

Course Materials from GAMS 2 Class  
**Advanced Bases**  
(Fixmodel.pdf ch 11, GAMSBAS.pdf)

Bruce A. McCarl

Specialist in Applied Optimization  
Professor of Agricultural Economics, Texas A&M  
Principal, McCarl and Associates

[mccarl@tamu.edu](mailto:mccarl@tamu.edu)  
[mccarl@bihs.net](mailto:mccarl@bihs.net)  
[agrinet.tamu.edu/mccarl](http://agrinet.tamu.edu/mccarl)

409-693-5694  
409-845-1706

## GAMS and Advanced Bases

### Concept

(Fixmodel.pdf ch 11, GAMSBAS.pdf)

The solution to a linear programming problem generally has one nonzero variable for each constraint. This means in a model with  $M$  constraints all one really needs to know is exactly which  $M$  variables are nonzero.

Generally linear programming solvers take three or more times as many iterations as the number of constraints to reach a solution. The linear programming solver guesses which  $M$  variables should be in the solution then one by one inserts better variables into the solution until it reaches the ultimate, optimal set.

If one can supply a fairly accurate guess for the solution then solution time can be reduced. An advanced basis is a user defined suggestion on variables in the solution. A good advanced basis can reduce solution time by more than an order of magnitude. I've often seen solution time reduced from something like **36 hours to 1 hour**.

Often in applied linear programming work the advanced basis will be suggested based on the solution of a related model and will be close to the final solution.

GAMS and Advanced Bases  
**Bases in GAMS**  
([Fixmodel.pdf](#) ch 11, [GAMSBAS.pdf](#))

Advanced bases are not generally present in GAMS for the first solution of model.

Rather advanced bases are used for all repeated solutions of the model.

When a solve is executed GAMS forms a suggested bases from the stored solution information from the last time the model was solved.

Such information is typically not available the first time a model is solved unless the user has explicitly provided it.

Here I cover how to provide such information and also discuss some of the difficulties that may arise when using advanced bases.

## GAMS and Advanced Bases

### GAMS use of a Basis

GAMS forms a basis from marginal and level values

Information used are:

- a) Non zero levels for variables;
- b) Non zero values equal to the upper or lower bounds for variables at their bound. Variables held at bound also have non zero marginals;
- c) Zero levels and non zero marginals for variables not in the basis;
- d) Zero marginals for nonbinding constraints; and
- e) Non zero marginals for binding constraints.

You can provide a basis by

repeatedly solving

guessing

using GAMSBAS

## GAMS and Advanced Bases

### Repeated Solves

([twotran.gms](#))

Suppose we examine the effect of an advanced basis by solving a model twice and seeing what happens the second time vs. solving the model alone. In particular in a transportation model we solve it first with the conventional transportation costs than we triple transportation costs and solve it again. The code to do this appears below ([twotran.gms](#))

```
SOLVE FIRM USING LP MAXIMIZING NETINCOME;  
TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)  
  =TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)*3;  
SOLVE FIRM USING LP MAXIMIZING NETINCOME;
```

The first solution of the model takes 19 iterations the second one takes 11. The second model solved all by itself ([twotrana.gms](#)) takes 15.

The reason for fewer iterations when a second solve is issued is that GAMS suggested an advanced basis using the saved solution information from the first solve. In larger models with less radical changes defining the second model solution time is often reduced in a much more substantial manner.

## GAMS and Advanced Bases

### Steps for Using GAMSBAS

1. Run a model from which you wish to save the basis information causing GAMSBAS to save that information for you
  - A. Insert the command  
option LP=GAMSBAS;  
Before the solve statement
  - B. This will cause a file to be saved which contains the model solution information.

The file name will be your model name plus the suffix .bas. If your model is named **transport** then **transport.bas** will be saved. If **farmmod** then **farmmod.bas** will result.

2. Run the related model in which you wish the basis to be used. After variable and equation definitions but before the solve statement add in the command

```
$include transport.bas
```

or the filename of the saved basis file.

# GAMS and Advanced Bases

## Simple Example Generating a Basis Using GAMS BAS (Simple.gms)

```

sets
    var          /x1*x3/
    constraint   /r1*r3/;

variables
    objfun;
positive variables x(var);
equations
    objective
    resource(constraint);
parameter
    objcoef(var) objective
function coefficients
    /x1 30,x2 20,x3 10 /;
parameter
    rhs(constraint)
constraint rhs's
    /r1 10,r2 2,r3 8/;
table   amatrix(constraint,var) aij matrix
        x1      x2      x3
    r1      2      1
    r2      1      1      -1.5
    r3              -1      1;
objective..   objfun =e= sum(var,
objcoef(var) * x(var));
resource(constraint)..
    sum(var, amatrix(constraint,var)*
x(var)) =l= rhs(constraint) ;
model mymodel /all/
option limrow=0; option limcol=0;
option lp=GAMSBAS;
solve mymodel using lp maximizing objfun;

```

Solution takes two iterations

# GAMS and Advanced Bases

## Simple Example Using a Basis from GAMSBAS

### Resultant GAMSBAS Basis File ([simple.bas](#))

```
OBJECTIVE.m =      1.000000000000      ;
RESOURCE.m('r1') =      25.0000000000      ;
RESOURCE.m('r2') =      5.000000000000      ;
OBJFUN.l =      260.0000000000      ;
X.l('x1') =      6.000000000000      ;
X.l('x2') =      4.000000000000      ;
$offlisting
X.m('x3') =      -7.500000000000      ;
```

### Just GAMS replacement statements

### Modifying our base model ([Simpleb.gms](#))

```
model mymodel /all/
option limrow=0; option limcol=0;
option lp=bdmlp;
$include simple.bas
mymodel.optfile=1;
solve mymodel using lp maximizing objfun;
```

Solution takes zero iterations

Thus given the basis we did no work at all.

## GAMS and Advanced Bases

### Guessing

One does not have to use GAMS BAS and may not be able to for the very first solve of a model.

One can try to guess at a solution by specifying

nonzero marginals for the constraints that are felt to be binding and

nonzero levels for the variables that are felt to be nonzero in the solution as follows.

```
OBJECTIVE.m           =      1 ;
RESOURCE.m ("R1")     =      1 ;
RESOURCE.m ("R2")     =      1 ;
RESOURCE.m ("R3")     =      0 ;
OBJFUN.l              =      1 ;
X.l ("X1")            =      1 ;
X.l ("X2")            =      1 ;
X.m ("X3")            =      1 ;
```

This basis if included would cause our simple problem to solve in zero iterations

## GAMS and Advanced Bases

### Why use a GAMSBAS Basis

Above we were using a GAMSBAS basis for the same model that the basis was generated from. Some might ask: Why would this ever be done?

Suppose we had a model that we're working on and at some point we find we need to revise the data in the initial stages before the model statements.

Two alternatives would be available to us.

We could include a new table of revised data and use replacement statements to replace the old data

We could go back fix the original data and rerun the model from scratch.

The obvious deficiency of the second strategy is that we lose our basis. The first strategy results in a model that has the most current data spread throughout.

I find the problem with the first alternative more odious and use GAMSBAS to overcome the shortcoming of the second .

GAMS and Advanced Bases  
**Problems with a BASIS**  
(Fixmodel.pdf ch 11, GAMSBAS.pdf)

Provision of an advanced basis does not always help. This can be particularly true in a repeated set of solutions with a radically altered model.

Symptoms - Solver terminates saying  
sorry guys we seem to be stuck or  
after 3 factorizations the basis is singular.  
The solver may also not make significant progress.

Underlying causes

Altered model size caused by conditionals  
**Elimination of key variables in the basis**  
**Deletion of constraints**  
Radical alteration of model coefficients

Solution

Don't change model size eliminate things  
economically – rather than eliminate a variable  
make it very high cost - rather than eliminate a  
constraint make it nonbinding.

Revert to a saved basis

Tell the solver to dump the basis using the BRATIO=1  
option

## GAMS and Advanced Bases

### Comments on using Advanced Bases

Provision of advanced bases does not always reduce solution times.

The slide immediately above talks about problems we may have in LPs. We may also have a lack of success when dealing with nonlinear or mixed integer problems.

Mixed integer programming problem bases may not be all that successful as what's really needed is storage of the branch and bound tree. The only real success from bases may be giving an initial good **objective function bound** and getting the linear programming part of the solver started relatively quickly.

In nonlinear models advanced bases are often helpful but there are cases where they do not contribute to shortened solution time. There are no general rules here, only experience with a problem will show how effective the provision of an advanced basis is.

Nonlinear programming is the place where I've had **largest gains in terms of solution time reduction with things being reduced from day to hours.**

## GAMS and Advanced Bases

### Save and restart (Fixmodel.pdf ch 11)

A final strategy regarding bases involves using saves and restarts.

It's quite common for me to have a model with its initial data and bring up to its initial solve then use save asking GAMS to save model status .

In turn I restart the model from the saved status file and carry out alternative runs.

This means that the alternative runs all start from the basis implied by the first model solution. An example of this is inherent in the following dos bat file.

```
command /c GAMS frstpart s=f1  
command /c GAMS nextpart r=f1
```