DATA AND SOFTWARE INTEROPERABILITY WITH GAMS: A USER PERSPECTIVE

Erwin Kalvelagen
erwin@amsterdamoptimization.com
Modeling Languages

- Specialized Modeling Languages (GAMS, AMPL,...) are very good in what they do
  - Efficient, compact representation of a model
  - In a way that allows thinking about the complete model
  - And that allows and encourages experiments and reformulations
  - Through maintainable models
  - Especially when they become really big
  - Handle irregular and messy data well
API attraction: Σειρήν

- Many new users are seduced to program against solver API’s
- Familiar environment, no new funny language to learn
- But for large, complex models this is really a step back
- Even when using higher level modeling-fortified API’s
## Modeling Environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Solver API</th>
<th>Modeling API</th>
<th>Modeling Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAMS, AMPL</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IBM ILOG</td>
<td>Cplex API</td>
<td>Concert</td>
<td>OPL</td>
</tr>
<tr>
<td>MS Solver Foundation</td>
<td>Solver Level API</td>
<td>SFS</td>
<td>OML</td>
</tr>
<tr>
<td>GLPK</td>
<td>API</td>
<td>Mathprog</td>
<td></td>
</tr>
</tbody>
</table>

Some programmers love:

- Malloc, Linker errors, Compiler versions, Library versions, Makefiles, Windows vs. Unix, Debuggers, ...

But for others this is time taken away from modeling...
Also increased use of optimization in...

- SAS (Improved OR module)
- Mathematica, Maple (Numerical Routines)
- Excel (MSF, Ilog, ...)
- MATLAB (Tomlab, Solver Interfaces)
- R (Solver Interfaces, Automatic Differentiation)
- GAMS, AMPL, ...

Excel (MSF, Ilog, ...)

SAS (Improved OR module)

MATLAB (Tomlab, Solver Interfaces)

GAMS, AMPL, ...
Answer: Interoperability

- Make GAMS more attractive for programmers by allowing to use external software
- Make the user decide what to do in GAMS or in other environment
- Make data exchange as easy as possible
  - Even for large data
  - Safety: this is a spot where lots of things can and will go wrong
Flexibility

- Do not decide for user
  - Data manipulation can be done in GAMS or Excel
  - Computation can be done in GAMS or external software
  - Allow these decisions to be made depending on the situation
    - Skills
    - Available software
    - Suitability
E.g. Data handling

Where to put functionality:
In Date Source Environment
or Modeling System

This has also to do with procedural vs declarative and with data manipulation capabilities.
GDX in practice

- It really works
  - Binary, fast
  - You can look at it
- Limitations:
  - Cannot add records or symbols
    - Eg: combine two gdx files
    - GDX is immutable
  - Not self contained wrt GAMS:
    - Needs declarations
  - Zero vs non-existent
  - GAMS interface
    - $load is dangerous
    - Compile time vs execution time
    - Execution time limits (each call separate gdx, cannot add set elements at execution time)
Example: USDA Farm database

- Imports 3 MDB + 1 XLS file → 1 GDX file
  - ± 5 million records (raw data)

### File Size (bytes)

<table>
<thead>
<tr>
<th>File</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FARMLandWaterResourcesDraft.mdb</td>
<td>80,650,240</td>
</tr>
<tr>
<td>FARMResourcesProductionDraft.mdb</td>
<td>429,346,816</td>
</tr>
<tr>
<td>GTAP54-NonLandIntensive.mdb</td>
<td>303,616,000</td>
</tr>
<tr>
<td>FIPS2&amp;FAOCountries.xls</td>
<td>29,184</td>
</tr>
<tr>
<td>farm.gdx</td>
<td>119,811,434</td>
</tr>
<tr>
<td>farm.gdx (compressed)</td>
<td>52,924,664</td>
</tr>
</tbody>
</table>

- Good, compact way to distribute and reuse large data sets (input or output)
- Aggregation easier in GAMS than in Access!
Example: USDA Reap Model

- Combine data from different sources
Conversion mdb -> gdx

```bash
$onecho > cmd.txt
I=FeedGrainsData.mdb
X=FeedGrainsData.gdx

q1=select commodity from tblCommodities
s1=commodity

q2=select attribute from tblAttributes
s2=attribute

q3=select period from tblTimePeriods
s3=period

q4=select unit from tblUnits
s4=unit

q5=select distinct(iif(isnull(isource),'blank',isource)) \ 
    from tblFG_update where \ 
    not isnull(year)
s5=isource

q6=select geo from tblgeography
s6=geocode

q7=select commodity,attribute,unit,iif(isnull(isource),'blank',isource),geocode,year,period,value \ 
   from tblFG_update where \ 
   not isnull(year)
p7=feedgrains
$offecho

$call mdb2gms @cmd.txt
```
Typical Problems

- NULL’s
- Duplicate records
- Multivalued tables
- More difficult processing:
  - Get latest available number
    - Difficult in SQL and in GAMS

Advantage of SQL: we can repair a number of problems on the fly.
Data manipulation

- Role of data manipulation in a modeling language
- OML
  - No data manipulation at all
  - Do it in your data source environment (e.g. Excel)
- AMPL
  - More extensive data manipulation facilities
  - Powerful if fits within declarative paradigm
- GAMS
  - Extensive use of data manipulation
  - Procedural
Policy Evaluation Models

- Often have serious data handling requirements
  - Aggregation/disaggregation
  - Estimation/Calibration
  - Simulation

- Examples:

<table>
<thead>
<tr>
<th>Model</th>
<th>LOC</th>
<th>LOC for eq’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polysys</td>
<td>22576</td>
<td>&lt; 500</td>
</tr>
<tr>
<td>Impact2000</td>
<td>17284</td>
<td>0</td>
</tr>
<tr>
<td>IntegratedIW5</td>
<td>20177</td>
<td>&lt; 500</td>
</tr>
</tbody>
</table>
Sparse Data: matrix multiplication

Ampl

param N := 250;
set I := {1..N};
param A{i in I, j in I} := if (i=j) then 1;
param B{i in I, j in I} := if (i=j) then 1;
param C{i in I, j in I} := sum{k in I} A[i,k]*B[k,j];
param s := sum{i in I, j in I} C[i,j];
display s;
end;

GAMS

set i /1*250/;
alias (i,j,k);
parameter A(i,j), B(i,j), C(i,j);
A(i,i) = 1;
B(i,i) = 1;
C(i,j) = sum(k, A(i,k)*B(k,j));
scalar s;
s = sum((i,j), C(i,j));
display s;
Timings

- gams-dense
- gams-sparse
- ampl-dense
- ampl-sparse
- glpk-dense
- glpk-sparse
Solving Linear Equations

- Solve $Ax=b$ for $x$
- Often not a good idea to calculate $A^{-1}$
- In GAMS we can solve by specifying $Ax=b$ as equations

```plaintext
linsys(i).. sum(j, a(i,j)*x(j)) =e= b(i);
```
If you really want the inverse of a matrix:

\[
A^{-1}A = I
\]
We can provide advanced basis so the calculation takes 0 Simplex iterations

- Inv.m(i,j) = 0; (var: basic)
- Inverse.m(i,j) = 1; (equ: non-basic)

<table>
<thead>
<tr>
<th>n</th>
<th>method=1</th>
<th>method=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.637</td>
<td>0.433</td>
</tr>
<tr>
<td>100</td>
<td>8.267</td>
<td>4.036</td>
</tr>
<tr>
<td>200</td>
<td>313.236</td>
<td>53.395</td>
</tr>
</tbody>
</table>
execute_unload 'a.gdx',i,a;
execute '=invert.exe a.gdx i a b.gdx pinva';
parameter pinva(i,j);
execute_load 'b.gdx',pinva;

GAMS
a.gdx
i,A(i,j)
inveA(i,j)
b.gdx
Invert
Test Matrix: Pei Matrix

\[
\begin{pmatrix}
1+\alpha & 1 & 1 & 1 & 1 \\
1 & 1+\alpha & 1 & 1 & 1 \\
1 & 1 & 1 & 1+\alpha & 1 \\
1 & 1 & 1 & 1 & 1+\alpha \\
\end{pmatrix}
\]

<table>
<thead>
<tr>
<th>n</th>
<th>method=1</th>
<th>method=2</th>
<th>method=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.637</td>
<td>0.433</td>
<td>0.027</td>
</tr>
<tr>
<td>100</td>
<td>8.267</td>
<td>4.036</td>
<td>0.055</td>
</tr>
<tr>
<td>200</td>
<td>313.236</td>
<td>53.395</td>
<td>0.118</td>
</tr>
</tbody>
</table>
Other tools

- Cholesky
- Eigenvalue
- Eigenvector
Max Likelihood Estimation

- NLP solver can find optimal values: estimates
- But to get covariances we need:
  - Hessian
  - Invert this Hessian
- We can do this now in GAMS
  - Klunky, but at least we can now do this
  - GDX used several times
MLE Estimation Example

* Data:
  * Number of days until the appearance of a carcinoma in 19 rats painted with carcinogen DMBA.

```
set i /i1*i19/;
table data(i,*)
   days  censored
     i1   143
     i2   164
     i3   188
     i4   188
     i5   190
     i6   192
     i7   206
     i8   209
     i9   213
     i10  216
     i11  220
     i12  227
     i13  230
     i14  234
     i15  246
     i16  265
     i17  304
     i18  216 1
     i19  244 1
;
set k(i) 'not censored';
k(i)$data(i,'censored')=0 = yes;
parameter x(i);
x(i) = data(i,'days');
scalars
   p 'number of observations'
   m 'number of uncensored observations'
;
p = card(i);
m = card(k);
display p,m;
```
MLE Estimation

*---------------------------------------------------------------*
* estimation                                                  *
*---------------------------------------------------------------*

scalar theta 'location parameter' /0/;

variables
    sigma 'scale parameter'
    c    'shape parameter'
    loglik 'log likelihood'

equation eloglike;

    c.lo = 0.001;
    sigma.lo = 0.001;

eloglike.. loglik =e= m*log(c) - m*c*log(sigma)
   + (c-1)*sum(k,log(x(k)-theta))
   - sum(i,((x(i)-theta)/sigma)**c);

model mle /eloglike/;
solve mle maximizing loglik using nlp;
Get Hessian

*--------------------------------------------------------------------------
* get hessian
*--------------------------------------------------------------------------

option nlp=convert;
$onecho > convert.opt
hessian
$offecho
mle.optfile=1;
solve mle minimizing loglik using nlp;

*
* gams cannot add elements at runtime so we declare the necessary elements here
*
set dummy /e1,x1,x2/;

parameter h(*,*,:) 'hessian';
execute_load "hessian.gdx",h;
display h;

set j /sigma,c/;
parameter h0(j,j);
h0('sigma','sigma') = h('e1','x1','x1');
h0('c','c') = h('e1','x2','x2');
h0('sigma','c') = h('e1','x1','x2');
h0('c','sigma') = h('e1','x1','x2');
display h0;
Invert Hessian

*-------------------------------------------------------------------------------
* invert hessian
*-------------------------------------------------------------------------------

execute_unload "h.gdx",j,h0;
execute "=invert.exe h.gdx j h0 invh.gdx invh";
parameter invh(j,j);
execute_load "invh.gdx",invh;
display invh;
Normal Quantiles

*-------------------------------------------------------------------------------
* quantile of normal distribution
*-------------------------------------------------------------------------------

* find
*   p = 0.05
*   q = probit(1-p/2)

scalar prob / .05 /

* we don't have the inverse error function so we calculate it
* using a small cns model
equations e;
variables probit;
e.. Errorf(probit) = e = 1-prob/2;
model inverterrorf / e /;
solve inverterrorf using cns;

display probit.l;

* verification:
*  > qnorm(0.975);
*  [1] 1.959964
* }
Finally: confidence intervals

*-------------------------------------------------------------------------------
* calculate standard errors and confidence intervals
*-------------------------------------------------------------------------------

```
parameter result(j,*)
result('c','estimate') = c.l;
result('sigma','estimate') = sigma.l;

result(j,'stderr') = sqrt(abs(invh(j,j)))
result(j,'conf lo') = result(j,'estimate') - probit.l*result(j,'stderr')
result(j,'conf up') = result(j,'estimate') + probit.l*result(j,'stderr')

display result;
```

<table>
<thead>
<tr>
<th></th>
<th>estimate</th>
<th>stderr</th>
<th>conf lo</th>
<th>conf up</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma</td>
<td>234.319</td>
<td>9.646</td>
<td>215.413</td>
<td>253.224</td>
</tr>
<tr>
<td>c</td>
<td>6.083</td>
<td>1.068</td>
<td>3.989</td>
<td>8.177</td>
</tr>
</tbody>
</table>
New developments

- Instead of calling external programs with a GDX interface
- Call a user provided DLL
- With simplified syntax:

```plaintext
Parameter A(i,j), B(j,i);
A(i,j) = ...
Scalar status;
BridgeCall('gamslapack', 'invert', A, B, Status);
```
Behind the scenes

- Map GAMS data to fortran, c, ...
- Deal with calling conventions (stdcall)
- Specified in a small spec file

```ini
file bridgelibrary.ini
[bridge]
id=GAMS bridge library
lib1=gamslapack
lib2=gamsgsl

beginning of file gamslapack.ini
[Library]
Version=1
Description=GAMS interface to LAPack
LibName=gamslapack
DLLName=gamslapack
Storage=F

[invert]
name=invert
i1=Q  // param1 = square matrix
i1d=i2
i2=D  // param 2 = n
o1=Q  // param 3 = square matrix
o1d1=i1,2
o1d2=i1,1
o2=N  // param 4 = info
status=o2
```

subroutine invert(a,n,b,info)
Gets more exciting when...

- We can parse subroutine headers
- In libraries
- And generate this automatically
- This will open up access to
  - Numerical, statistical libraries
  - Current tools as function (sql2gms, LS solver,...)
  - Etc.
- Longer term: also for equations
  - derivatives