GAMS Introduction

Erwin Kalvelagen
Amsterdam Optimization

GAMS: General Algebraic Modeling System

- GAMS: Modeling Language and its implementation
- Goal: concise specification of Math Programming models
  - Quick implementation of models
  - Maintainable models
  - Use of state-of-the-art solvers (Cplex, ….)
  - Support for large scale models
  - Support for linear and nonlinear models

History

- Developed at World Bank to achieve
  - Self documenting models
  - Quick turnaround when model changes
  - Maintainability
  - Solver independence
  - Support for nonlinear models
  - Automatic derivatives for NLP's
  - Initial versions developed in 1978-1979

GAMS: The Modelling Language

Sets are used for indexing

Decision variables

```
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
                  new-york       chicago      topeka
    seattle          2.5           1.7          1.8
    san-diego        2.5           1.8          1.4   ;

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 ;
```

```
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/ ;

Solve transport using lp minimizing z ;

Display x.l, x.m ;
```

Parameters don’t change inside a solve

Solve calls external optimizer

Equations are declared and then defined

Set Declarations

- ***<u>Set elements are strings</u>***
- Even if declared as
  - Set i /1*10/;
  - Set i /1,2,3,4,5,6,7,8,9,10/;
- Sets can have explanatory text:
  - Set y ‘years’ /year2000*year2010/;
- To get sequence number use ord()
    - P(i) = ord(i);
- Parameters, equations are expressed in terms of sets.

Set element names

- If contain blanks then need to be quoted

```
Set jx 'for use with X/XB variable'  /
      Imports
      "Food,Seed & Industial"
      Production
      'Paid Diversion'
/;
```

Explanatory text: these quotes are not needed if we had no / in the text

Double quotes

Single quotes. This can be important if the string already contains a single or double quote.

A valid set element can not contain both ' and "

Alias

- Often the same set is used in different index positions. E.g.
  - Parameter p(i,i);
  - p(i,i) = 1;   // assigns only diagonal
- Use Alias:
  - Alias(i,j);
  - Parameter p(i,j); // in declaration same as p(i,i)
  - p(i,j) = 1;   // assigns all i × j

Sub sets

- Subset:
  - Set j(i)
  - Hierarchy: start with supersets, then define subsets
  - You can have a subset of a subset
  - GAMS will check if elements are in superset (domain checking)

```

sets
  i0      /a,b,c,d/
  i1(i0) /a,b,c/
  i2(i1) /b,c,d/
****                $170
**** 170  Domain violation for element
;
```

Multi-dimensional Sets

- Specification of multi-dimensional sets

```
sets
 i    /a,b,c,d/
 j    /1,2,3/
 k(i,j) /
        a.1
        b.(2,3)
        (c,d).(1,3)
     /
;
display k;
```

```
----      12 SET k

             1           2           3

a          YES
b                      YES         YES
c          YES                     YES
d          YES                     YES
```

This is also domain checked

Multidimensional sets can not be used as domain.

Dynamic Sets

- Calculate sets dynamically.
- A.k.a. assigned sets
- Dynamic sets can not be used as domains.

```
set i /i1*i5/;
alias(i,j);

set offdiag(i,j) 'exclude diagonal';
offdiag(i,j) = yes;
offdiag(i,i) = no;

display offdiag;
```

```
----      8 SET offdiag  exclude diagonal

            i1          i2          i3          i4          i5

i1                     YES         YES         YES         YES
i2         YES                     YES         YES         YES
i3         YES         YES                     YES         YES
i4         YES         YES         YES                     YES
i5         YES         YES         YES         YES
```

Parameters

- Can be entered as
  - Scalar s 'scalar parameter' / 3.14/;
  - Parameter p(i) 'one dimensional parameter' /

```
  i1  2.5
  i2   4.8
/;
```

  - Table t(i,j) 'tabular specification of data'

```
        j1        j2        j3
i1      12                  14
i2                8.5
;
```

  - Assignment

```
p("i2") = 4.8;
t(i,j) = p(i) + 3;
```

The famous $ operator

- ‘Such that’ operator
- Used very often in GAMS models
  - Assignment of parameters
    - P(i,j)$(q(i,j)>0) = q(i,j);
    - P(i,j) = q(i,j)$(q(i,j)>0);
    - Note: these are different
  - Assignment of sets
  - Sum, prod, smax, smin, loop etc
    - S = Sum((i,j)$(q(i,j)>0),q(i,j));
  - In equation definitions (discussed later…)

Assignment: Lhs $ vs rhs $

```
set i /i1,i2/;
alias(i,j);

parameter p(i,j);

parameter q(i,j);
q(i,j) = -2;
q(i,i) = 2;

p(i,j) = 1;
P(i,j)$(q(i,j)>0) = q(i,j);
display p;

p(i,j) = 1;
P(i,j) = q(i,j)$(q(i,j)>0);
display p;
```

```
----      12 PARAMETER p

             i1          i2

i1        2.000       1.000
i2        1.000       2.000


----      15 PARAMETER p

             i1          i2

i1        2.000
i2                    2.000
```

Parallel Assignment

- Parallel assignment:
  - P(i,j) = xxx;
  - No loop needed
- With loop

```
Loop((i,j),
   p(i,j)=xxx;
);
```

- Sometimes beginners use loops too much

Sparse storage

- Only nonzero elements are stored
  - Zero and 'do not exist' is identical in GAMS

```
set i/ i1,i2/;
alias (i,j);

table t(i,j)
      i1  i2
  i1   1
  i2       3
;

scalar n1,n2;
n1 = card(t);
n2 = sum((i,j)$t(i,j),1);
display n1,n2;
```

Domain Checking

- Makes models more reliable
- Like strict type checking in a programming language

```
      set
        i /a,b,c/
        j /d,e,f/
      ;

      parameter p(i);
      p(i) = 1;
      p(j) = 2;
****     $171
**** 171  Domain violation for set
      p('g') = 3;
****      $170
**** 170  Domain violation for element
```

Bypassing domain checking

- Use * as set to prevent domain checking
  - Parameter p(*);
- This is not often needed, sometimes useful to save a few key-strokes.

```
table unitdata(i,*)
          capacity minoutput mindown minup inistate coefa coefb   coefc     chot ccold tcool
*               MW        MW       H     H        H   $/h $/MWh $/MW^2h      $/h   $/h     h
  unit1        455       150       8     8        8  1000 16.19 0.00048     4500  9000     5
  unit2        455       150       8     8        8   970 17.26 0.00031     5000 10000     5
  unit3        130        20       5     5       -5   700 16.60 0.00200      550  1100     4
  unit4        130        20       5     5       -5   680 16.50 0.00211      560  1120     4
  unit5        162        25       6     6       -6   450 19.70 0.00398      900  1800     4
  unit6         80        20       3     3       -3   370 22.26 0.00712      170   340     2
  unit7         85        25       3     3       -3   480 27.74 0.00079      260   520     2
  unit8         55        10       1     1       -1   660 25.92 0.00413       30    60     0
  unit9         55        10       1     1       -1   665 27.27 0.00222       30    60     0
  unit10        55        10       1     1       -1   670 27.79 0.00173       30    60     0
;
```

Data Manipulation

- Operate on parameters
- Often large part of the complete model
- Operations:
  - Sum,prod,smax,smin,
  - Functions: sin,cos,max,min,sqr,sqrt etc
  - $ conditions
  - If, loop
  - For, while (not used much)

Checks

- Abort allows to add checks:

```
scalars total_demand, total_capacity;
total_demand = sum(j, b(j));
total_capacity = sum(i, a(i));
display total_demand, total_capacity;

abort$(total_demand > total_capacity + 0.001) "Capacity too small to meet demand";
```

```
*
* check for balanced demand and supply
*
scalar totalsupply, totaldemand;
totalsupply = sum(i, s(i));
totaldemand = sum(j, d(j));
abort$(abs(totalsupply-totaldemand) > 0.01) "Unbalanced supply and demand";
```

Variables

- Declaration:
  - Free variable x(i); // default!
  - Positive variable y(i,j); // this means non-negative
  - Binary variable z;
  - Integer variable d;
  - Can be declared in steps, as long as no contradiction:
    - Variable x,y,z; Positive Variable x(i);
- For MIP/MINLP models extra variable types:
  - Sos1, sos2, semicont, semiint
- Free variable is the default. Most other systems have positive variables as the default.

Variables (2)

- x.lo=1; sets lower bound
- Y.up(i)=100; sets upper bound
- Z.L is level
- X.M is marginal (reduced cost, dual)
- Z.Scale sets scale for NLP
- Z.prior sets priorities for MIP
- X.fx=1 is shorthand for x.lo=1;x.up=1;x.L=1; (cannot by used in rhs)

Equations

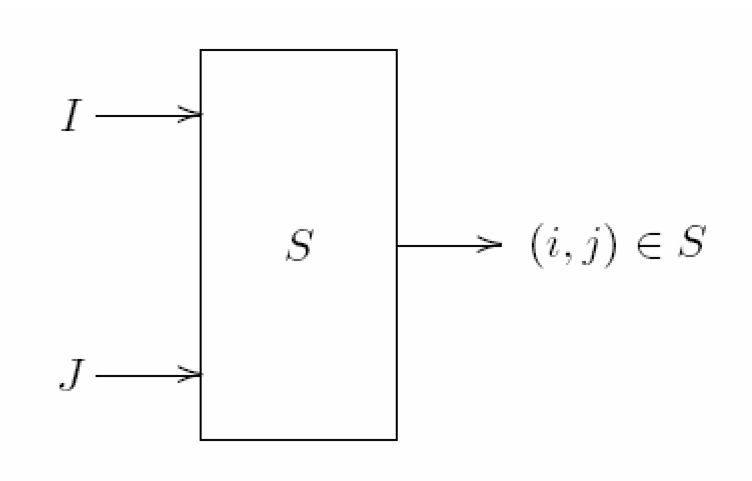
- Declaration:
  - Equation e(i) ‘some equation’;
- Definition:
  - e(i).. sum(j, x(i,j)) =e= 1;
- This generates card(i) equations
- $ conditions:
  - e(i)$subset(i).. sum(j, x(i,j)) =e= 1;
- Equation types
  - =E=, =L=, =G=
  - =X= (external functions)
  - =N= (nonbinding, not used much)
  - =C= (conic equation, not used much)

Maps

```
distance(i,j)$(lt(i,j)).. d =l= sqrt( sqr(x(i)-x(j)) + sqr(y(i)-y(j)) );
```

identical to

```
distance(lt(i,j)).. d =l= sqrt( sqr(x(i)-x(j)) + sqr(y(i)-y(j)) );
```



A map is a filter

In the rhs both i,j and lt can be used:

```
distance(lt(i,j))..
  d(lt) =e= sqrt(sqr[x(i)-x(j)]+sqr[y(i)-y(j)]);
```

Parameter vs variable

- Nonlinear

```
Variable y;
e.. x =e= sqr(y);
```

- Linear

```
Parameter p;
e.. x =e= sqr(p);
```

```
Variable y;
e.. x =e= sqr(y.L);
```

Special Values

- INF
  - Infinity: often used for bounds
- -INF
  - Minus infinity: mostly for bounds
- NA
  - Not available: not much used
- EPS
  - Numerically zero
  - Marginal is zero but nonbasic → EPS
- UNDF
  - Eg result if division by zero

```
   1  parameter x,y;
   2  x=0;
   3  y=1/x;
   4  display y;

**** Exec Error at line 3: division by zero (0)

----      4 PARAMETER y                    =          UNDF
```

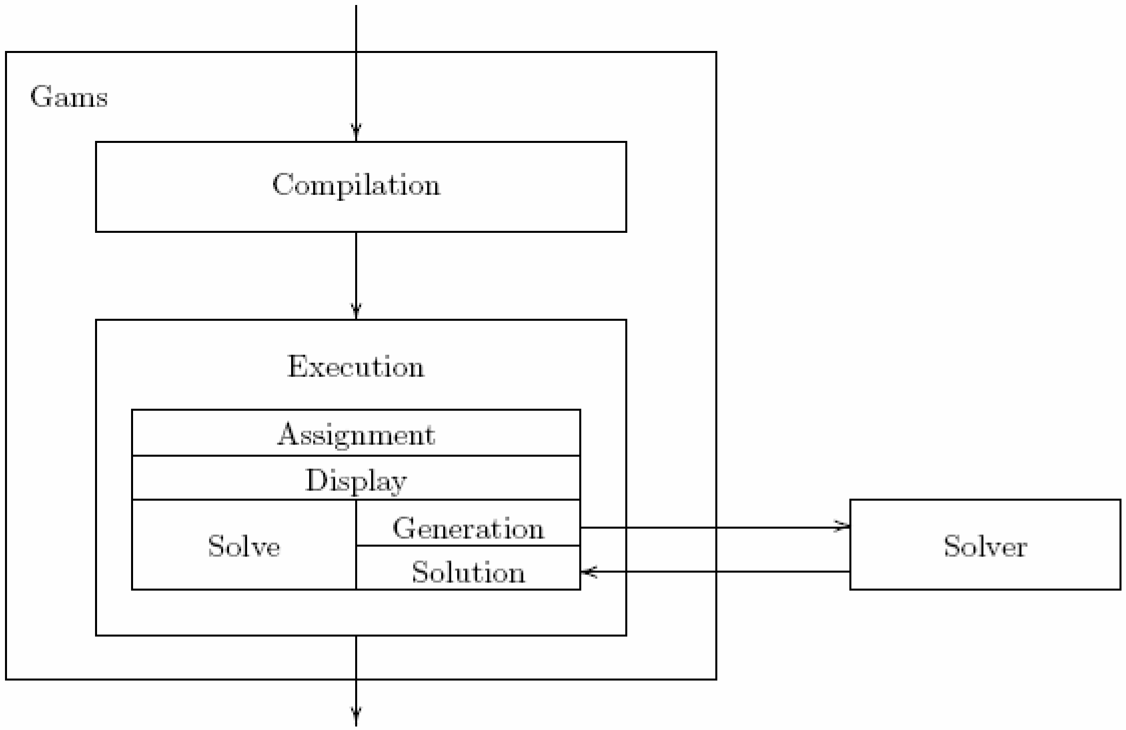
Model statement

- Model m /all/;
- Model m /cost,supply,demand/;
- Special syntax for MCP models to indicate complementarity pairs:
  - Model m /demand.Qd, Psupply.Qs, Equilibrium.P/

Solve Statement

- Solve m minimizing z using lp;
- GAMS uses objective variable instead of objective function
- Model types
  - LP: linear programming
  - NLP: nonlinear programming
  - DNLP: NLP with discontinuities (max,min,abs)
  - MIP: linear mixed integer, RMIP: relaxed MIP
  - MINLP: nlp with integer vars, RMINP: relaxed minlp
  - QCP,MIQCP: quadratically constrained
  - CNS: constrained non-linear system (square)
  - MCP: mixed complementarity
  - MPEQ: NLP with complementarity conditions

GAMS Flow of Control



Solvers

- To select solver
  - Option lp=cplex;
  - Command line parameter: lp=cplex
  - Change defaults (IDE or GAMSINST)
- Switching solvers is easy and cheap

Solver/Model type availability - 22.7 May 1, 2008												
	LP	**MIP**	**NLP**	**MCP**	**MPEC**	**CNS**	**DNLP**	**MINLP**	**QCP**	**MIQCP**	***Stoch.***	***Global***
ALPHAECP								✔		✔		
BARON 8.1	✔	✔	✔				✔	✔	✔	✔		✔
BDMLP	✔	✔										
COIN	✔	✔										
CONOPT 3	✔		✔			✔	✔		✔			
CPLEX 11.0	✔	✔							✔	✔		
DECIS	✔										✔	
DICOPT								✔		✔		
KNITRO 5.1	✔		✔				✔		✔			
LINDOGLOBAL 5.0	✔	✔	✔				✔	✔	✔	✔		✔
LGO	✔		✔				✔		✔			✔
MILES				✔								
MINOS	✔		✔				✔		✔			
MOSEK 5	✔	✔	✔				✔		✔	✔		
MPSGE												
MSNLP			✔				✔		✔			✔
NLPEC				✔	✔							
OQNLP			✔				✔	✔	✔	✔		✔
OSL V3	✔	✔										
OSLSE	✔										✔	
PATH				✔		✔						
SBB								✔		✔		
SNOPT	✔		✔				✔		✔			
XA	✔	✔										
XPRESS 18.00	✔	✔							✔			
Contributed Plug&Play solvers												
AMPLwrap	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
DEA	✔	✔										
Kestrel	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Linear Programming

- Very large models can be solved reliably
- Primal and Dual Simplex and interior point (barrier) methods.
  - Free solvers:
    - BDMLP
    - COINGLPK
    - COINCBC
  - CPLEX (Ilog)
    - commercial, parallel, state-of-the-art, simplex+barrier
  - XPRESS (Fair Isaac)
    - commercial, parallel, state-of-the-art, simplex+barrier
  - MOSEK
    - Very good parallel interior point
  - XA
    - cheaper alternative

Linear Programming (2)

- Many additional algorithms determine success
  - Scaling
  - Presolver (reduce size of model)
  - Crash (find good initial basis)
  - Crossover (interior point solution → basic solution)
- Very large models (> 10 million nonzero elements) require much memory
- 64 bit architecture can help then (available on pc's, so no need for super computers like this Cray C90)

Performance improvement

- Indus89 model ran for 6-7 hours on a DEC MicroVax in 1990 using MINOS as LP solver
- This model runs now with Cplex on a laptop well within 1 second

LP Modeling

- Almost anything you throw at a good LP solver will solve without a problem
- If presolver reduces the model a lot or if you have many x.fx(i)=0 then revisit equations and exclude unwanted variables using $ conditions.

LP Modeling (2)

- Don’t reduce #vars,#equs if this increases the number of nonzero elements significantly

```
e(k).. x(k)  =L= sum(j, y(j))
```

K equations
K+J variables
K×(J+1) nonzeroes

e.g.
100 equations
200 variables
10100 nonzeroes

```
e(k).. x(k) =L= ysum;
Ydef.. ysum =e= sum(j,y(j));
```

K+1 equations
K+J+1 variables
2K+J+1 nonzeroes

e.g.
101 equations
201 variables
301 nonzeroes

LP Listing File

- Part 1: echo listing of the model. Occasionally useful to look at syntax errors or run time errors.
- The compilation time is usually small

```
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   / ;
```

```
COMPILATION TIME      =          0.016 SECONDS
```

LP Listing File (2)

- Part 2: equation listing
  - Shows first 3 equations for each block
  - INFES is for initial point, so don't worry
  - Note how explanatory text is carried along
  - Especially useful for difficult equations with leads and lags
  - More or less can be shown with OPTION LIMROW=nnn;

```
---- demand  =G=  satisfy demand at market j

demand(new-york)..  x(seattle,new-york) + x(san-diego,new-york) =G= 325 ; (LHS = 0, INFES = 325 ****)

demand(chicago)..  x(seattle,chicago) + x(san-diego,chicago) =G= 300 ; (LHS = 0, INFES = 300 ****)

demand(topeka)..  x(seattle,topeka) + x(san-diego,topeka) =G= 275 ; (LHS = 0, INFES = 275 ****)
```

This was generated by: demand(j) .. sum(i, x(i,j)) =g= b(j) ;

LP Listing File (3)

- Part 3: Column Listing
  - Shows variables appearing in the model and where
  - First 3 per block are shown
  - Can be changed with OPTION LIMCOL=nnn;
  - By definition feasible (GAMS will project levels back on their bounds)

```
---- x  shipment quantities in cases

x(seattle,new-york)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
       -0.225   cost
        1       supply(seattle)
        1       demand(new-york)

x(seattle,chicago)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
       -0.153   cost
        1       supply(seattle)
        1       demand(chicago)

x(seattle,topeka)
                (.LO, .L, .UP, .M = 0, 0, +INF, 0)
       -0.162   cost
        1       supply(seattle)
        1       demand(topeka)

REMAINING 3 ENTRIES SKIPPED
```

LP Listing File (4)

- Part 4
  - Model statistics
  - Model generation time: time spent in SOLVE statement generating the model
  - Execution time: time spent in GAMS executing all statements up to the point where we call the solver

```
MODEL STATISTICS

BLOCKS OF EQUATIONS           3     SINGLE EQUATIONS          6
BLOCKS OF VARIABLES           2     SINGLE VARIABLES          7
NON ZERO ELEMENTS            19


GENERATION TIME      =        0.000 SECONDS

EXECUTION TIME       =        0.000 SECONDS
```

LP Listing File (5)

- Solve info
  - Search for ‘S O L’
  - Solver/model status can also be interrogated programmatically
  - Resource usage, limit means time used, limit

```
 S O L V E      S U M M A R Y

     MODEL   transport           OBJECTIVE  z
     TYPE    LP                  DIRECTION  MINIMIZE
     SOLVER  CPLEX               FROM LINE  66

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE             153.6750

 RESOURCE USAGE, LIMIT          0.063      1000.000
 ITERATION COUNT, LIMIT         4         10000
```

Model/Solver Status

MODEL STATUS CODE	DESCRIPTION
1	Optimal
2	Locally Optimal
3	Unbounded
4	Infeasible
5	Locally Infeasible
6	Intermediate Infeasible
7	Intermediate Nonoptimal
8	Integer Solution
9	Intermediate Non-Integer
10	Integer Infeasible
11	Licensing Problems - No Solution
12	Error Unknown
13	Error No Solution
14	No Solution Returned
15	Solved Unique
16	Solved
17	Solved Singular
18	Unbounded - No Solution
19	Infeasible - No Solution

SOLVER STATUS CODE	DESCRIPTION
1	Normal Completion
2	Iteration Interrupt
3	Resource Interrupt
4	Terminated by Solver
5	Evaluation Error Limit
6	Capability Problems
7	Licensing Problems
8	User Interrupt
9	Error Setup Failure
10	Error Solver Failure
11	Error Internal Solver Error
12	Solve Processing Skipped
13	Error System Failure

Model/Solver Status (2)

abort$(m.solvestat <> 1) 'bad solvestat';

```
  model m /all/;
  option nlp=conopt2;
  option mip=cplex;
  option rminlp=conopt2;
  option minlp=dicopt;
*
* solve relaxed model
*
  solve m using rminlp minimizing z;
  abort$(m.modelstat > 2.5) "Relaxed model could not be solved";

*
* solve minlp model
*
  solve m using minlp minimizing z;
```

LP Listing file (6)

- Part 6: messages from solver

```
ILOG CPLEX       BETA  1Apr 22.7.0 WEX 3927.4246 WEI x86_64/MS Windows
Cplex 11.0.1, GAMS Link 34

Optimal solution found.
Objective :         153.675000
```

More information can be requested by OPTION SYSOUT=on;

Note: this part is especially important if something goes wrong with the solve. In some cases you also need to inspect the log file (some solvers don't echo all important messages to the listing file).

LP Listing File (7)

- Part 7: Solution listing
  - Can be suppressed with m.solprint=0;

```
---- EQU demand  satisfy demand at market j

                LOWER          LEVEL          UPPER         MARGINAL

new-york       325.0000       325.0000        +INF             0.2250
chicago        300.0000       300.0000        +INF             0.1530
topeka         275.0000       275.0000        +INF             0.1260

---- VAR x  shipment quantities in cases

                          LOWER          LEVEL          UPPER         MARGINAL

seattle  .new-york          .            50.0000        +INF             .
seattle  .chicago           .           300.0000        +INF             .
seattle  .topeka            .              .            +INF            0.0360
san-diego.new-york          .           275.0000        +INF             .
san-diego.chicago           .              .            +INF            0.0090
san-diego.topeka            .           275.0000        +INF             .
```

Solver Option File

- Write file solver.opt
- Tell solver to use it: m.optfile=1;
- Option file can be written from GAMS

```
$onecho > cplex.opt
lpmethod 4
$offecho

Model m/all/;
m.optfile=1;
Solve m minimizing z using lp;
```

```
--- Executing CPLEX: elapsed 0:00:00.007

ILOG CPLEX       May  1, 2008 22.7.1 WIN 3927.4700 VIS x86/MS Windows
Cplex 11.0.1, GAMS Link 34

Reading parameter(s) from "C:\projects\test\cplex.opt"
>>  lpmethod 4
Finished reading from "C:\projects\test\cplex.opt"
```

Integer Programming

- Combinatorial in nature
- Much progress in solving large models
- Modeling requires
  - Skill
  - Running many different formulations: this is where modeling systems shine
  - Luck
- Often need to implement heuristics

MIP Solvers

- Free solvers:
  - Bdmlp, coinglpk, coincbc,coinscip
- Commercial solvers:
  - Cplex, Xpress (market leaders)
  - XA, Mosek

MIP Modeling

- Difficult, not much automated
- Many MINLPs can be linearized into MIPs.
- Eg

$$z = x \cdot y, \quad x, y \in \{0,1\}$$

can be formulated as:

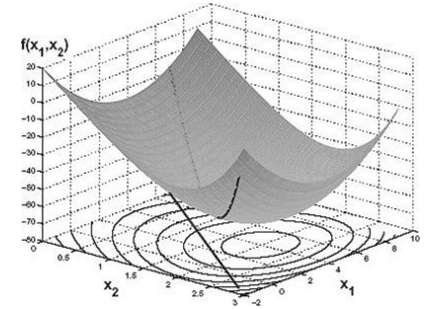
$$z \leq x$$

$$z \leq y$$

$$z \geq x + y - 1$$

$$x, y \in \{0,1\}, z \in [0,1]$$

Nonlinear Programming



- Large scale, sparse, local solvers:
  - Conopt (ARKI)
    - Reliable SQP, 2nd derivatives
    - Scaling, presolve, good diagnostics
    - Often works without options
  - Minos (Stanford)
    - Older augmented Lagrangian code
    - Good for models that are mildly nonlinear
  - Snopt (Stanford, UCSD)
    - SQP based code
    - Inherits much from Minos but different algorithm
  - Knitro (Ziena)
    - Interior point NLP
    - Sometimes this works very well on large problems
  - CoinIpOpt (IBM, CoinOR, CMU)
    - Free, interior point

Special Nonlinear Programming

- PathNLP
  - Reformulate to MCP
- BARON
  - Global solver
  - Only for small models
- Other global solvers:
  - LGO, OQNLP, Lindoglobal
- Mosek
  - For convex NLP and QCP only
- Cplex
  - For QCP

MINLP Solvers

- Free Solvers
  - CoinBonmin
- Dicopt
- SBB
- AlphaEcp
- Baron, lgo, oqnlp (global)

NLP Modeling

- Models fail mostly because of:
  - Poor starting point
    - Specify X.L(i)=xx; for all important nonlinear variables
  - Poor scaling
    - You can manually scale model use x.scale, eq.scale
  - Poorly chosen bounds
    - Choose x.lo,x.up so that functions can be evaluated
- Note: changing bounds can change initial point

NLP Modeling

- Minimize nonlinearity
- Measure
  - --- 429 nl-code 30 nl-non-zeroes
- Example:

e1.. Z =e= log[sum(i,x(i))]

X(i) is non linear

e1.. z =e= log(y);
e2.. y =e= sum(i,x(i));

X(i) is linear

Additional advantage:
We can protect log by y.lo=0.001;

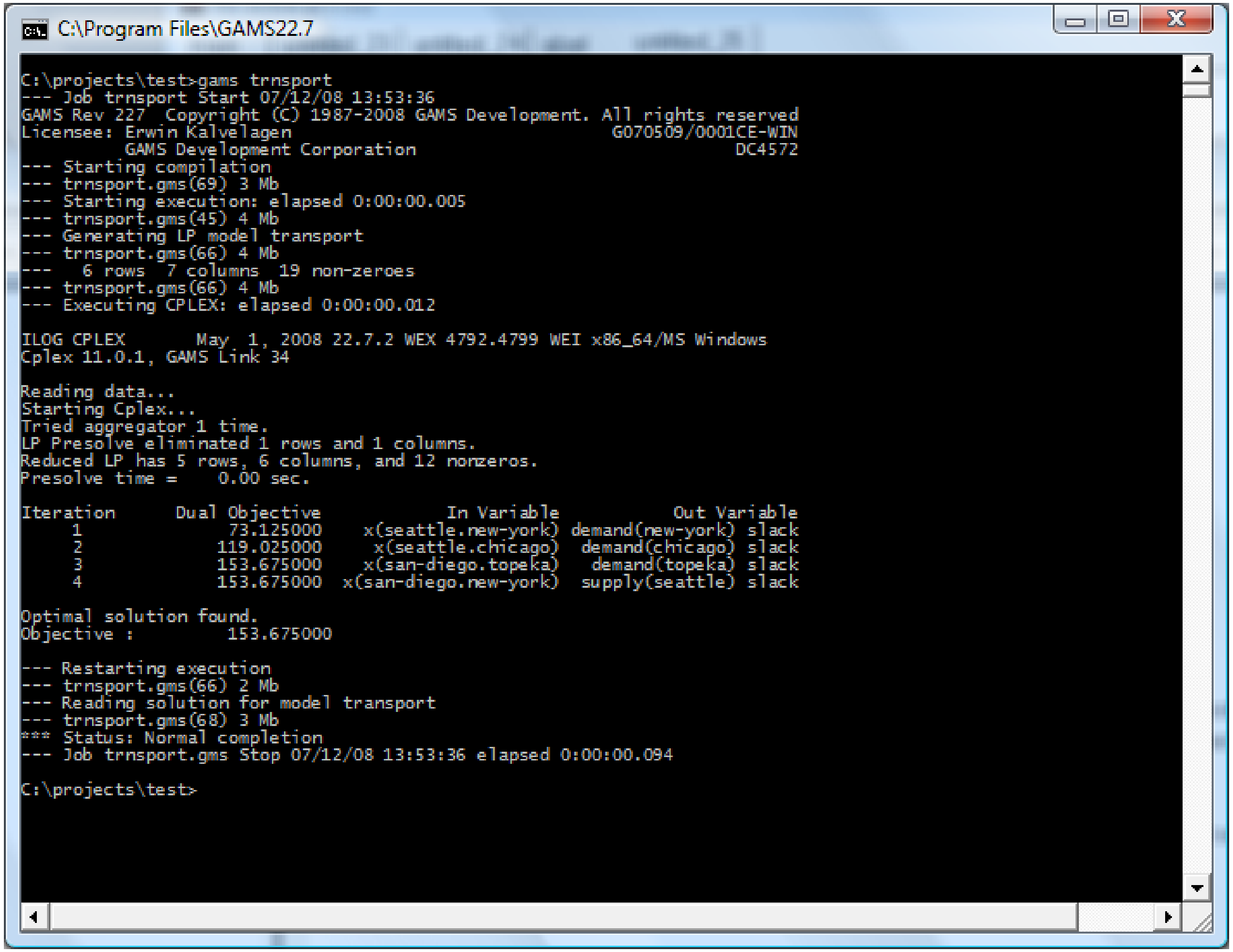
Functions

Function	Allowed In equations	Notes
abs	DNLP	Non-differentiable, use alternative: variable splitting
execseed	no	Seed for random number generation. Can also be set.
Exp,log,log2,log10	NLP	Add lowerbound for log
Ifthen(cond,x,y)	DNLP	Non-differentiable, use binary variables
Min(x,y),max(x,y,z), smin(i,..), smax(i,…)	DNLP	Non-differentiable, use alternative formulation
Prod	NLP	
Sum	LP/NLP	
Round, trunc, fract	no	
Sqr,sqrt,power	Yes	Protect sqrt with lowerbound
Power(x,y), x**y	NLP	Power: integer y x**y = exp(y*log(x)), add x.lo=0.001;
Cos,sin,tan,arccos,arcsin,arctan,arctan2,cosh,sinh,tanh,	NLP	

Functions (2)

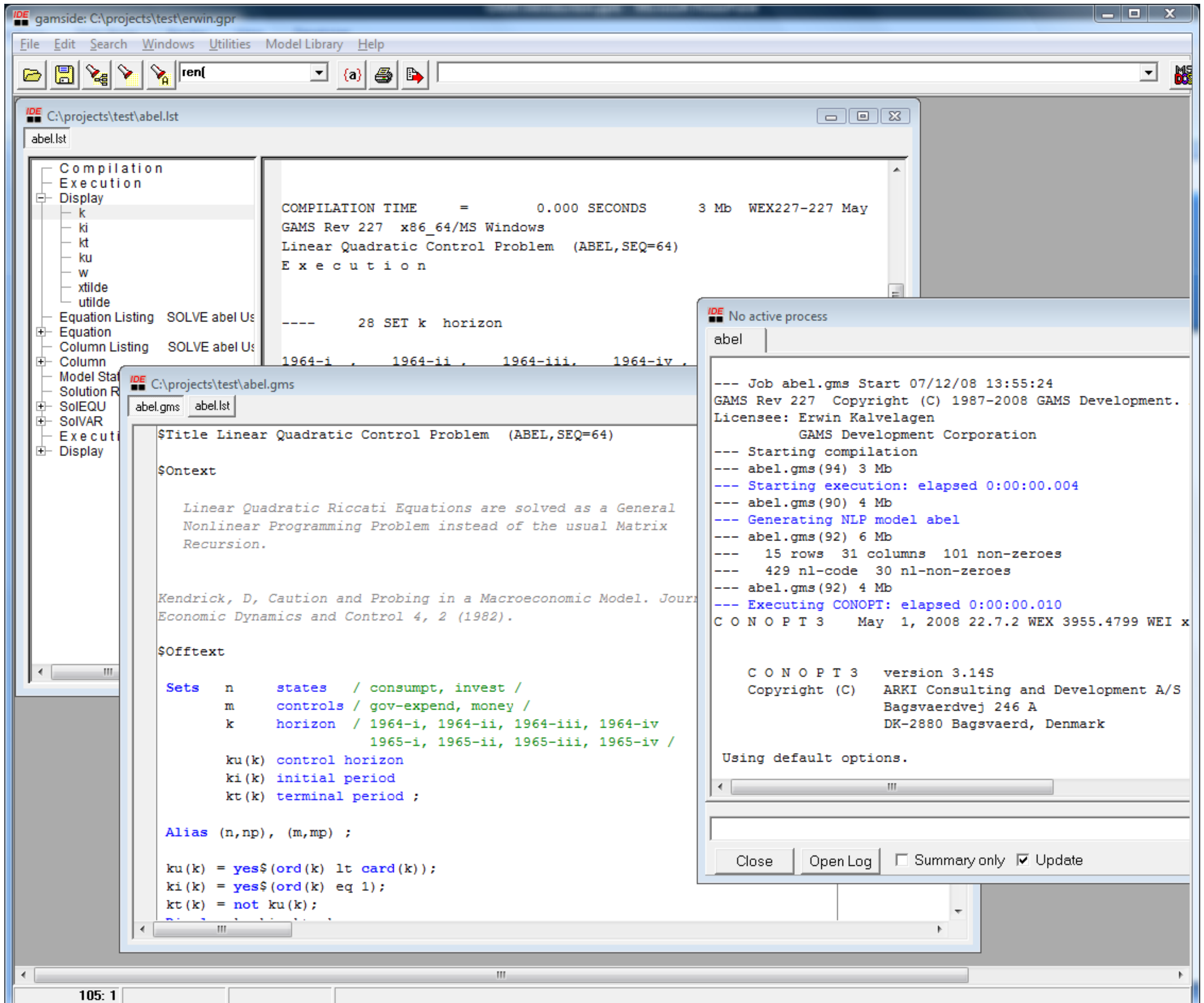
Function	Allowed In equations	Notes
Fact	no	In equations use gamma
Gamma,Beta,BetaReg,Gamma Reg, LogGamma,LogBeta	DNLP	
Binomial(x,y)	NLP	Generalized binomial function
Errorf	NLP	Error function. Inverse not available: use equation: z =e= errorf(x) to find x.
Mod	No	
Normal, uniform, uniformint	No	Random number generation
Pi	Yes	
Edist, entropy, ncpf, ncpcm, poly,	Yes	Not often used
Calendar functions	no	

Command Line Version



1. Edit .gms file
2. Run GAMS
3. View .lst
4. Go back to 1.

IDE


gamside: C:\projects\test\erwin.gpr
File Edit Search Windows Utilities Model Library Help
C:\projects\test\abel.lst
abel.lst
Compilation
Execution
Display
k
ki
kt
ku
w
xtilde
utilde
Equation Listing SOLVE abel Us
Equation
Column Listing SOLVE abel Us
Column
SolEQU
SolVAR
Display
COMPILATION TIME = 0.000 SECONDS 3 Mb WEX227-227 May
GAMS Rev 227 x86_64/MS Windows
Linear Quadratic Control Problem (ABEL,SEQ=64)
E x e c u t i o n
---- 28 SET k horizon
C:\projects\test\abel.gms
abel.gms
abel.lst
$Title Linear Quadratic Control Problem (ABEL,SEQ=64)
$Ontext
Linear Quadratic Riccati Equations are solved as a General
Nonlinear Programming Problem instead of the usual Matrix
Recursion.
Economic Dynamics and Control 4, 2 (1982).
$Offtext
Sets n states / consumpt, invest /
m controls / gov-expend, money /
k horizon / 1964-i, 1964-ii, 1964-iii, 1964-iv
1965-i, 1965-ii, 1965-iii, 1965-iv /
ku(k) control horizon
ki(k) initial period
kt(k) terminal period ;
Alias (n,np), (m,mp) ;
ku(k) = yes$(ord(k) lt card(k));
ki(k) = yes$(ord(k) eq 1);
kt(k) = not ku(k);
No active process
abel
--- Job abel.gms Start 07/12/08 13:55:24
GAMS Rev 227 Copyright (C) 1987-2008 GAMS Development.
Licensee: Erwin Kalvelagen
GAMS Development Corporation
--- Starting compilation
--- abel.gms(94) 3 Mb
--- Starting execution: elapsed 0:00:00.004
--- abel.gms(90) 4 Mb
--- Generating NLP model abel
--- abel.gms(92) 6 Mb
--- 15 rows 31 columns 101 non-zeroes
--- 429 nl-code 30 nl-non-zeroes
--- abel.gms(92) 4 Mb
--- Executing CONOPT: elapsed 0:00:00.010
C O N O P T 3 version 3.14S
Copyright (C) ARKI Consulting and Development A/S
Bagsvaerdvej 246 A
DK-2880 Bagsvaerd, Denmark
Using default options.
Close
Open Log
Summary only
Update
105: 1

IDE Editor

- Syntax coloring can help detect syntax errors very early.
- Block commands are often useful

```
Sets   n     states   / consumpt, invest
       m     controls / gov-expend, money /
       k     horizon  / 1964-i, 1964-ii, 1964-iii, 1964-iv
                        1965-i, 1965-ii, 1965-iii, 1965-iv /
```

```
Sets   n     states   / consumpt, invest /
       m     controls / gov-expend, money /
       k     horizon  / 1964-i, 1964-ii, 1964-iii, 1964-iv
                        1965-i, 1965-ii, 1965-iii, 1965-iv /
```

```
  table fert(p2,c,z) fertilizer applications  (kg per acre)
                      nwfp   pcw   pmw   prw   psw  scwn  scws  srwn  srws

nitrogen.basmati             26.6  26.6  21.9  23.4
nitrogen.irri                39.4  39.4  23.3  26.8  68.6  61.5  48.9  41.7
nitrogen.cotton       26.7   42.3  30.0  30.0  19.7  55.0  54.9  39.6  39.6
nitrogen.maize        27.0   27.1  27.1  23.6  19.5  42.0  42.0  42.0  42.0
nitrogen.kha-fod      21.0   25.3  18.1  19.2  18.4  49.0  49.0  49.0  49.0
nitrogen.wheat        46.8   40.9  36.5  32.2  33.3  54.9  53.5  29.5  39.8
nitrogen.rab-fod      10.0   25.3  18.1  19.2  18.4  28.0  28.0  28.0  28.0
nitrogen.sc-mill      83.4   44.8  63.2  33.9  33.9  65.1  65.1  65.1  65.1
nitrogen.sc-gur       24.0   19.0  19.0  19.0  19.0  28.0  28.0  28.0  28.0
nitrogen.onions       60.6   48.0  48.0  48.0  48.0  70.7  70.7  70.7  70.7
nitrogen.potatoes     48.6   38.5  38.5  38.5  38.5  56.7  56.7  56.7  56.7
nitrogen.mus+rap      33.6   30.8  30.8  30.8  30.8  45.4  45.4  45.4  45.4
nitrogen.chilli       48.6   38.5  38.5  38.5  38.5  56.7  56.7  56.7  56.7
nitrogen.orchard      60.0   47.5  47.5  47.5  47.5  70.0  70.0  70.0  70.0
```

IDE Tricks

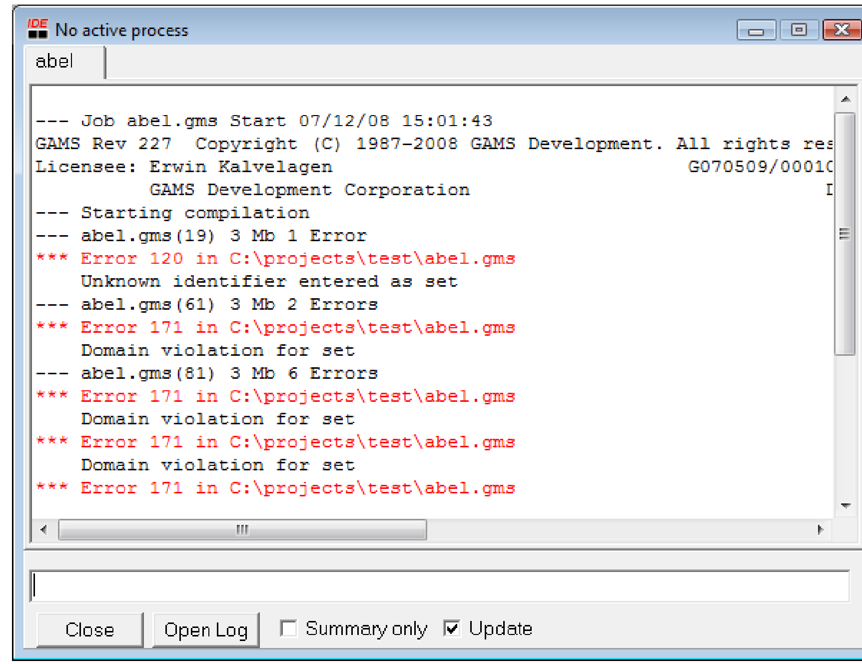
- F8 to find matching parenthesis
- Search in files

Project File

- The project file determines where files (.gms,.lst,.log) are located.
- Start new model by creating new project file in new directory

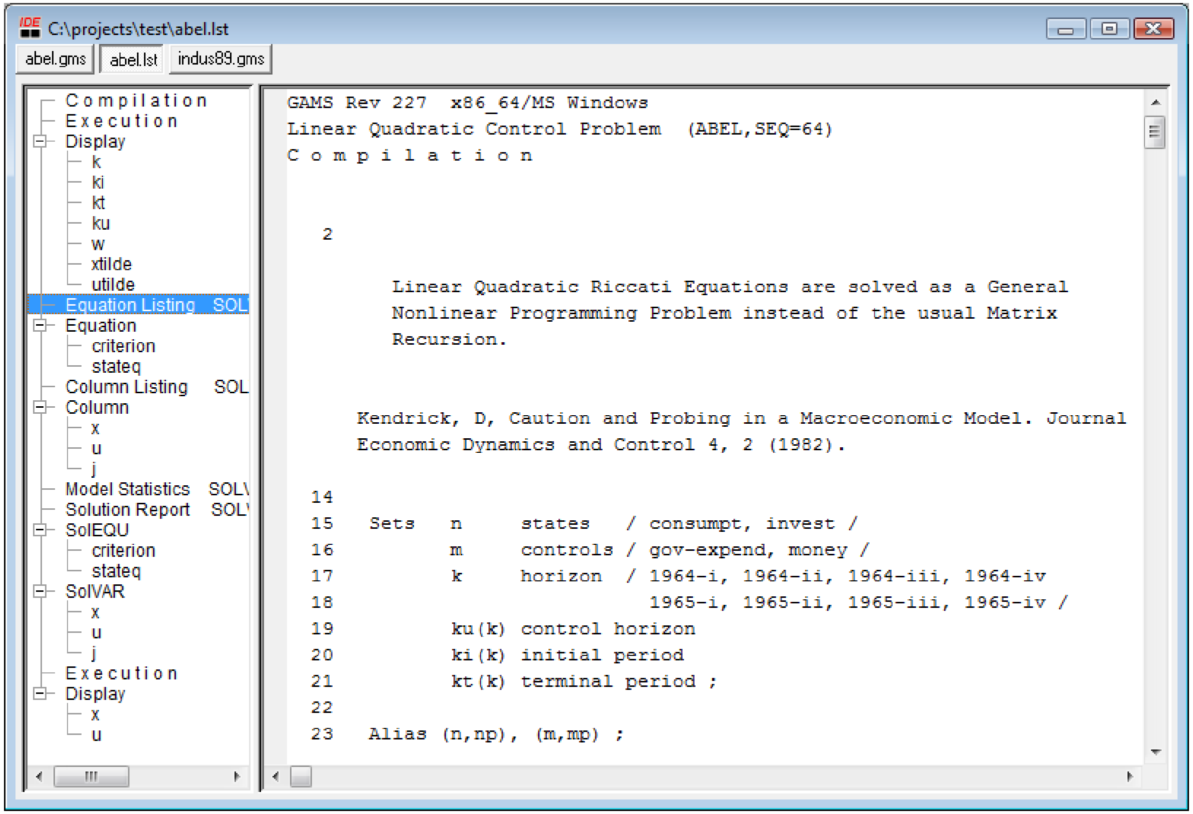
Edit,Run,…

- After hitting Run Button (or F9), process window shows errors
- Clicking red line brings you to location in .gms file
- Clicking back line bring you to location in .lst file
- This is only needed for obscure errors



Lst File Window

- Use tree to navigate
- Search for 'S O L' to find 'S O L V E S U M M A R Y'

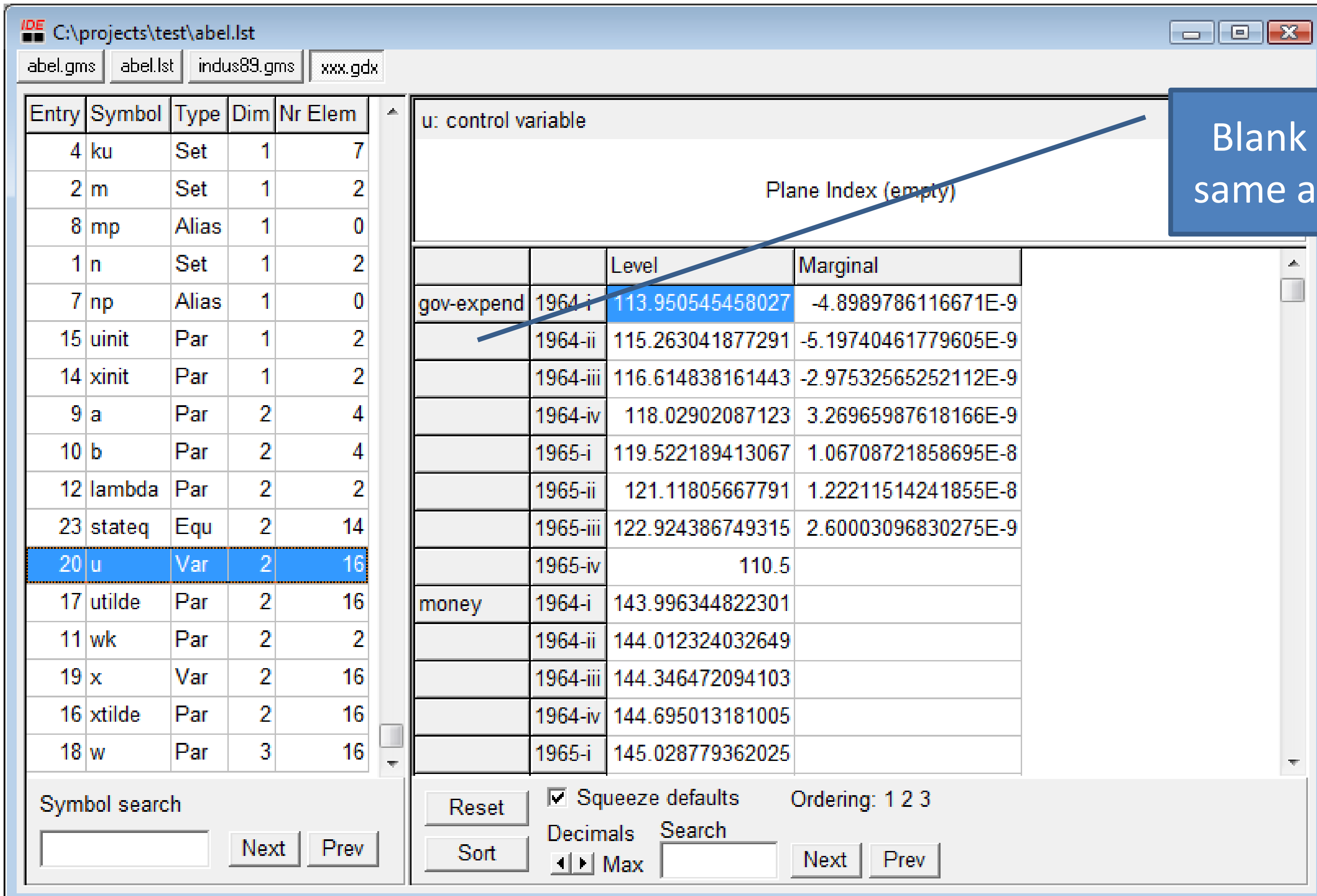


Debug Models

- Use DISPLAY statements
- Use GDX=xxx on command line
- Then click on blue line


```
--- abel.gms(92) 2 Mb
--- Executing after solve: elapsed 0:00:00.102
--- abel.gms(94) 3 Mb
--- GDX File C:\projects\test\xxx.gdx
*** Status: Normal completion
--- Job abel.gms Stop 07/12/08 15:14:47 elapsed 0:00:00.104
```

GDX Viewer



Blank means same as above

GDX Cube

sylds: straw yield and seed data

Plane Index (empty)

basmati	pmw	bullock	standard	standard	straw-yld	2.33
					seed	6.4
		semi-mech			straw-yld	2.33
					seed	6.4
	pcw	bullock			straw-yld	2.33
					seed	6.4
		semi-mech			straw-yld	2.33
					seed	6.4
	psw	bullock			straw-yld	2.12
					seed	6.4
		semi-mech			straw-yld	2.12
					seed	6.4

On the plane

sylds: straw yield and seed data

bullock	heavy	la-plant
		qk-harv
	january	standard
		la-plant
		qk-harv

Column headers

		straw-yld	seed
wheat	nwfp	1.3	40.1
	pmw	1.3	34.8
	pcw	1.5	34.8
	psw	1.5	34.8
	prw	1.6	34.8
	scwn	1.5	49.8
	srwn	1.5	49.8
	scws	1.5	49.8
	srws	1.5	49.8

Row headers

Index positions can be placed:

1. On the plane
2. On the left (row header)
3. On the top (column header)

Generating GDX files

- From command line (gdx=xxx)
- $gdxout (not used much)
- Execute_unload ‘xxx.gdx’,a,b,x;
- Or via some external tool:
  - Gdxxrw can create a gdx file from an Excel spreadsheet
  - Mdb2gms can create a gdx file from an Access database
  - Sql2gms can create a gdx file from any sql database

Reading GDX file

- $gdxin

```
Set i;
Parameter p(i);

$gdxin a.gdx
$load i
$load p

Display i,p;
```

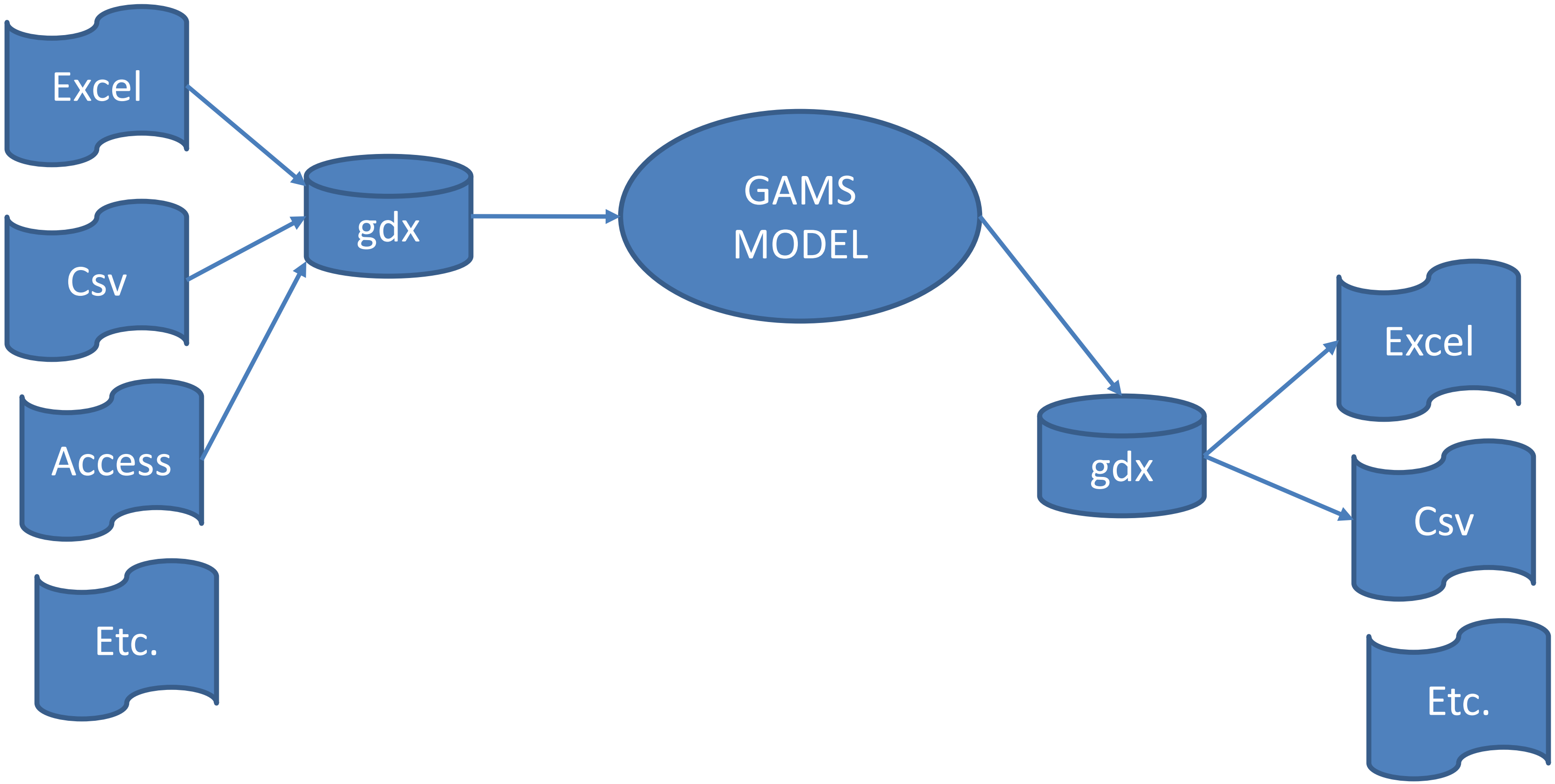
Compile time

- Execute_load

Execution time

```
set i  /i1*i3 /;
alias (i,j);

table a(i,j) 'original matrix'
       i1      i2      i3
i1     1       2       3
i2     1       3       4
i3     1       4       3
;

parameter inva(i,j) 'inverse of a';

execute_unload 'a.gdx',i,a;
execute '=invert.exe a.gdx i a b.gdx inva';
execute_load 'b.gdx',inva;

display a,inva;
```

GDX is hub for external I/O



Gdxxrw: read xls

trnsport.xls [Compatibility Mode]

	A	B	C	D
1		new-york	chicago	topeka
2	seattle	2.5	1.7	1.8
3	san-diego	2.5	1.8	1.4
4				

```
$onecho > x.txt
trace=2
i=trnsport.xls
par=c
rng=A1
rdim=1
cdim=1
$offecho

$call =gdxxrw.exe @x.txt

parameter c(*,*);
$gdxin trnsport.gdx
$load c
display c;
```

```
----     15 PARAMETER c

            new-york     chicago      topeka

seattle        2.500       1.700       1.800
san-diego      2.500       1.800       1.400
```