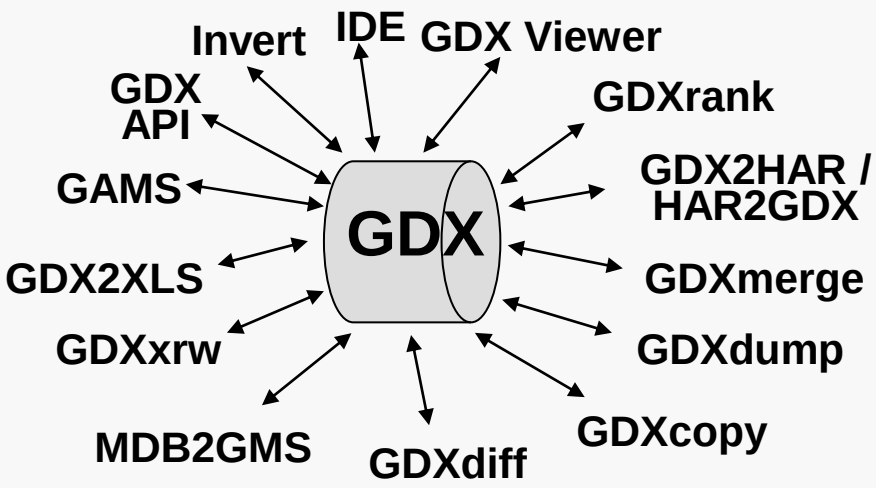
Gams Data eXchange

Binary Data Exchange

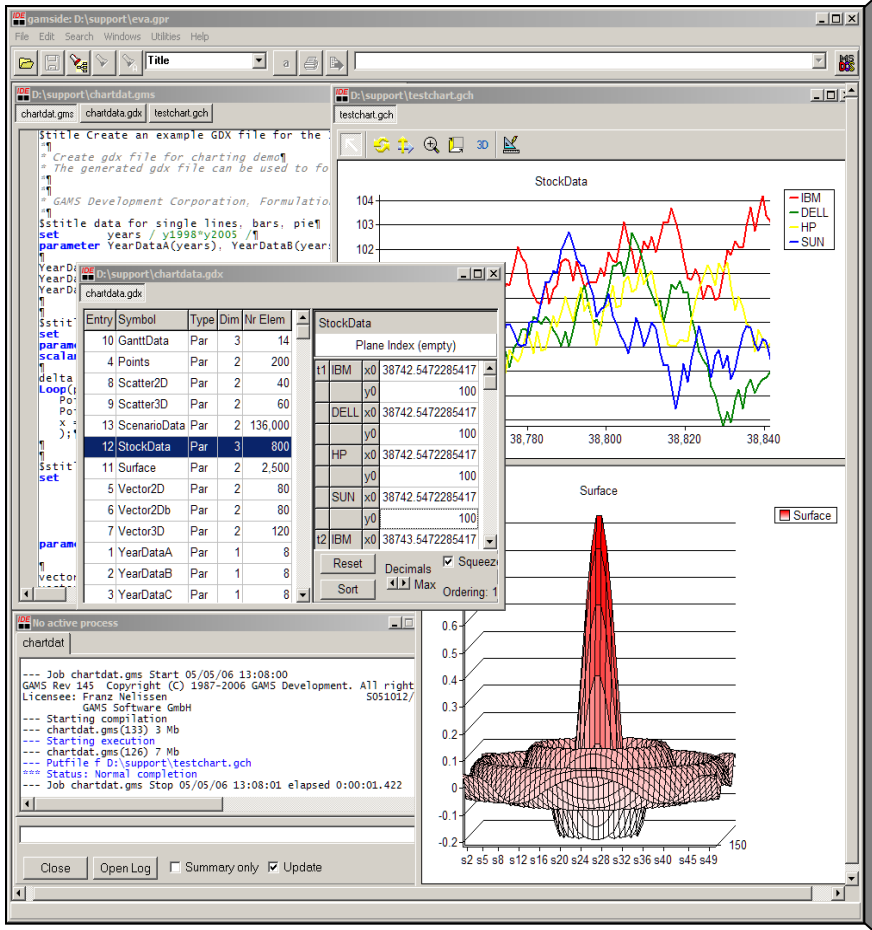


- Fast exchange of data
- Syntactical check on data before model starts
- Data Exchange at any stage (Compile and Run-time)
- Platform Independent
- Direct GDX interfaces and general API
- Scenario Management Support
- Full Support of Batch Runs

GDX Tools



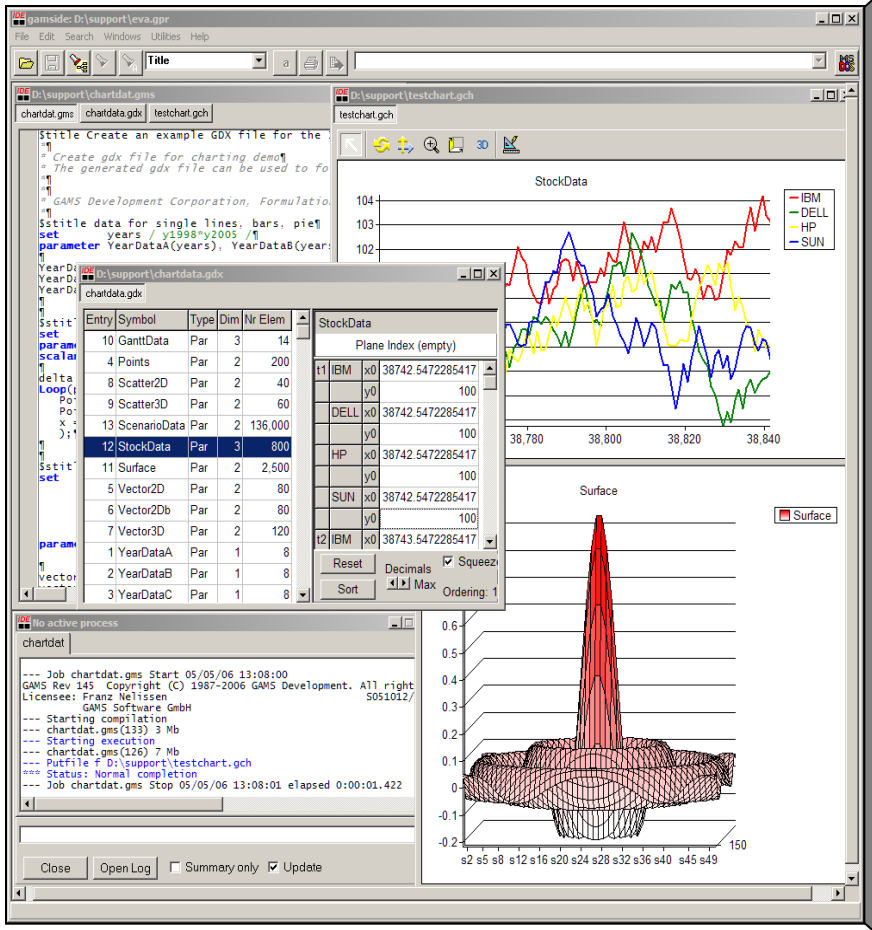
GAMS at a Glance



The GAMS/BASE Module

- Compiler and Execution System
- GAMS IDE (Windows)
- Documentation + Model libraries
- GDX Utilities
- Free Solvers/Solver Links

GAMS at a Glance



The GAMS/BASE Module

Free Solvers

- Convert
- EMP/JAMS, LOGMIP, NLPEC
- BENCH, EXAMINER, GAMSCHK
- BDMLP, LS, and MILES
- COIN-OR
  Cbc, IpOpt, BonMin, Couenne
- Scip (academic only)

New GAMS Distribution 23.7.3

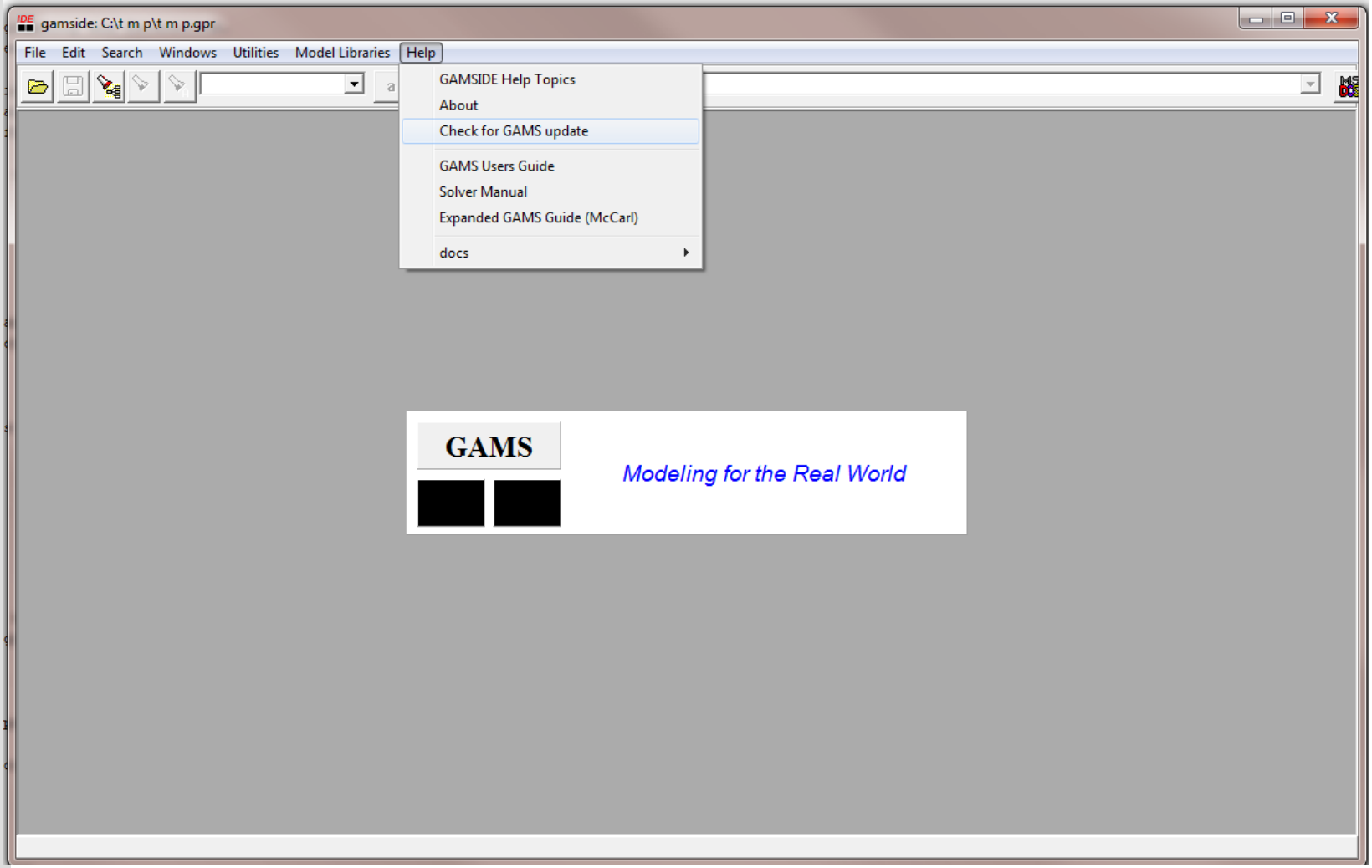
Released August, 24th www.gams.com/download

- **Solver updates**
  - AlphaECP 2.04.01
  - BARON 9.3
  - Conopt 3.14Y
  - CPLEX 12.3
  - GUROBI 4.5.1
  - Knitro 7
  - XPRESS 22.01
  - …

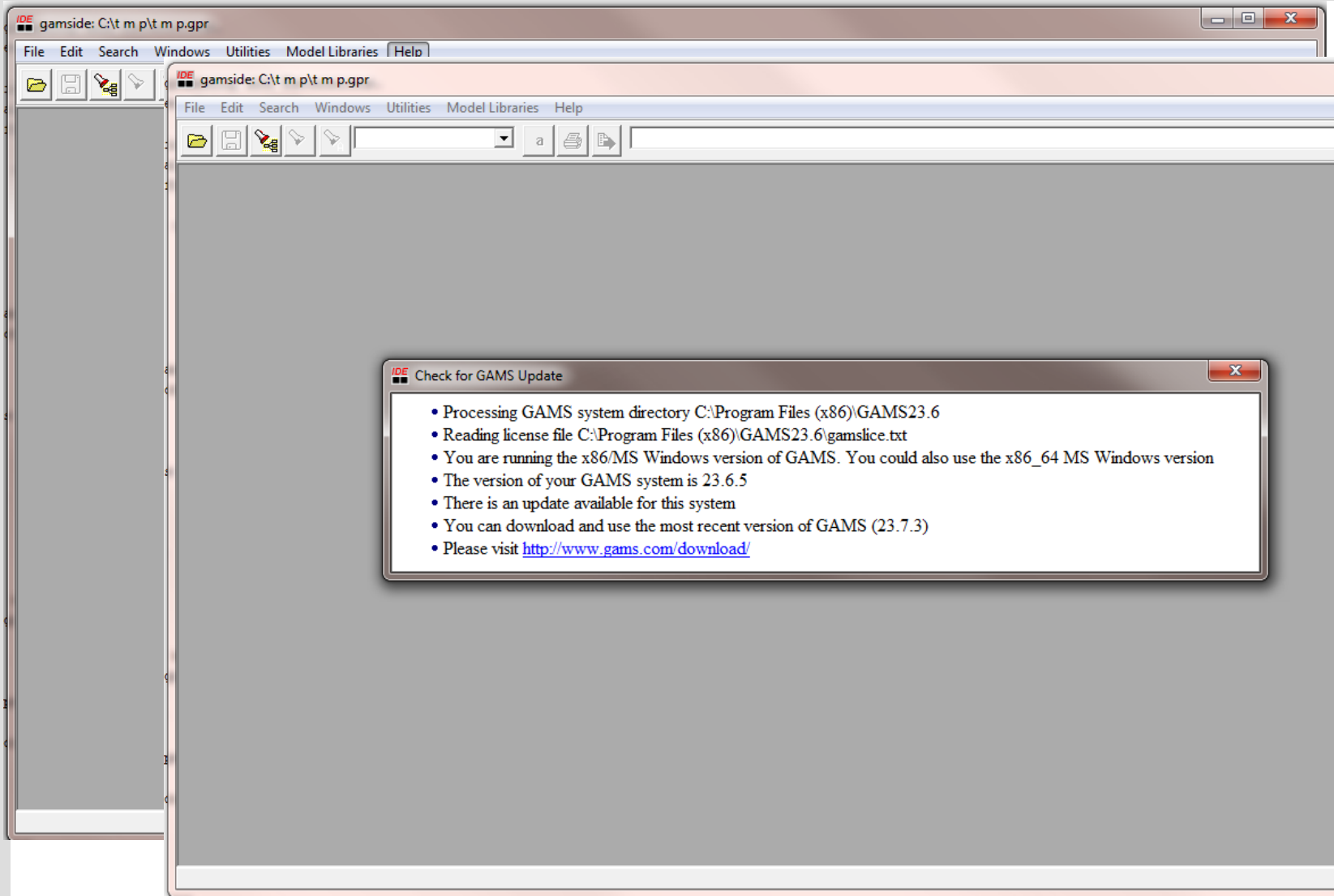
New GAMS Distribution 23.7.3

- Utilities
  - Check for GAMS Updates
- GAMS
  - Function Libraries
  - Asynchronous Execution
- Solvers
  - GUSS
- Application Programming Interface
  - GAMS Component Libraries

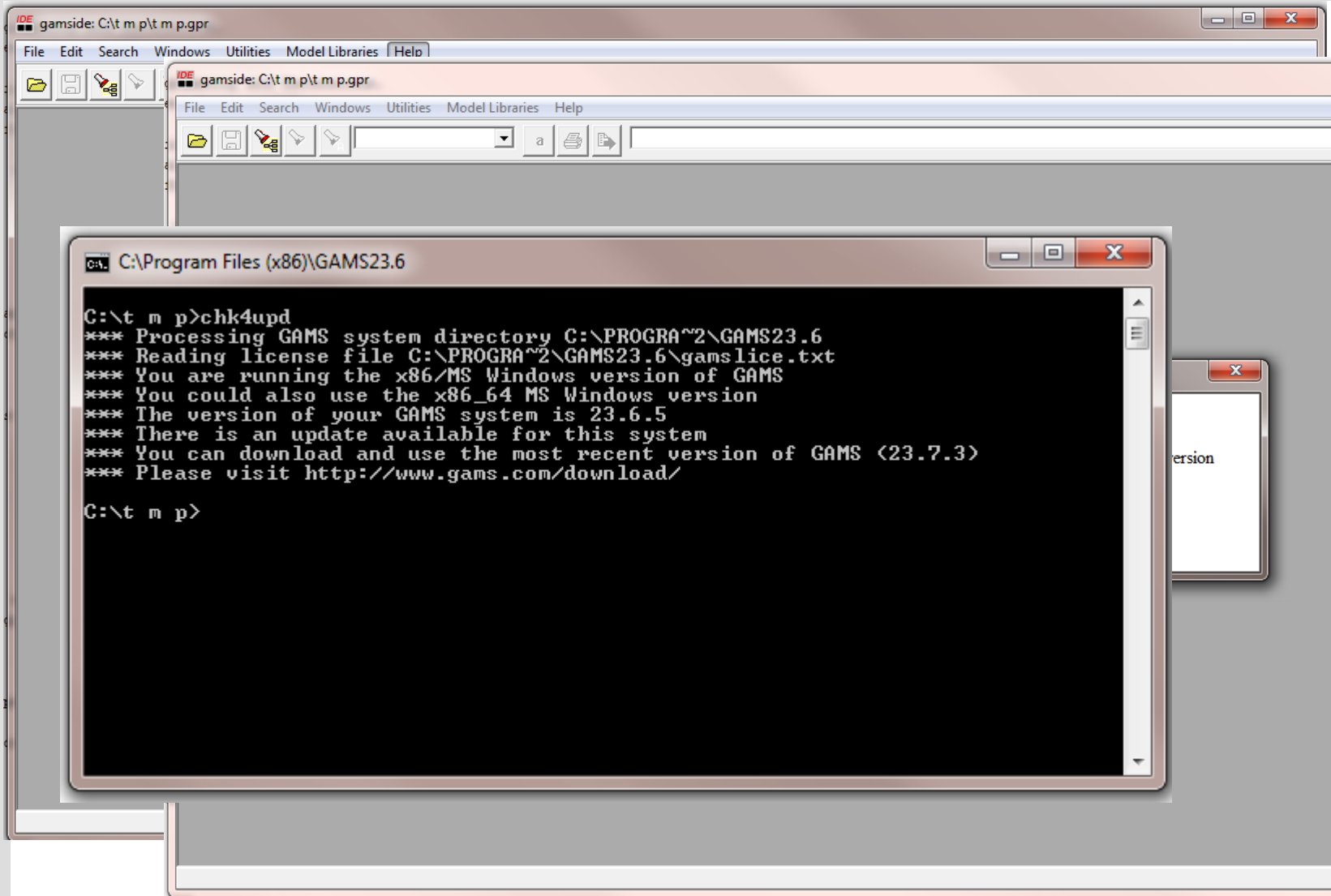
Check for GAMS Updates



Check for GAMS Updates



Check for GAMS Updates



Function Libraries

- Allows users to import functions from an external library into a GAMS model
- Imported functions can be used in the same way as intrinsic GAMS functions
- Some function libraries are included in the GAMS distribution
- Users can create their own libraries using an open programming interface (simple examples written in C, Delphi and Fortran come with every GAMS system)
- To make a library available call

```
$FuncLibIn <IntLibName> <ExtLibName>
```

- Declare functions similar to sets, parameters, ..., :

```
Function <IntFuncName> /<IntLibName>.<FuncName>/;
```

Function Libraries – Included Examples

- Fitfclib
  - FITPACK from P. Dierckx
  - One and two dimensional spline interpolation
- Pwpcclib
  - Piecewise polynomial function evaluation
- Stodclib
  - Random deviates, probability density functions, cumulative density functions and inverse cumulative density functions
  - E.g., ChiSquare, Gumbel, Logistic, Rayleigh, …
- Tricclib, Tridclib, Trifclib
  - Simple examples compiled and as source code written in C, Delphi and Fortran respectively

Function Libraries – Interface

- `int LibInit(`
  `abcRec_t *abc,` // in handle
  `const int version,` // in library version
  `char *msg)` // out message

- `int <FUNCTIONNAME>(`
  `abcRec_t *abc,` // in handle
  `const int DR,` // in derivative request
  `const int args,` // in number of arguments
  `const double x[],` // in arguments
  `double *f,` // out function value
  `double g[],` // out gradient
  `double h[],` // out hessian
  `void *cb,` // in error callback
  `void *usermem)` // in user memory for error callback

Asynchronous Execution

- `$Call /Execute /put_utility 'Exec':`
  Start jobs from GAMS

Asynchronous Execution

- `$Call / Execute / put_utility 'Exec':`
  Start jobs from GAMS

- Three ways to start a job asynchronously,
  - at compile time (CT):
    - `$Call.ASync …`
  - at execution time (ET):
    - `Execute.Async '…';`
    - `put_utility 'Exec.ASync' / '…';`

Asynchronous Execution

- Control job spawned:
  - Collect job handle:
    - CT: `$eval jh JobHandle`
    - ET: `jh = JobHandle;`
  - Check job status (error, still running, finished):
    - CT: `$eval status JobStatus(%jh%)`
    - ET: `status = JobStatus(jh);`
  - Interrupt/Kill a job:
    - CT: `$eval rc JobTerminate(%jh%)`
    - CT: `$eval rc JobKill(%jh%)`
    - ET: `rc = JobKill(jh);`
    - ET: `rc = JobTerminate(jh);`

Solving Scenarios

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

Setting	Solve time (secs)
Solvelink=0 (default)	40.297
Solvelink=%Solvelink.LoadLibrary%	03.625

Gather-Update-Solve-Scatter (GUSS)

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

- Update model data instead of matrix coefficients/rhs
- Hot start (keep the model hot inside the solver and use solver's best update mechanism)
- Save model generation and solver setup time

Gather-Update-Solve-Scatter (GUSS)

Setting	Solve time (secs)
Solvelink=0 (default)	40.297
Solvelink=%Solvelink.LoadLibrary%	03.625
GUSS	00.797

- Update model data instead of matrix coefficients/rhs
- Hot start (keep the model hot inside the solver and use solver's best update mechanism)
- Save model generation and solver setup time

Gather-Update-Solve-Scatter (GUSS)

Setting	Solve time (secs)
Solvelink=0 (default)	40.297
Solvelink=%Solvelink.LoadLibrary%	03.625
GUSS	00.797

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

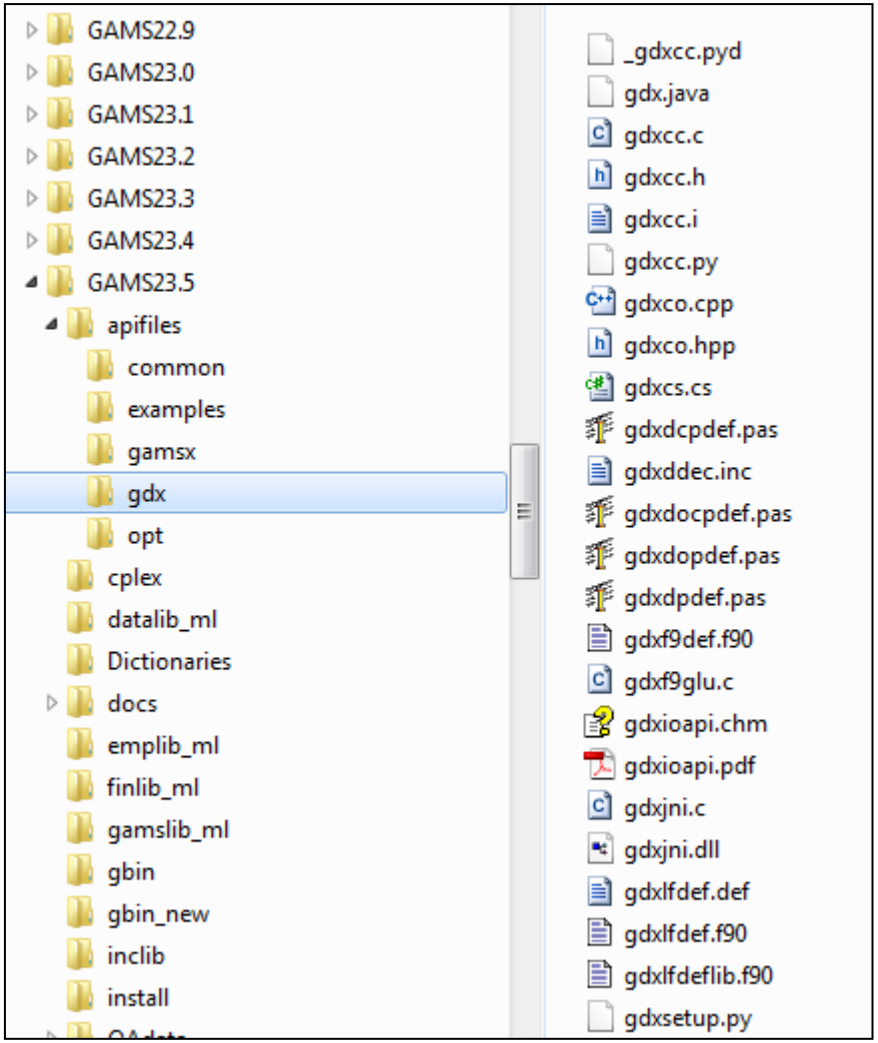
GUSS

- Dynamic model – rolling horizon

- Example:
  - Combined Heat and Power Planning with Heat Storage. All data known apriori but heat storage level
  - Can’t use GUSS
  - Implement GUSS in programming language
    - Identify some parameters as “modifiable” parameters
    - Implement rolling horizon in programming language

Distributed GAMS APIs

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



Calling GAMS from Python

```
if __name__ == "__main__":

    numberParams = len(sys.argv)
    if numberParams != 2 :
        print "Usage:", sys.argv[0], "sysDir"
        os._exit(1)

    gdxHandle   = new_gdxHandle_tp()
    optHandle   = new_optHandle_tp()
    gamsxHandle = new_gamsxHandle_tp()

    sysDir = sys.argv[1]
    print sys.argv[0], "using GAMS system directory:", sys.argv[1]

    assert gamsxCreateD(gamsxHandle, sysDir, GMS_SSSIZE)[0]
    assert gdxCreateD  (gdxHandle,   sysDir, GMS_SSSIZE)[0]
    assert optCreateD  (optHandle,   sysDir, GMS_SSSIZE)[0]
```

```
from gdxcc import *
from gamsxcc import *
from optcc import *
import sys
import os
```

```
    status = writeModelData(gdxHandle, "demanddata.gdx")
    if not status:
        print("Model data not written")
        terminate(gdxHandle, gamsxHandle, optHandle)
```

Creating Input for GAMS Model

```
    status = callGams(gamsxHandle, optHandle, sysDir)
    if not status:
        print("Call to GAMS failed")
        terminate(gdxHandle, gamsxHandle, optHandle)
```

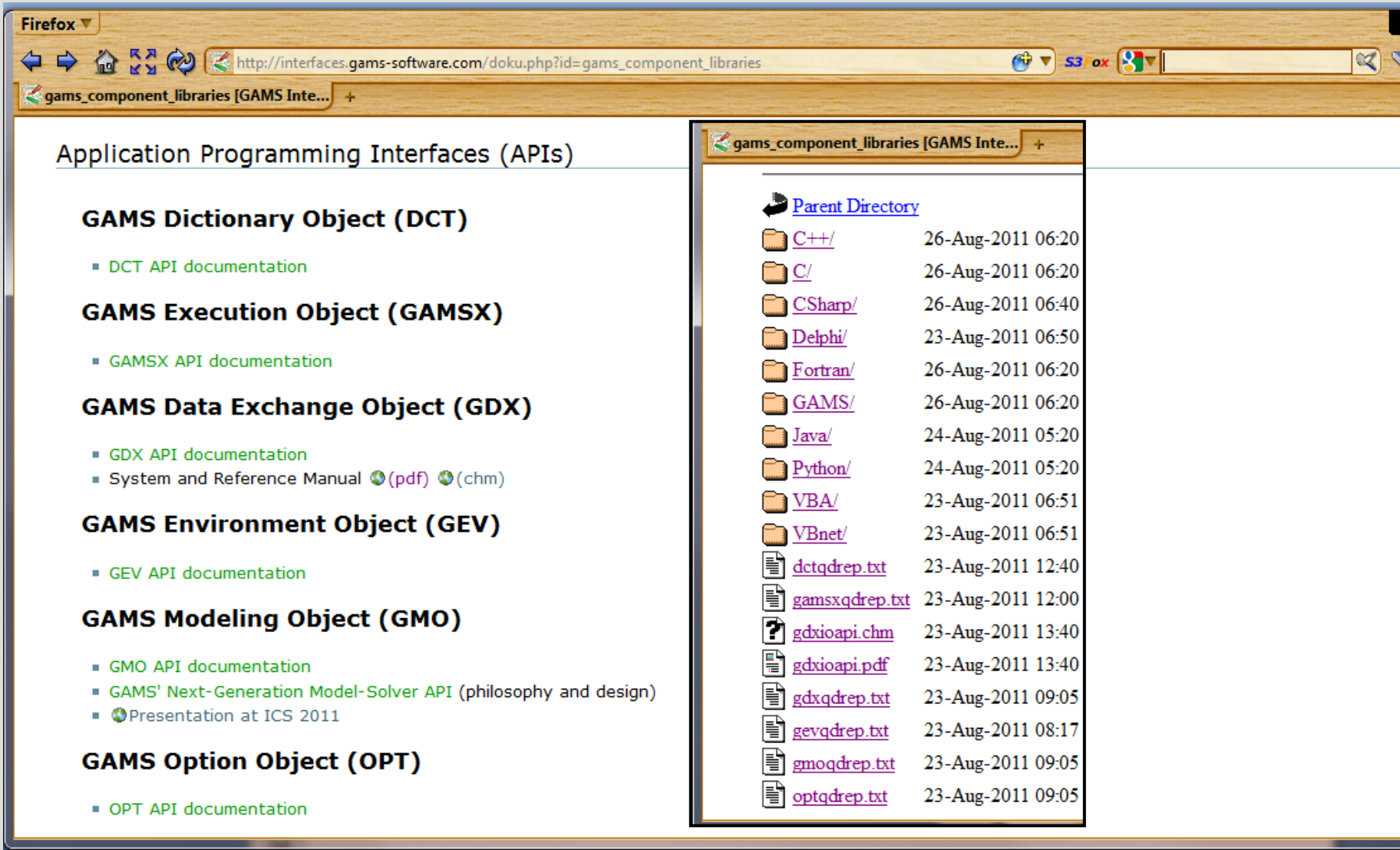
Callout to GAMS

```
    status = readSolutionData(gdxHandle, "results.gdx")
    if not status:
        print("Could not read solution back")
```

Reading Solution from GAMS Model

```
    terminate(gdxHandle, gamsxHandle, optHandle)
```

GAMS Component Libraries



GAMS Modeling Object (GMO)

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

Programming Language with GMO

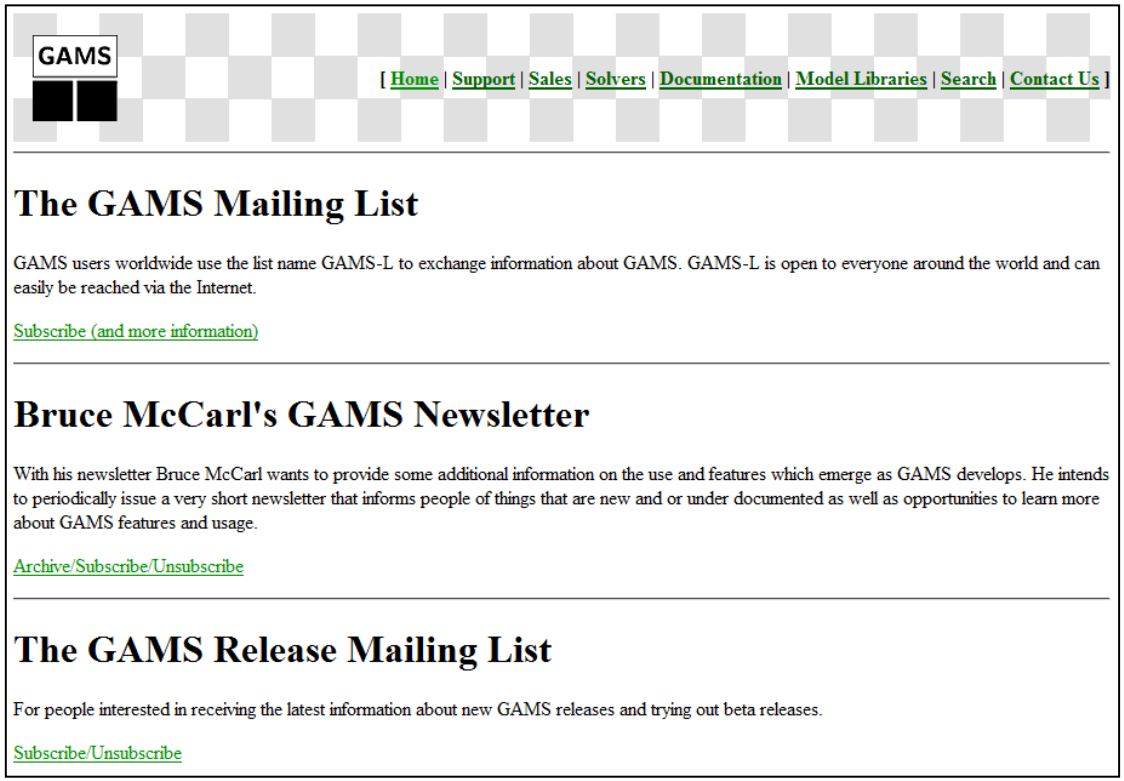
- Populated GMO object (e.g. by GAMS)
- GMO API to allow modification and alteration of bounds, rhs, “modifiable” parameters (NL expression evaluation)
- GMO/GEV (GAMS Environment Object) based solver links
- Various programming languages (Python, Java, …)

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

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

How to stay Up To Date

http://www.gams.com/maillist/



GAMS

[Home | Support | Sales | Solvers | Documentation | Model Libraries | Search | Contact Us]

The GAMS Mailing List

GAMS users worldwide use the list name GAMS-L to exchange information about GAMS. GAMS-L is open to everyone around the world and can easily be reached via the Internet.

Subscribe (and more information)

Bruce McCarl's GAMS Newsletter

With his newsletter Bruce McCarl wants to provide some additional information on the use and features which emerge as GAMS develops. He intends to periodically issue a very short newsletter that informs people of things that are new and or under documented as well as opportunities to learn more about GAMS features and usage.

Archive/Subscribe/Unsubscribe

The GAMS Release Mailing List

For people interested in receiving the latest information about new GAMS releases and trying out beta releases.

Subscribe/Unsubscribe

Please visit us at our booth in the exhibit area!

Contacting GAMS

Europe

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

Phone: +49 221 949 9170
Fax: +49 221 949 9171
http://www.gams.de

info@gams.de

USA

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

Phone: +1 202 342 0180
Fax: +1 202 342 0181
http://www.gams.com

sales@gams.com
support@gams.com