So Your GAMS Model Didn't Work Right
A Guide to Model Repair

Bruce A. McCarl
Professor of Agricultural Economics
Texas A&M University
College Station TX
mccarl@tamu.edu
agrinet.tamu.edu/mccarl

Acknowledgments

A large number of people have contributed to this document Alex Meeraus and Wilfred Candler directly taught me many of the techniques. However, the contributions of Erwin Kalvelagen, Ramesh Ramen, Pete Stacy all of GAMS corporation to my knowledge stock were not inconsiderable. Bill Nayda and Chi-Chung Chen contributed in terms of technique development. Harvey Greenberg and Rick Rosenthal also contributed ideas at various times. In terms of the writing Erwin Kalvelagen, Chi-Chung Chen, Uwe Schneider, Darius Adams, Tasana Gillig, Heng-Chi Lee and Bill Nayda made numerous suggestions and corrections.

So Your GAMS Model Didn't Work Right
A Guide to Model Repair

Bruce A. McCarl

List of Tables and Figures

So Your GAMS Model Didn't Work Right

A Guide to Model Repair

Bruce A. McCarl

Chapter 1 Introduction

GAMS models rarely work perfectly the first time they are submitted. This document provides a guide to fixing imperfectly working models. Modeling problems span the gamut from GAMS compilation and JCL difficulties through improper model formulation and excessive use of computing resources. Repair of these problems involves many potential types of activities including changing JCL, correcting typing or syntax errors, recoding GAMS equations for improved performance, or altering model structure. This document discusses problem symptoms, ways of finding out more about problems, tools to aid in structural diagnosis and other useful utilities.

In presenting this material we adopt a two-part approach. First, we discuss the symptoms defining: a) what type of problems can arise; b) how to recognize problems; and c) aspects of the GAMS output that provide additional information. Second, we discuss a set of remedies which may be employed. We also provide a cross reference table that gives a prioritized list of remedies to try given a problem has arisen.

The presentations in this document spans multiple levels of GAMS and modeling expertise. For example, the discussion of compiler error repair is for the novice, while the discussion of execution time reduction is aimed at experienced GAMS users. Similarly, the model structural error detection section is oriented toward experienced modelers.

The reader should note that in presenting this material we include a number of GAMS features which are only present in GAMS 2.25 versions above release 90. Thus, those with older systems may need to update in order to use all referenced features.

Finally, we should note that in construction of this document our philosophy is oriented toward fixing a model by oneself. Users may find that the remedies here are inadequate or may not be able to find a way to a problem by themselves. In such a case, one can seek help through the GAMS mailing list which hosts an email based dialog between users. One can use this forum to address questions to others including a number of very experienced users and possibly get answers. The list may be joined by sending a message to GAMS-L@vm.gms.de.

Part I

Common Problems

Chapter 2 Symptoms -- What Can Go Wrong?

The GAMS modeler is far from finished when the GAMS input stream has been typed and submitted for initial processing. Problems of varying degrees of subtlety can occur. Here we provide a list of potential problems and reference some remedies.

2.1 GAMS Didn't Get Started -- Job Control problems

One can submit a job and get messages such as "file not found", "unknown procedure" etc. This indicates job control language problems. This topic will not be extensively discussed herein as the problems usually involve either bad typing, improper file locations, inadequate file reference paths, or an incorrect GAMS installations among other possibilities. Furthermore the remedies are often computer system specific and we are attempting a computer system independent presentation. Such problems are often best resolved by retyping the command or contacting a local computer expert. The GAMS user manual (Brooke, Kendrick and Meeraus along with subsequent editions including the one now on the web at www.gams.com), the GAMS installation notes for a computer system, local README and TXT or DOC files are the best sources of hints to repair job control problems. The GAMS mailing list (gams-L@vm.gmd.de)also provides a way to get advice from experienced users. The are only a few instances below where such material is covered.

2.2 GAMS --Compilation Problems

Once the GAMS job is properly submitted the first stage, which virtually always generates errors, is compilation. Users watching the execution of a program are sometimes dismayed to get the message: **COMPILATION ERRORS** with an indication of numerous errors. Chapter 6 covers finding and fixing compilation errors.

2.3 GAMS Execution Errors

After passing the compilation stage, GAMS executes all calculations preceding the solve, generates the model, sends it to the solver then does post solution calculations. During these stages execution errors may occur. These generally involve two causes: 1) mathematical difficulties due to division by zero or improper exponentiation; 2) generation of an obviously infeasible model; or 3) solver failure. Suggestions for diagnosis and repair appear in Chapter 7.

2.4 Computer Capacity

GAMS can terminate because it reached computer hardware limits. Such limits include running out of disk storage space or RAM memory. Chapter 10 covers capacity expansion and problem size reduction.

2.5 GAMS Capacity

One can run into problems with GAMS capacity. These generally involve limits on the problem size or the amount of code that is allowed in a particular module. This requires that one: a) insure that the requested solver is available in other than demonstration mode, b) cut down the problem size if needed, or c) instruct GAMS to make provisions for a larger problem. In the latter case, the GAMS printout usually makes suggestions on how to repair the problem. For example one might be told to add the option CODEX=1 to DOS implementations or -CODEX 1 in UNIX implementations.

The capacity constraints for GAMS solvers as mentioned Brooke, Kendrick and Meeraus are very out of date. GAMS can handle much larger problems than those mentioned. There are capacity limits still imposed, but to the authors knowledge are not in the available literature and are quite large. Some of these limits are machine specific so no attempt will be made here to

specify limits as that information too would soon be out of date.

2.6 Solver Stopped Saying it Ran Out of Resources

Solvers may also fail because they exceed a resource (time) or iterations limit. In such cases one can repair the problem using the OPTION statements involving RESLIM or ITERLIM as discussed in Chapter 10. There also are solver specific iteration or other resource limits which they can reach. See the solver manuals or the treatments below on GAMS solver option files for details.

2.7 The Solver Started But Terminated Unexpectedly

GAMS may terminate with a message indicating a lack of progress. This indicates numerical problems. When solvers fail because of numerical difficulties or use an unrealistically large amount of resources to make little progress, the modeler is often in an awkward position. However, several actions may alleviate the situation.

First, examine whether the model specification is proper. The section on structural checking in chapter 8 gives some techniques for examining model structure.

Second, try to resolve degeneracy induced cycling as discussed in chapter 10. Apparently, even the best solvers can become stuck or iterate excessively in the presence of massive degeneracy. Some solvers in GAMS, at least OSL and CPLEX, contain automatic degeneracy perturbation procedures while others try to manage cycling through other schemes. When this is not the case, our experience indicates one should use an a priori degeneracy resolution scheme as discussed below. We have always observed reduced solution times with this modification.

Third, a solver may fail because of poor scaling. Often one needs to rescale the model to narrow the disparity between coefficient magnitudes. Scaling techniques and tools to help in

scaling are discussed in chapter 10.

Fourth, the model may have problems with the basis. Chapter 11 discusses the issue and possible remedies.

All of the preventive techniques for avoiding solver failures can be used before solving a model. Modelers should check structure and consider scaling before attempting model solutions. However, they should not usually employ degeneracy resolution until they identify a problem.

2.8 GAMS Just Stopped

One may encounter the situation where GAMS begins but stops for no apparent reason. There may be several causes for this. First, one may in fact has reached some kind of limit inside GAMS and needs to investigate the listing file farther to see what is happening when alter GAMS option or the solver option file. Second, one may have run out of memory. One may have reached some a kind of unusual termination. Here one needs to rerun the model and observe messages on the screen. One can also redirect the screen output to a file using the option LO=2 and examine that file (ordinarily if they call the GAMS input my model the LO = 2 file will direct output to MYMODEL.LOG). Chapter 10 discusses repair of the types of problems which may cause this failure.

2.9 Model Declared Infeasible or Unbounded

GAMS can terminate the solve process with the message that the model is infeasible or unbounded. This situation often marks the beginning of a difficult exercise directed toward finding the underlying cause of such a result. There are several techniques one can use when this occurs as discussed in the beginning of Chapter 9. First, examine the LST file and observe the messages about the infeasibility. Second, check the model structure looking for obvious model

formulation defects. Third, use artificial variables to identify the equations involved with an infeasibility. Fourth, use large upper bounds to find the variables involved with the unboundedness. Finally, use the techniques called budgeting and row summing.

2.10 The Model Reached an Optimal, Nonsensical Solution

Unfortunately, optimal solutions can be unrealistic. A declaration of optimality means the problem has a mathematical optimum. However, mathematical optimality does not necessarily imply real world consistency (Heady and Candler). Usually, improper problem specification or assumption violations may cause unrealistic solutions. Cases arise where the model solution is improper because of: a) omitted constraints or variables; b) errors in coefficient estimation; c) algebraic errors; or d) coefficient placement errors.

Basically, a model may be judged improper because of incorrect valuation or allocation results. Valuation difficulties arise from the reduced cost or shadow price information. Such items take on values when primal reduced costs are formed. Allocation difficulties arise when the slack or decision variable values are unrealistic. The values of these items are formed through the constraint interactions. Thus, to diagnose the cause of the unrealistic solution, one investigates either the reduced costs associated with the nonbasic primal variables or the calculations inherent in the primal constraints. Two techniques are presented in Chapter 9, one for the investigation of reduced costs, which we call "budgeting"; and another for the reconstruction of the constraint calculations, which we call "row summing."

2.11 Model Worked But Took Excessive Time or Memory

GAMS may unnecessarily execute slowly or use excessive memory. There are several ways of speeding up execution. These divide into two classes: speeding up solution processes

and speeding up non solution calculations. One can also use the GAMS Profile command to find speed problems. Memory use also can be managed. Chapters 10 and 12 contain material on the resolution of such problems.

2.12 Models Worked But Created Excessive Output

New users to GAMS can solve models and in effect lose the answer because all the output GAMS creates. Output can be managed by a) suppressing row and column listings using the options statement on LIMROW and LIMCOL; b) eliminating the cross reference list by using the $ sign control

$OFFSYMLIST,OFFSYMXREF

c) eliminating the presence solution printout by using OPTION SOLPRINT=OFF or d) by using the onlisting and offlisting syntax. Similarly, one must make sure that the active display statements are those wished. Each of these is discussed in the original GAMS manual (Brooke, Kendrick and Meeraus). One can also manage output by using GAMS put statements to save a compact output file or by using the JCL save and restart options in conjunction with a small file which carefully manages the display output. Chapter ?? covers output management.

2.13 Items Eliminated from the Model Appear in the Output

Sometimes one needs to solve a model repeatedly and in this repeated solution process eliminate various variables' and/or equations then find that these variables' and/or equations still appear in the output. This occurs because by default GAMS merges rather than replaces the solution, thus old values will be retained for eliminated features. A GAMS option statement (OPTION SOLPRINT=REPLACE) for the most part will eliminate this difficulty(note a difficulty remains in the variables section when individual entries in a variable block are eliminated with $

commands users now need to manually zero variables and levels in repeat solutions if difficulties are encountered). Chapter 18 discusses this case.

2.14 Items Appear as Zeros in the Solution Which Should Not Be

Models which are not well scaled can generate solutions where for example shadow prices which should be non zero are reported as EPS. Then one should consider lowering the optimality tolerance in the solver option file. For example in MINOS shadow prices and reduced costs are zeroed if they fall below the optimality tolerance times the norm of the dual vector.

2.15 A Preanalysis Procedure Concluded the Problem was Improper

The presolve analysis in OSL and some other solvers prescan the model formulation eliminating equality constraints , examining the feasibility of constraints and converting constraints into upper or lower bounds to make the problem simpler to solve. On occasions these procedures can solve the problem and/ or conclude it is infeasible. Occasionally unusual output occurs and users may not wish this to occur. In such cases one should suppress the presolve option. The solver manuals discuss how to do this.

Chapter 3 Matching Symptoms to Solutions

This section will contain a prioritized listing of what to do given a symptom has arisen in the format of a table such as the one attached

Table 3. Priorities of Techniques to Use to Diagnose Improper Model Solution Outcomes

Type of Solution Outcome	Structural Check[a]	Degen. Resol.	Scaling[a]	Artificial Variables	Upper Bounds	Budget	Row Sum
Solver Failure	1	3	2	5	4		
Unbounded Solution	1		3		2	4	
Infeasible Solutions	1		3	2		4	5
Unsat. Optimal Solutions	1					2	2

Notes: The entry in the table gives information on the order in which to try techniques with the technique numbered 1 being the item to try first.

[a] This technique could be employed before any solving occurs. The technique also can be used when problems appear.

Solutions may be speeded up in seven ways: 1) the algebraic model structure may be changed to facilitate faster execution; 2) the gams instructions may be altered to make sure that frivolous elements are not considered; 3) an advanced basis may be entered to facilitate solver operation; 4) the problem can be scaled; 5) a starting point may be entered to facilitate nonlinear solver operation; 6) pre-solution analysis tools may be used to speed up problem execution; 7) solver options may be altered; and 8) cycling can be avoided.

Chapter 4 Looking Deeper to Find Problems

Often when GAMS fails the information one gets from the screen does not identify what is wrong. For example, the message "42 COMPILATION ERRORS" or the message "3000 EXECUTION ERRORS" identifies the general type of problem, but does not tell why or where this happens. Usually additional detail can be found in the list (LST) and possibly the LOG file. There are ways one can expand the information content of the LST file and create the LOG file. Finally there is information which appears on the screen. This chapter discusses each of these items briefly.

4.1 Messages in the LST File

The LST file contains, depending upon the termination status, several types of information about GAMS difficulties. These include:

a) Compilation messages
b) Execution messages
c) Error condition messages
d) The solver status flag
e) Solution file messages

Each of these will be discussed in order.

4.1.1 Compilation Errors

When a model is solved and there are errors due to typing, absence of semicolons, mismatched parentheses, etc. GAMS generates a set of compilation error messages. These messages identify the place at which an error occurs, a numerical code relative to that error, and a description of messages by error number. Chapter 6 discusses how to repair such errors. Error locations are demarcated by four asterisks.

4.1.2 Execution Errors

After a model has successfully been compiled, GAMS then begins execution. During execution, errors can arise either when calculations are performed or when the model is being generated (while the variables and equations are being setup for the solver). There are three main causes of execution errors. First, arithmetic difficulties can arise usually involving division by zero or improper exponentiation (i.e. raising a negative number to a fractional power). Second, the solver may fail causing an execution error (see the section on solver failures). Third, obvious infeasibilities may be encountered during model generation. These infeasibilities are usually caused by, for example, lower bounds being greater than upper bounds or an equation without any variables present equaling a positive value.

The exact type of execution error can only be found by looking at the LST file. The LST file contains a message indicating the line number in the source GAMS code on which the error occurred. For example one might find a message indicating an undefined operator in line 407. Chapter 7 gives techniques for finding such errors.

4.1.3 Error Conditions

There are a number of error conditions which can arise during execution of GAMS. One that is somewhat infrequently encountered involves GAMS running out of internal memory for code. In such a case GAMS provides a message on the screen which suggests the listing file be examined. In the listing file the message "MAXIMUM Executable Code Size of 30000 Exceeded" followed by the suggestion "TRY the GAMS Parameter CODEX = 1 or Higher." This would tell the user to rerun with the command GAMS MYFILE CODEX = 1. This error messages again marked by four asterisks in the list file.

4.1.4 Solver Status File

On occasions GAMS will fail and then include the solver status file in the LST file (or leave behind the file GAMSSOLU.SCR . One should look at this printout including the scaling information summarized at the top. The top or bottom of the status file can contain an EXIT condition. For example running MINOS with a small Super Basics limit causes a message like "EXIT.-The Super Basics Limit is Too Small".

4.1.5 Solution Printout

One may also find information about the optimality status and other solution difficulties in the solution printout. Two types of information are present, the solver status and the variable/equation status. The solver status indicates whether the model is optimal, unbounded, infeasible, an intermediate optimal, etc. The variable status identifies variables which are non-optimal (marked with NOPT) in a unbounded or intermediate solution and variables or equations which are not feasibly satisfied (marked with INFES) in an infeasible solution. A section on fixing unbounded variables and unfeasible equations appears in chapter 9.

4.2 Expanding the Material in the LST File

One may find that the LST file does not contain all that it could to diagnose a problem. In such a case the modeler should consider: a) expanding the printout using the OPTIONS LIMROW/LIMCOL, and SOLPRINT; b) adding $ONSYMXRFF ONSYMLIST to create the cross reference listing; c) using the PROFILE and PROFILETOL options to insert information on the execution time used by the GAMS statements (see discussion in chapter 10); d) adding memory dumps to receive information on the memory utilization of the arrays (see discussion in chapter 10); e) use GAMSCHK to receive structural information (see discussions in chapters 8

and 9); and f) create a LOG file as discussed next.

4.3 Messages from the Screen - Making a LOG File

During GAMS execution information is listed on the Screen. There is not a lot of information from this screen that is not in the LST file, but in some cases vital additional information is present. Several pieces of information can be gained from observing the screen. First, one can gain added appreciation for termination conditions. The screen reports number of compilation errors, optimality status, incidence of execution errors, etc. However, all of this also appears in the LST file. One can gain information about the relative execution time of individual statements. For example if a particular statement is not in a loop and during generation of execution its number stays on the screen for a long time, then one can conclude this is a statement that is using a lot of execution time. On the other hand the OPTION PROFILE command is a much more effective way of systematically gathering this information(see chapters 10 and 12). This problem with getting information off the screen is one must be staring at the screen all the time and some computers execute very quickly. Third, certain types of execution errors and presolve messages only come to the screen. Fourth in cases with unusual terminations the screen may be the only place where such information is found. One can redirect the screen output by using the JCL command LO=2 which saves the screen output to a file. The file is ordinarily named with the GAMS code name with a .LOG extension. Thus, GAMS MYFILE LO=2 will create the file MYFILE.LOG which contains the screen contents. However, this may not save all of the screen output since the last output buffer is lost(see the discussion in chapter 10). In particular system messages like segmentation faults will not be redirected.

Part II

Repairing Problems

Chapter 5 A Strategy for GAMS Modeling -- Small to Large

GAMS allows one to work with large models. PC based GAMS can be used to solve problems with tens of thousands of variables and equations. However, debugging formulations that big is not easy.

Many GAMS users are overly impressed with how easily GAMS handles large models. They often feel such a facility means they should always work on the full model. The result is a very large, sometimes extremely large, model in the very early stages of model development.

Our advice is don't take this route. First, concentrate on a small-scale implementation. Carefully check out how the data are entering the model. Insure that the model formulation is correct. Satisfy yourself that the solution contains appropriate levels of the decision variables. Implement report writing instructions and any repeated scenario based solves that need to be done. Make sure that the model is numerically robust and appropriately scaled. Generally get the model in good shape before proceeding to the large data set.

All of these things can easily done in a small model setting without having to deal with the complexities of the large data set. The larger the model the longer everything takes . This includes the solution process, the time for model compilation, the time it takes for an the editor to load in the listing. Generally, time expands exponentially. Often frustration will be the result even when one is trying to find some relatively small data problems.

The structure of GAMS does provide one with the ability to **work from small to large**. Our recommendation is that this always be exploited. Suppose we illustrate the reasons for our recommendation with examples .

5.1 Example in Transportation Context

Let us illustrate the small to large procedure in the context of a transportation model. Table 5.1 contains a GAMS input file for a transportation model. This file contains several sections. The first part (lines 1-22) defines the data identifying the supply points (called PLANT) and destinations (called MARKET) as well as the available supply at each plant, the required demand at each market and the distance from each plant to each market. The second section (lines 23-26) contains data calculations where cost is expressed as a function of distance In the third section (lines 27-49) the model is defined. The fifth section (lines 51-53) solves the model. The sixth section (lines 54-93) engages in report writing.

We can use this example to illustrate the small to large point. Suppose we set up another version of the model where we have more supply and demand locations. In that input string (Table 5.2) we expand the problem to 10 supply and 5 demand points. Let us examine what happened when the formulation was expanded. In this case:

a) The supply and demand sets were expanded to their new size;

b) the supply availability and demand requirement data were expanded and altered to cover all points; and

c) the distance table was expanded to include all entries.

The data calculation, model definition, model solution and report section writings are identical to that in lines 23-93 in Table 5.1. Also the structure and set names in the data section are identical but the data contents vary (i.e. lines 1-3, 6, 9, 14 and 19 have exact counterparts in Table 5.1).

The question now is so what? The example shows why the small to large approach is beneficial because of what we did not have to do. There were no changes in the data structure

nor in the data calculation, model structure, or report writing. Only the specific data elements were changed. One can fully work out the GAMS implementation in a smaller setting then expand to a larger setting without bearing the computational and human cost of dealing with a large data set and the associated output.

This does not mean that one will always be able to do everything perfectly within the small model. However, if one judiciously develops the small data set so it has all the features of the large data set, these authors have found that most of the work can be done in the simpler setting. Thus, if we were going to eventually use 200 supply and 100 demand locations we could preserve exactly the same code from lines 23 there on and only have to revise the data element definitions.

5.2 A More Complex Example

The primary importance of the small to large point cannot be overemphasized. Thus, we develop a second more complex example. One of these authors maintains the US Agricultural Sector model called ASM (McCarl, et al.). The small version of ASM is included on the disk under the Chapter 05 subdirectory ASM. This model has all the structural features of the actual ASM implementation. This version is run by the file R.BAT which appears below.

```
COMMAND /C GAMS ALLOFIT          PW=80                 S=.\T\SAVE1
COMMAND /C GAMS ASMMODEL.93      PW=80 R=.\T\SAVE1     S=.\T\SAVE2
COMMAND /C GAMS ASMSOLVF.NB      PW=80 R=.\T\SAVE2     S=.\T\SAVE3
COMMAND /C GAMS ASMREPT.93       PW=80 R=.\T\SAVE3
```

The file “ALLOFIT” integrates most of the ASM data and is listed below.

```
$OFFSYMLIST OFFSYMXREF
$INCLUDE “SETS.SML”
$INCLUDE “REPTSETS.93"
$INCLUDE “DEMAND.SML”
$INCLUDE “FPDATA.SML”
$INCLUDE “PROC.SML”
$INCLUDE “CROP.SML”
```

```
$INCLUDE “LIVE.SML”
$INCLUDE “MIX.SML”
$INCLUDE “NATMIX.SML”
$INCLUDE “EROSION.SML”
$INCLUDE “ASMCALSU.93"
PARAMETER CBUDDATA (ALLI, SUBREG, CROP, WTECH, CTECH, TECH) CROP BUDGET
DATA; CBUDDATA (ALLI, SUBEG, CROP, WTECH, CTECH, TECH)
                    = CCCBUDDATA(ALLI, SUBREG, CROP, WTECH, CTECH, TECH);
OPTION KILL=CCCBUDDATA;
$INCLUDE “CRP.SML”
$INCLUDE “ASMSEPER”
```

This ALLOFIT file includes a number of other files. All the files with the names that end with .SML are small versions of larger data sets. These include data sets which define the GAMS sets (SETS.SML), sectoral demand (DEMAND.SML), farm program data (FPDATA.SML), processing (PROC.SML), crop production (CROP.SML), livestock production (LIVE.SML), historical crop mixes (MIX.SML), livestock commodity distribution among states (NATMIX.SML), crop erosion data (EROSION.SML), and conservation reserve program participation (CRP.SML). The ASM structure also contains files which are independent of problem size which define the sets used in report writing (REPTSETS.93), the calculations used in setting up the model (ASMCALSU.93), parameters for the use of separable programming (ASMSEPER), the programming model structure (ASMMODEL.93), the model solution process (ASMSOLVF.NB) and the report writer (ASMREPT.93). These latter files are the same regardless of whether small or large data sets are being used. The .SML files are relatively small subsets of the more general data. We use this version in implementing model and report writing changes. When we have verified our model structure we turn to a second ALLOFIT file where all the .SML extensions are changed to .LRG and the full model is examined. For example, if one contrasts the SETS.SML file its layer counterpart with (SETS.LRG) one can see the same structure is used, although many more elements appear in the .LRG file.

This is also one other point relative to the use of small model versions. In the ASM

Sector model use there are times we wish to add various features or modify features with respect to particular set elements. For example at one point we were interested in examining and working with energy crop (biomass) production. In that particular case we altered the small model data sets so we were sure to include a region where we would implement biomass production, since in previous small model versions we didn't have such production. Thus we tailored the small model to allow us to develop and test additional model components before we fully implement them in the large model. Again this strategy allows one to fully work on the GAMS calculation, model and report writing instructions making sure that the structure is proper before turning to the larger empirical counterpart.

5.3 Making Small Parts of Large Models

The small to large strategy can also be applied in the large model context in terms of computer implementation. This involves use of restart Files. In particular consider the overall ASM structure of the model depicted in R.BAT including the ALLOFIT, model, solve and report writing files. We could have made ALLOFIT an all inclusive file rather than running the ASMMODEL.93, ASMSOLVF.NB and ASMRREPT.93 separately through restart files as in R.BAT. Such an expanded ALLOFIT File follows.

```
      $OFFSYMLIST OFFSYMXREF
      $INCLUDE "SETS.SML"
      $INCLUDE "REPTSETS.93"
      $INCLUDE "DEMAND.SML"
      $INCLUDE "FPDATA.SML"
      $INCLUDE "PROC.SML"
      $INCLUDE "CROP.SML"
      $INCLUDE "LIVE.SML"
      $INCLUDE "MIX.SML"
      $INCLUDE "NATMIX.SML"
      $INCLUDE "EROSION.SML"
      $INCLUDE "ASMCALSU.93"
      PARAMETER CBUDDATA (ALLI, SUBREG, CROP, WTECH, CTECH, TECH) CROP BUDGET
      DATA; CBUDDATA (ALLI, SUBEG, CROP, WTECH, CTECH, TECH)
                           = CCCBUDDATA(ALLI, SUBREG, CROP, WTECH, CTECH,
TECH);
```

```
OPTION KILL=CCCBUDDATA;
$INCLUDE “CRP.SML”
$INCLUDE “ASMSEPER”
$INCLUDE “ASMMODEL.93"
$INCLUDE “ASMSOLVF.NB”
$INCLUDE “ASMREPT.93"
```

where the last 3 lines are newly included. Now suppose we wished to use this implementation to fix the report writer the (ASMREPT.93 File). That means that in order to work on that file we have to redefine and execute the data setup, data calculation, model setup and model solution before our report writer runs. On the other hand if we use save and restart files as in the R.BAT file above notice that we could (on a PC) simply comment out the first three instructions in R.BAT as follows and simply work with the report writer.

```
rem command /c gams ALLOFIT.SML pw=80 s=.\t\save1
rem command /c gams ASMMODEL.93 pw=80 r=.\t\save1 s=.\t\save2
rem command /c gams ASMSOLVF.NB pw=80 r=.\t\save2     s=.\t\save3
command /c gams ASMREPT.93 pw=80 r=.\t\save3
```

Further we can go into the report writer file and by the judicious use of $ONTEXT/$OFFTEXT commands deactivate the other parts of the report writer just working with the particular calculation that we are focused on. By using this strategy one can usually revise a calculation and test its execution very quickly. This is particularly important with large models as some models take a day or two to solve and one does not wish to repeat the solution very often.

5.4 The Golden Rule Of Model Debugging

The above material puts us in a position where we can now summarize a recommended set of steps which we think should be pursued by all when debugging GAMS models. The rule guiding these steps is **WORK FROM SMALL TO LARGE**. The steps are:

a) Set up a small data set representing the full model with all structural features included (i.e. the final set names and all parameter names but with less elements in the sets)

b) Implement all data calculations, model features and report writing calculations.

c) Exhaustively check the results of Step b with the small data set making sure calculations are correct for data items, model coefficients and report writing. Also verify that the model equations are proper and that the solution makes sense.

d) Save the small model and implement a full version with the full data set. It may be desirable at this point to segment off the data parts of the input from the calculation, model definition and report writing so independent data files are maintained (see the R.BAT file in the ASM example).

e) Test the larger model implementation and verify its accuracy. Do this by isolating parts of large model to the extent possible. For example, when working on input data calculations, suppress solutions until the input data are correct. Similarly, when working on the model structure start from a restart file with all input data calculations complete.

One additional suggestion: we feel that one should keep the small model alive. As additional structural features are added to the large model one should also alter the small model. We realize that not everything can be done in the small model. However, we have encountered rarely encountered a case where the "extra" effort to work with the small model did not save 10 times the work and frustration of dealing with the large model.

5.5 Small to Large and Data

A strategy that can be exploited in the small to large model development context involves data. Often data development lags behind model development and is the most expensive part of the modeling exercise. When data are gathered for models which have not been completely

conceptualized it is often the case that either data are collected which are not needed in the model or essential data are not initially collected (when they could have been gathered cheaply) and need to be gathered at higher cost later. However, with GAMS it is often desirable that one implement a model using “invented or made up” data then later completed using the real data. Employment of such a process often reveals information about desirable data forms and characteristics as well as relationships between multiple data items before beginning the detailed work of obtaining that data. This process often improves communication with those providing data shortening the data gathering time.

One caution, modelers need to careful to assure that the data forms that they assume are consistent with the data available. Modelers must hold general discussions with the modeling client on general data availability but then can structure the model data requirements before the data absolutely become available. This will help the project avoid false starts as data needs will be completely conceptualized before they are sought.

Such a process also helps the modeler to more thoroughly think about the problem as one has to conceptualize develop the data interrelationships without being terribly influenced by the exact form and limitations of the data at hand. Thus one can come up with "proper" data requirements even though they may involve a more complex data specification process.

5.5 Simple Structural Revision

One more argument can be made for using the small to large strategy. Sometimes models need to be reformulated as the model structure or calculations need to be dramatically changed when something present or omitted is causing the model to have unrealistic solution. In such cases, modelers often find themselves playing with the model, examining alternative formulations..

This exercise is facilitated greatly by the use of a modeling system such as GAMS where alternative formulations can easily be tried out. The speed this exercise is greatly enhanced when dealing with a small data set that solves rapidly after new features are added. Don't let this advantage slip away by dealing with too large of a data set. A small data set allows one to rapidly manipulate while intimately studying formulation structure and solution details

Table 5.1 Example Transport Model

```
 1  *                          DATA DEFINITION
 2
 3  SETS  PLANT    PLANT LOCATIONS
 4               /NEWYORK  , CHICAGO , LOSANGLS /
 5        MARKET   DEMAND MARKETS
 6                  /MIAMI,   HOUSTON, MINEPLIS, PORTLAND/
 7
 8  PARAMETERS   SUPPLY(PLANT)  QUANTITY AVAILABLE AT EACH PLANT
 9                  /NEWYORK   100, CHICAGO   275, LOSANGLS   90/
10               DEMAND(MARKET)   QUANTITY REQUIRED BY DEMAND MARKET
11                  /MIAMI     100, HOUSTON    90,
12                   MINEPLIS  120, PORTLAND   90/;
13
14   TABLE DISTANCE(PLANT,MARKET) DISTANCE FROM EACH PLANT TO EACH MARKET
15
16                         MIAMI   HOUSTON   MINEPLIS   PORTLAND
17           NEWYORK        1300     1800       1100       3600
18           CHICAGO        2200     1300        700       2900
19           LOSANGLS       3700     2400       2500       1100
20
21        ;
22
23  *                          DATA CALCULATION
24
25   PARAMETER COST(PLANT,MARKET)    CALCULATED COST OF MOVING GOODS;
26             COST(PLANT,MARKET) = 50 + 1 * DISTANCE(PLANT,MARKET);
27
28  *                          MODEL DEFINITION
29
30   POSITIVE VARIABLES
31           SHIPMENTS(PLANT,MARKET) AMOUNT SHIPPED OVER A TRANSPORT ROUTE;
32   VARIABLES
33           TCOST                   TOTAL COST OF SHIPPING OVER ALL ROUTES;
34   EQUATIONS
35           TCOSTEQ                  TOTAL COST ACCOUNTING EQUATION
36           SUPPLYEQ(PLANT)          LIMIT ON SUPPLY AVAILABLE AT A PLANT
37           DEMANDEQ(MARKET)        MINIMUM REQUIREMENT AT A DEMAND MARKET;
38
39   TCOSTEQ..            TCOST =E=
40                          SUM((PLANT,MARKET), SHIPMENTS(PLANT,MARKET)*
41                                                COST(PLANT,MARKET));
42
43   SUPPLYEQ(PLANT)..   SUM(MARKET, SHIPMENTS(PLANT, MARKET))
44                                          =L= SUPPLY(PLANT);
45
46   DEMANDEQ(MARKET)..  SUM(PLANT,  SHIPMENTS(PLANT, MARKET))
47                                          =G= DEMAND(MARKET);
```

Table 5.1 (continued)

```
 48
 49   MODEL TRANSPORT /ALL/;
 50
 51  *                          MODEL SOLUTION
 52
 53   SOLVE TRANSPORT USING LP MINIMIZING TCOST;
 54
 55  *                          REPORT WRITING
 56
 57  PARAMETER MOVEMENT(*,*)  COMMODITY MOVEMENT;
 58  MOVEMENT(PLANT,MARKET)=SHIPMENTS.L(PLANT,MARKET);
 59  MOVEMENT("TOTAL",MARKET)=SUM(PLANT,SHIPMENTS.L(PLANT,MARKET));
 60  MOVEMENT(PLANT,"TOTAL")=SUM(MARKET,SHIPMENTS.L(PLANT,MARKET));
 61  MOVEMENT("TOTAL","TOTAL")=SUM(MARKET,MOVEMENT("TOTAL",MARKET));
 62
 63  OPTION DECIMALS=0;
 64  DISPLAY MOVEMENT;
 65
 66  PARAMETER COSTS(*,*)  COMMODITY MOVEMENT COSTS BY ROUTE;
 67  COSTS(PLANT,MARKET)=COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET);
 68  COSTS("TOTAL",MARKET)
 69      =SUM(PLANT,COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET));
 70  COSTS(PLANT,"TOTAL")
 71      =SUM(MARKET,COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET));
 72  COSTS("TOTAL","TOTAL")=TCOST.L;
 73  OPTION DECIMALS=0;
 74  DISPLAY COSTS;
 75
 76  PARAMETER SUPPLYREP(PLANT,*)  SUPPLY REPORT;
 77  SUPPLYREP(PLANT,"AVAILABLE")=SUPPLY(PLANT);
 78  SUPPLYREP(PLANT,"USED")=MOVEMENT(PLANT,"TOTAL");
 79  SUPPLYREP(PLANT,"MARGVALUE")=ABS(SUPPLYEQ.M(PLANT));
 80  OPTION DECIMALS=2;
 81  DISPLAY SUPPLYREP;
 82
 83  PARAMETER DEMANDREP(MARKET,*)  DEMAND REPORT;
 84  DEMANDREP(MARKET,"REQUIRED")=DEMAND(MARKET);
 85  DEMANDREP(MARKET,"RECIEVED")=MOVEMENT("TOTAL",MARKET);
 86  DEMANDREP(MARKET,"MARGCOST")=ABS(DEMANDEQ.M(MARKET));
 87  OPTION DECIMALS=2;
 88  DISPLAY DEMANDREP;
 89
 90  PARAMETER CMOVEMENT(*,*) COSTS OF CHANGING COMMODITY MOVEMENT PATTERN;
 91  CMOVEMENT(PLANT,MARKET)=SHIPMENTS.M(PLANT,MARKET);
 92  OPTION DECIMALS=2;
 93  DISPLAY CMOVEMENT;
```

Table 5.2 Example Transport Model -- Larger Version

```
1  *                          DATA DEFINITION
3  SETS  PLANT    PLANT LOCATIONS
4               /NEWYORK  , CHICAGO , LOSANGLS , BALTIMORE , WASHINGTON
5                PHILADEL , LASVEGAS,  RENO    , SEATTLE   , BOISE/
6        MARKET   DEMAND MARKETS
7                 /MIAMI,   HOUSTON, MINEPLIS, PORTLAND,BOSTON/
8
9  PARAMETERS   SUPPLY(PLANT)  QUANTITY AVAILABLE AT EACH PLANT
10                  /NEWYORK   100, CHICAGO    75, LOSANGLS   90,
11                   BALTIMORE  80, WASHINGTON 70, PHILADEL   60,
12                   LASVEGAS   40, RENO       20, SEATTLE    55,
13                   BOISE      10/
14               DEMAND(MARKET)   QUANTITY REQUIRED BY DEMAND MARKET
15                  /MIAMI     100, HOUSTON    90,
16                   MINEPLIS  120, PORTLAND   90
17                   BOSTON    180/;
18
19   TABLE DISTANCE(PLANT,MARKET) DISTANCE FROM EACH PLANT TO EACH MARKET
20
21                        MIAMI   HOUSTON   MINEPLIS   PORTLAND  BOSTON
22           NEWYORK      1300      1800       1100       3600     150
23           CHICAGO      2200      1300        700       2900     800
24           LOSANGLS     3700      2400       2500       1100    3800
25           BALTIMORE    1100      1600       1200       3700     350
26           WASHINGTON    1050      1550       1200       3700     400
27           PHILADEL     1200      1700       1150       3650     250
28           LASVEGAS     3300      2100       2300       1300    3600
29           RENO         3400      2200       2200        900    3400
30           SEATTLE       3700      2500       1900        250    3500
31           BOISE        3500      2200       1700        450    3300
33         ;
34
35  *                          DATA CALCULATION
36
37   PARAMETER COST(PLANT,MARKET)    CALCULATED COST OF MOVING GOODS;
38             COST(PLANT,MARKET) = 50 + 1 * DISTANCE(PLANT,MARKET);
39
40  *                          MODEL DEFINITION
41
42   POSITIVE VARIABLES
43           SHIPMENTS(PLANT,MARKET) AMOUNT SHIPPED OVER A TRANSPORT ROUTE;
44   VARIABLES
45           TCOST                  TOTAL COST OF SHIPPING OVER ALL ROUTES;
46   EQUATIONS
47           TCOSTEQ                 TOTAL COST ACCOUNTING EQUATION
48           SUPPLYEQ(PLANT)         LIMIT ON SUPPLY AVAILABLE AT A PLANT
49           DEMANDEQ(MARKET)       MINIMUM REQUIREMENT AT A DEMAND MARKET;
50
51   TCOSTEQ..           TCOST =E=
52                         SUM((PLANT,MARKET), SHIPMENTS(PLANT,MARKET)*
53                                                COST(PLANT,MARKET));
```

Table 5.2 (continued)

```
54
55   SUPPLYEQ(PLANT)..   SUM(MARKET, SHIPMENTS(PLANT, MARKET))
56                                          =L= SUPPLY(PLANT);
57
```

```
 58   DEMANDEQ(MARKET)..  SUM(PLANT,  SHIPMENTS(PLANT, MARKET))
 59                                        =G= DEMAND(MARKET);
 61   MODEL TRANSPORT /ALL/;
 63  *                          MODEL SOLUTION
 64
 65   SOLVE TRANSPORT USING LP MINIMIZING TCOST;
 66
 67  *                          REPORT WRITING
 68
 69  PARAMETER MOVEMENT(*,*)  COMMODITY MOVEMENT;
 70  MOVEMENT(PLANT,MARKET)=SHIPMENTS.L(PLANT,MARKET);
 71  MOVEMENT("TOTAL",MARKET)=SUM(PLANT,SHIPMENTS.L(PLANT,MARKET));
 72  MOVEMENT(PLANT,"TOTAL")=SUM(MARKET,SHIPMENTS.L(PLANT,MARKET));
 73  MOVEMENT("TOTAL","TOTAL")=SUM(MARKET,MOVEMENT("TOTAL",MARKET));
 74
 75  OPTION DECIMALS=0;
 76  DISPLAY MOVEMENT;
 77
 78  PARAMETER COSTS(*,*)  COMMODITY MOVEMENT COSTS BY ROUTE;
 79  COSTS(PLANT,MARKET)=COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET);
 80  COSTS("TOTAL",MARKET)
 81      =SUM(PLANT,COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET));
 82  COSTS(PLANT,"TOTAL")
 83      =SUM(MARKET,COST(PLANT,MARKET)*SHIPMENTS.L(PLANT,MARKET));
 84  COSTS("TOTAL","TOTAL")=TCOST.L;
 85  OPTION DECIMALS=0;
 86  DISPLAY COSTS;
 87
 88  PARAMETER SUPPLYREP(PLANT,*)  SUPPLY REPORT;
 89  SUPPLYREP(PLANT,"AVAILABLE")=SUPPLY(PLANT);
 90  SUPPLYREP(PLANT,"USED")=MOVEMENT(PLANT,"TOTAL");
 91  SUPPLYREP(PLANT,"MARGVALUE")=ABS(SUPPLYEQ.M(PLANT));
 92  OPTION DECIMALS=2;
 93  DISPLAY SUPPLYREP;
 94
 95  PARAMETER DEMANDREP(MARKET,*)  DEMAND REPORT;
 96  DEMANDREP(MARKET,"REQUIRED")=DEMAND(MARKET);
 97  DEMANDREP(MARKET,"RECIEVED")=MOVEMENT("TOTAL",MARKET);
 98  DEMANDREP(MARKET,"MARGCOST")=ABS(DEMANDEQ.M(MARKET));
 99  OPTION DECIMALS=2;
100   DISPLAY DEMANDREP;
101
102   PARAMETER CMOVEMENT(*,*) COSTS OF CHANGING COMMODITY MOVEMENT PATTERN;
103   CMOVEMENT(PLANT,MARKET)=SHIPMENTS.M(PLANT,MARKET);
104   OPTION DECIMALS=2;
105   DISPLAY CMOVEMENT;
```

Chapter 6 Fixing Compilation Errors

The execution of a GAMS program passes through a number of stages, the first of which is the compilation step. Users watching the execution of a program are sometimes dismayed to get the message: **COMPILATION ERRORS** with the message indicating 42 errors. These notes cover the process of finding and fixing GAMS compilation errors.

One note before launching into a discussion of compilation error repair. GAMS frequently marks compilation problems in latter parts of the code that are not really errors but rather the messages are caused by errors in the earlier code. We have seen cases where an omitted or extra semicolon or parenthesis in otherwise perfectly coded GAMS programs have caused hundreds of error messages. One should start fixing errors from the top and after fixing several errors, rerun the compilation to find out if those repairs took to care of later marked errors. It is hardly ever desirable to try to fix all errors pointed out in one pass.

6.1 Finding Errors

When the compilation errors message is found users should edit the .LST file and look for the cause of the errors. Errors are marked by lines which begin with 4 asterisks (****). For example one may find lines in the .LST file like the following

```
6  SET         PERIODS        TIME PERIODS          /T1*T5/;
7              ELAPSED        ELAPSED TIME           /1*12/
****             $140       $36
8              PRODUCTS       LIST OF PRODUCTS      /WHEAT,STRAWBERR
****           $108
```

which indicates errors were found in the 7th and 8th lines of the input file.

The **** GAMS compilation error line contains information about the nature of the error. Error messages are numbered and placed below the place in the line they were encountered and begin with a $. In the example above error number 140 occurred in line 7 and was caused by GAMS finding the word ELAPSED when it was looking for an instruction. In addition a number

36 error was caused by the second incidence of the word ELAPSED and a number 108 error occurred in line 8 caused by the word OF. GAMS also includes a list of the error message numbers encountered and a brief description of the error at the bottom of the .LST file. In the case above we find the following at the bottom of the file:

```
Error Messages
      36    '=' or '..' operator expected - rest of statement ignored
     108   Identifier too long
     140   Unknown symbol .
```

One may cause message explanations to be included at the place of their occurrence by using the JCL option ERRMSG=1 on DOS machines or ERRMSG 1 on UNIX machines in the GAMS call as follows

GAMS MYMODEL ERRMSG=1

or by placing the command ERRMSG 1 in the gamsprm95.txt for windows95 machines, which causes the default to be error message level one. That file has other names on other machine types being called gamsprm95.txt on NT machines or gamsprmun.txt on UNIX machines.

6.2 The Elusive Properly Placed Semi Colon

What is wrong in the example above? The cause is probably the most common GAMS error for new users -- the placement of semi colons. GAMS commands should be terminated with a semi colon (;). However, commands can occupy more than one line. In the above case the original input looked like the following.

```
SET   PERIODS     TIME PERIODS                  /T1*T5/;
      ELAPSED     ELAPSED TIME                  /1*12/
      PRODUCTS    LIST OF PRODUCTS              /WHEAT,STRAWBERRY/
```

This SET command is meant to continue for several lines, but the semi colon at the end of the first line terminates it. In turn, GAMS is looking for a command phrase in the second line and

does not recognize the word ELAPSED, so it says **UNKNOWN SYMBOL**. There are two ways of fixing this. One may move the semi colon to the actual end of the SET declaration (i.e., the end of the third line) or one may enter the word SET on the second line and place a semicolon at the end of the third line.

GAMS does not strictly require a semicolon at the end of each command. In particular when the next line begins with one of the recognized GAMS keywords (SET, PARAMETER, EQUATIONS etc.) then a semicolon is assumed. However, it is good practice to terminate all commands with a semicolon. Certainly the lines before all calculations and equation specifications (.. lines) must have a semicolon.

The example also points out another common occurrence. GAMS usually generates multiple error messages as the consequence of a mistake. Once an error is encountered numerous messages may appear as the compiler disqualifies all further usages of the item in question and/or becomes confused. In the case above, subsequent references to the ELAPSED or PRODUCTS sets would cause errors and the SOLVE statement would be disqualified. Thus, users should fix the errors starting from the beginning and skip later errors if in doubt of their validity.

6.3 Common Errors

Many types of errors are possible in GAMS. We cannot cover each one. Thus, we list a set of common errors as well as an indication of what types of GAMS error messages they cause.

6.3.1 Excess or insufficient semi colons - Error A

Too few or too many ;'s have been specified. This is discussed above. Normally this error is associated with GAMS error message $140.

6.3.2 Spelling mistakes - Error B

Named sets, parameters, equations etc. may be referenced with an different spelling than in

their declaration (i.e., the set CROPS is later referred to as CROP). GAMS identifies set name misspellings with message $120, set element misspellings with $170 and other misspellings with $140.

6.3.3 Omitted SET elements - Error C

One can forget to include elements in set declarations. In turn when these elements are referenced an error arises (i.e., an error would occur if CORN was omitted from the declaration of set CROP but used when data were defined under the PARAMETER PRICE(CROP)). GAMS identifies such errors with message $170.

6.3.4 Indexing problems - Error D

Parameters, variables, and equations are specified with a particular index order. Errors can be made where one inadvertently alters that order in subsequent references (i.e., X(A,B,C) is referred to as X(B,A,C). One can also use too many [(X(A,B,C,D)] or too few [(X(A,B)] indices. Cases where the order of sets are changed are marked with messages $170 and 171. Cases where more or less indices are used are marked with messages $148 and 149.

6.3.5 Summing over sets already indexed - Error E

Errors occur when one treats the same SET more than once [(i.e., I is summed over twice in the expression SUM(I,Y(I) +SUM(I,Z(I)))] or where an equation is defined over a set and one tries to sum over it [i.e., in the following two cases I and J define the equation and are summed over X(I,J)=SUM((I,J),Y(I,J)); or EQN(I,J)..SUM((I,J),X(I,J))=10;] Such errors are marked with message $125.

6.3.6 Neglecting to deal with sets - Error F

Errors occur when one does not sum or index over a SET referenced within an equation (i.e., in the following cases K is used occurs but are not declared Z=SUM(I,Y(I,K)); or

EQN(I,J).. X(I,J,K)=10;] This is marked with message $149.

6.3.7 Mismatched parentheses - Error G

Parentheses must match up in expressions. An excess number of open "(" parentheses are marked with $8 while excess closed ")" parentheses are marked with $12 and $36 [i.e., cases like SUM (I,X(I); or SUM (I,X(I)));] generate errors.

Two error prevention strategies are possible when dealing with parentheses. First, many editors, including in the one in the GAMS IDE program, contain a feature that allows one to ask the program to identify the matching parentheses with respect to the parenthesis that is sitting underneath the cursor. It is highly recommended that GAMS users employ this feature during model coding to make sure that parentheses are properly located for the end of sums, if statements, loops etc. Second, alternative characters can be used in place of parentheses. In particular, the symbols { } or [ ] can be used instead of the conventional ( ). GAMS is programmed to differentially recognize these symbols and generate compile errors if they do not match up. Thus a statement such as

```
x = sum( j, ABS ( TTS (j) ) );
```

can be restated as

```
x = sum[ j, ABS { TTS(j) } ];
```

Such a restatement would provide a visual basis for examining if the parentheses were properly matched in the program. It would also generate errors if one did not use the alternative parenthesis forms in the proper sequence. For example the following statement would stimulate compiler errors:

```
x = sum[ j, ABS { TTS(j} ] );.
```

6.3.8 Improper equation ".." statements - Error H

Each declared equation must be specified with a statement which contains certain elements. Omitting the ".." causes error $36. Omitting the equation type ("=L=", "=E=", or "=G=") causes error $37. Omitting the specification of a declared equation is marked with messages $71 and $256.

6.3.9 Entering improper nonlinear expressions - Error I

One get messages $51-$60 and $256 containing the word ENDOGENOUS when the equations contain nonlinear terms beyond the capability of the solver being used (i.e., nonlinear terms do not work in LP solvers)

6.3.10 Using undefined data - Error J

When data items are used which have not been declared (in a TABLE, PARAMETER or SCALE statement) one gets error $140. In addition, when declared items are used which have not received numerical values, one gets either: 1) message $141 when the items are used in calculations, or 2) messages $66 and $256 when the items are used in model equations. One can also get message $141 when referring to optimal levels of variables (i.e., X.L or X.M) when a SOLVE has not been executed.

6.3.11 Improper references to individual set elements - Error K

Individual set elements are referenced by entering their name surrounded by quotes. When the quotes are not entered one gets message $120 (i.e., if we have defined X(CROP) with CORN as an element in CROP, then X(CORN) is wrong, but X("CORN") is right).

6.3.12 Omitting a variable, parameter, or equation definition - Error L

When a variable, parameter, or equation is used which has not been declared one gets error $140

6.3.13 Duplicate names - Error M

Multiple declarations of items with the same name will cause message $150.

6.4 Common Error Cross Reference

Now are present a cross reference table between error messages in the discussion above and the above common error causes. In using this table, readers should also look at the GAMS error message text as it is may indicate additional causes.

GAMS Error Message	Potential Causes in the Above Error Discussion	Common Cause of Error
8	G	Mismatched parentheses-too many "("found
12	G	Mismatched parentheses-too many ")"found
36	G,H	Missing elements in equation definition
37	H	Missing equation type ("=L=" or "=E=" or "=G=" in equation specification
51-60	I	Illegal nonlinear specification
66	J	Item which has not been given numerical data appears in equation
71	H	Equation has been declared, but not algebraically specified with ".." statement
120	B, K	Cannot find a set with this name -- often a set element is referenced without properly being enclosed in "
125	E	Set is already in use in a sum or an equation definition
140	A, B, L	GAMS looking for a keyword or declared element and cannot find it. Check spelling and declarations. Also look for missing ;
141	J	Parameter without data used, or SOLVE does not proceed .L, and .M references
148	D	Item referenced with more or less indexed sets than in declaration
149	D, F	The set identified is not indexed either in a sum or an equation definition
150	M	Name used here duplicates that of an already defined item
170	B, C, D	Set element referred to cannot be found in set defined for this index position
171	D	Wrong set being referenced for this index position
256	H, I, J	Something wrong with model specification. Look for other error messages immediately after solve statement
257	A-M	Happens in conjunction with any GAMS error

[1]The entry below indicates when one gets error 8 the case is commonly that discussed under common error G above.

Chapter 7 Fixing Execution Errors

During GAMS usage one can encounter execution errors. Generally these occur either during calculation, during model generation or during model solution. Here we cover all three.

7.1 Execution Errors During Model Generation

Execution errors during model generation can either be calculation errors or model structure errors. Calculation errors are numerically based and can be caused by such diverse occurrences as improper exponentiation (such as raising a negative number to a real power), taking logs of negative numbers and dividing by zero. Model structure errors may occur if the equations are set up improperly or if the wrong solver is being used.

Discovery of execution errors is sometimes very straight forward and can at other times be fairly involved. In particular, when an error occurs in the middle of a multi-dimensional equation block and/or in a multi-dimensional equation term within a block, one can have difficulties finding the cause. The most practical way of finding such errors is to use the LIMROW/LIMCOL option commands.

Let us illustrate these procedures with an example. Suppose we have a GAMS input stream (Table 7.1) where we have a term with 25 elements in the objective function and 25 constraints. In this example we have execution errors: a) in the objective function term for the “S20" element where we are exponentiating a negative constant to a power (because of the assignment in line 9); b) in the XLIM constraint associated with element “S21" where we are dividing by zero (because of the assignment in line 11); and c) in the XLIM “S22" constraint where we set zero equal to one which results in an infeasible constraint (because of the assignment in line 15). When we run this small GAMS model, one sees the execution messages as follows:

```
---Starting compilation
```

```
---EXECUTMD (31)
---Starting execution
---EXECUTMD (29)
---Generating model EXECUTERR
---EXECUTMD (31)  3 Errors
***Execution error(s)
```

Notice this output says three execution errors were encountered. The LST file contains the further information as below.

```
***EXECUTION ERROR 10 AT LINE 23..ILLEGAL ARGUMENTS IN ** OPERATION
***EXECUTION ERROR   0 AT LINE 24..DIVISION BY ZERO
***EXECUTION ERROR 28 AT LINE 24..EQUATION INFEASIBLE DUE TO RHS
***INFEASIBLE EQUATIONS...
----XLIM        =E=     constraints with bad division
        XLIM (S22)..   0 =E= 1 ; (LHS = 0 ***)
```

The first line shows an exponentiation error occurs somewhere in line 23 which defines the objective function equation. The second two messages show numerical and equation structure problems in line 24.

We may easily solve the problem with respect to the infeasible equations since, in the above LST file messages, GAMS tells exactly where the problem occurs and one can go review the data for that specific term and eliminate the problem by removing the code in the GAMS input file line 15. On the other hand, the problem involving the objective function and the zero division in line 24 are not as easily found. Neither error occurs for all cases, i.e, we are not dividing by zero everywhere, so we need to know the specific elements in which the error occurs. We find this by setting LIMROW/LIMCOL to larger values. If we run with the default LIMROW value of three, while removing the error caused by line 15, we would find an undefined term in the objective function for item S20 (See Table 7.2), but we do not find the location of the constraint flaw. Thus, we need to make LIMROW bigger. In this particular case LIMROW has to be as great as 22 so that it reveals undefined terms in the equation with the faulty division. Information abstracted from the output with LIMROW = 25 is presented in Table

7.2. Here notice in the objective function equation for S20 we have a term listed as UNDF (undefined). This is the term where the zero exponentiation fault occurs. Also, notice in the equations we have an indication of the division by zero in the XLIM constraint in that there are undefined terms for XLIM(S21). We could then proceed to examine and fix the associated data.

7.2 Finding Execution Errors in Calculations

During modeling one can also receive an execution error message for undefined terms in calculations. For example, consider the input in Table 7.3 where we cause an exponentiation fault for the S21 element and a division fault for the S22 element. In that particular case when we run this we get the execution report below.

```
--- Starting compilation
--- EXECUTCL (21)
--- Starting execution
--- EXECUTCL (20)          2 Errors
** Execution error (s)
--- Erasing scratch files
```

The .LST file contains the following error message

**** EXECUTION ERROR 10 AT LINE 18 .. ILLEGAL ARGUMENTS IN ** OPERATION.

Such error may be readily fixed or one may need to display the calculated data to find wherein the array this is occurring (Table 7.4). The resultant display shows the undefined elements are associated with S20 and S21. We could then find our problem.

There is a quite confusing error which can occur during GAMS execution. That involves the square of a slight negative number. In particular, consider the example in the file solvesq. Here the program executes perfectly for the first solve in the loop, but generates an execution error during the second solve generating the message

**** EXECUTION ERROR 10 AT LINE 70 .. ILLEGAL ARGUMENTS IN ** OPERATION

where line 70 is as follows

```
70      VAR = SUM(EVENTS,(RETURN.l(EVENTS)-MEAN.l)**2)/CARD(EVENTS);
```

The problem is the return.l variable for some events is very close to the optimal level of mean.l. In fact, mean.l is ever so slightly larger. Thus GAMS has to square a very small negative number. Apparently in such a case the arithmetic processor called by GAMS cannot square the small negative and signifies an error. This unfortunately is a quite common occurrence in mean variance models. The only strategy the author has found is to rewrite line 70 in the following form:

```
70      VAR = SUM(EVENTS,(RETURN.l(EVENTS)-MEAN.l)*(RETURN.l(EVENTS)-MEAN.l))/CARD(EVENTS);
```

where the net effect is that we have eliminated the **2 term by completing the square.

7.3 Summary Procedure for Finding Execution Errors

One should follow the following steps

a) Observe the execution error message in the screen output.

b) Examine the list file for further information.

c) Look for messages starting with "***". Take note of the line numbers in the code where the errors occur.

d) Look up the statements in those line numbers. Now if the statement that is identified as an calculation statement within the program as opposed to a constraint ".." expression, then first examine whether the conditional i.e. $ should be imposed to avoid allowing zeros in the divisors, negative numbers into exponentiations or logs, etc. One does not feel that this should uniformly be the case and one wonders where the zero or the negative is coming from then immediately after the faulty statement insert a display statement for the item being calculated. In Table 7.3 in line 18 we find some difficulties in the display statement, and line 20 will display the results. In turn then, examine the

result and output and look for incidences of the word UNDF which indicates what elements the undefined terms are in and then look at the numerical items that go into those terms. This may imply one has to display the input items to the equation, i.e. in Table 7.3 line 18 we might want to display data 1 and DATADIV and examine the items for S20 and S21 which are identified in Table 7.4 as the two undefined items.

e) If on the other hand, the division fault is in an equation, then again examine for where the $ notation should be imposed to prevent zero divisions and negative exponentiations or use some combination of display statements to look at the included data in the equation and bigger use of LIMROW to know what is going into the equation. Third, if equations are identified as infeasible then examine those cases and again are resolved infeasibility or look at the data coming into those equations and inquire as to whether $ are needed to prevent those equations form being generated.

f) Watch out for the GAMS square/exponentiation error. In particular, when the formulation involves a squared term or exponentiation to a whole power, consider writing out the term in multiplication format or somehow rearrange terms so that the possibility of exponentially aiding a negative number is eliminated. This may also involve the provision of non-zero starting points as for example when lower and upper bounds are not provided, then GAMS uses starting point of zero for the variables. One may also need to provide bounds for the solution process to keep the variable away from 0 by lower bounding any variables which give problems if they equal zero to be above a number greater than the feasibility tolerance. This would insure that GAMS for example does not divide by a zero to in setting up the starting version of the problem.

7.3 Execution Errors During Model Solution

Execution errors during model solution are generally calculation errors or problems caused by a presolve. Calculation errors are numerically based and can be caused by such diverse occurrences as improper exponentiation (such as raising a negative value of a variable to a real power), taking logs of negative variables, squaring a negative term and dividing by zero. Presolve errors may indicate infeasibility or unboundedness and may also be caused by an overzealous presolve which eliminates the problem for all practical purposes. Both presolve and calculation induced errors will be discussed below.

7.3.1 Calculation Based Execution Errors During Model Solution

Execution errors can occur during problem solution. In particular, if the GAMS problem is nonlinear and some variables may be zero (or negative) which are exponentiated then the solver can stop on an execution error. Two examples of this are provided in the example problems.

In the problem solvelog.gms the objective function is optimized at a variable approaching zero where that value is logged. Running this problem through MINOS5 yields the error message

EXIT -- Termination requested by User in subroutine FUNOBJ

The use of CONOPT leads to message:

```
**    Domain error(s) in nonlinear functions.
      Check bounds on variables.
```

Both packages also cause GAMS to include the following message in the listing file:

```
****  ERRORS(S) IN EQUATION R1

      1 INSTANCE OF - UNDEFINED LOG OPERATION (RETURNED -0.1E+05)
```

Similarly the solution of the problem solvediv.gms causes equivalent error messages coming to the screen from the solvers and in the listing file the message:

```
****  ERRORS(S) IN EQUATION R1
      1 INSTANCE OF - DIVISION BY ZERO (RESULT SET TO  0.1E+05)
```

Other numerical cases causing solver failure would generate similar messages either involving data domain problems, or improper calculations in functions.

7.3.1.1 Solver calculation error cause discovery and repair strategies

Suppose we find that our model causes the solver to fail for numerical problems. The resultant GAMS identification of problematic calculations in equation terms may not tell the user where the problem is when the equation is complex. Thus one may need to investigate within the structure of the equations to discover exactly which variable in which term is causing problems. We recommend following an approach like used under the calculation and generation execution error discovery sections earlier in this manuscript. Namely you could resolve with LIMROW large and look for UNDF terms. You could also set up the calculations of the terms in an equation in a report writing model using the variable level values and display them in turn looking for terms that were marked with UNDF.

Once the cause has been found there are two repair strategies: equation reformatting and variable bounding. Equation reformatting involves investigating whether the equation is properly specified and if not manipulating it into proper form. For example a division sign may have been typed instead of a minus bound the variable away from zero by placing for example a lower bound of 0.001 or some other small positive number which is greater than the feasibility tolerance. One may also be able to manipulate the equation to avoid problems i.e. logging a variable + 0.001 or dividing by the same if it does not do damage to the practical results of the formulation.

Provision of a starting point is also another powerful problem avoidance technique. In the absence of a starting point, GAMS chooses one somewhere between the lower and upper bounds. But when bounds are not specified GAMS generally uses a starting value of zero for non negative variables. Provision of non-zero starting point is done by using command of the form:

x.l=k;

where k is the staring point. Starting points may cause the solver to start some distance from the problematic variable values and avoid execution errors

7.3.2 Presolve Based Execution Errors During Model Solution

Yet another place where solvers can run in a errors is during presolves. There are three circumstances where we have seen errors arise.

First, the presolve can, in relatively simple problems, essentially eliminate the problem and terminate with an error condition. This generally occurs because presolves commonly substitute away bounds and equality constraints to simplify the problem and may in effect simplify the problem out of existence. Second, a pre-solve may detect the problem is unbounded or infeasible and terminate. Third, in mixed in programming problems, the presolve may discover that there is no feasible integer solution and terminate. In all cases the solution returned to GAMS is generally unusual.

Consider an example. Suppose we solve the problem (presol1.gms) below

```
 1  variables z;
 2  positive variables y1,y2;
 3  equations r1,r2,r3,r4;
 4    r1..  z=e=y1+y2;
 5    r2..  y1=l=10;
 6    r3..  y2=l=10;
 7    r4..  y1+y2=e=10;
 8  model badpresol /all/
 9  option lp=osl;
10  solve badpresol using lp maximizing z;
```

The solution listing reports

```
**** SOLVER STATUS     ERROR SOLVER FAILURE
**** MODEL STATUS      6 INTERMEDIATE INFEASIBLE
```

and later we see

```
**** PRESOLVE has deleted all rows
```

In this case, what has happened is the OSL presolve has eliminated constraints r2 and r3 and

made them into simple upper bounds. Also constraint r4 has been manipulated to express y1 in terms of y2 and y1 has been substituted out of the problem. In turn the resultant model is a linear program with one variable and no explicit constraints but with a upper bound. When that resultant model is passed on to the OSL linear programming solver it cannot function (as LP problems must have constraints) so it terminates. In turn it tells GAMS the model is infeasible although it also causes the LST file to contain to message was the presolve eliminated all constraints. One would have to suppress the presolve to adequately get OSL to solve the model. A similar trial was tried using CPLEX. Under those circumstances the solver could function and return an optimal solution.

Yet another trial was tried out to see what would happen with mixed integer programming problems which did not have a feasible integer solution. The example model in this case follows

```
 1  variables z;
 2  integer variables y1,y2;
 3  equations r1,r2,r3,r4;
 4    r1..  z=e=y1+y2;
 5    r2..  y1=g=0.10;
 6    r3..  y2=g=0.10;
 7    r4..  y1+y2=l=1;
 8  model badpresol /all/
 9  option mip=cplex;
10  solve badpresol using mip maximizing z;
```

Solving this problem with CPLEX yielded the solution messages

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      10 INTEGER INFEASIBLE
```

followed by

```
Presolve found the problem infeasible or unbounded.
Turning all presolve options off before rerun.
Problem is integer infeasible.
```

The OSL solver presolve did not discover problems.

In these cases, one needs to investigate the listing to find the messages from the presolve processor to see what happened. One may need to suppress the presolve to use other techniques (GAMSCHK etc) to discover the difficulty. More material on presolves and solver terminations appears in chapter 8.

Table 7.1. GAMS Input with Model Generation Errors

```
2       sets elements /s1*s25/
3
4       parameter       data1(elements)         data to be exponentiated
5                       datadiv(elements)       divisors
6                       datamult(elements)      x limits;
7
8       data1(elements)=1;
9       data1("s20")=-1;
10
11      datadiv(elements)=1;
12      datadiv("s21")=0;
13
14      datamult(elements)=1;
15      datamult("s22")=0;
16
17      positive variables      x(elements)             variables
18      variables               obj;
19
20      equations               objr                    objective with bad exponentiation
21                              xlim(elements)          constraints with bad divisor;
22
23      objr..                  obj=e=sum(elements,data1(elements)**2*x(elements));
24      xlim(elements)..        datamult(elements)/datadiv(elements)*x(elements)=e=1;
25
26      model  executerr /all/
27
28      option limrow=0;
29      option limcol=0;
30
31      solve executerr using lp maximizing obj;
```

Table 7.2. GAMS LST File for Model With Generation [Errors]

```
**** EXECUTION ERROR  10 AT LINE 23 .. ILLEGAL ARGUMENTS IN ** OPERATION

---- OBJR        =E=  objective with bad exponentiation
OBJR..       - X(S1) - X(S2) - X(S3) - X(S4) - X(S5) - X(S6) - X(S7) - X(S8)
             - X(S9) - X(S10) - X(S11) - X(S12) - X(S13) - X(S14) - X(S15) - X(S16)
       - X(S17) - X(S18) - X(S19) + UNDF*X(S20) - X(S21) - X(S22) - X(S23)
       - X(S24) - X(S25) + OBJ =E= UNDF ; (LHS = UNDF ***)

**** EXECUTION ERROR           0 AT LINE 24 .. DIVISION BY ZERO

---- XLIM        =E=  constraints with bad divisor
XLIM(S1)..   X(S1) =E= 1 ; (LHS = 0 ***)
XLIM(S2)..   X(S2) =E= 1 ; (LHS = 0 ***)
...
XLIM(S20)..  X(S20) =E= 1 ; (LHS = 0 ***)
XLIM(S21)..  UNDF*X(S21) =E= UNDF ; (LHS = UNDF ***)
XLIM(S22)..  X(S22) =E= 1 ; (LHS = 0 ***)
```

[a]Note to replicate this output you have to modify the file in Table 7.1 by putting an asterisk in column 1 of line 15 and changing LIMROW to 25 in line 28.

Table 7.3 Example with Execution Errors in Calculations

```
 2        sets elements /s1*s25/
 3               parameter data1(elements)   data to be exponentiated
 4               datadiv(elements) divisors
 5               datamult(elements)     x limits;
 6
 7               data1(elements)=1;
 8               data1("s20")=-1;
 9
10               datadiv(elements)=1;
11               datadiv("s21")=0;
12
13               datamult(elements)=1;
14               datamult("s22")=0;
15
16               parameter result(elements);
17
18               result(elements)=data1(elements)**2.1/datadiv(elements);
19
20               display result;
21
```

Table 7.4 LST file for Model Calculation Execution Error Example.

```
**** EXECUTION ERROR            0 AT LINE 18 .. DIVISION BY ZERO

----     20 PARAMETER RESULT

S1  1.000,         S2  1.000,  S3  1.000,  S4  1.000,  S5  1.000,  S6  1.000
S7  1.000,  S8  1.000,  S9  1.000,  S10 1.000,  S11 1.000,  S12 1.000
S13 1.000,  S14 1.000,  S15 1.000,  S16 1.000,  S17 1.000,  S18 1.000
S19 1.000,  S20  UNDF,  S21  UNDF,  S22 1.000,  S23 1.000,  S24 1.000
S25 1.000
```

Chapter 8. Presolution Diagnosis and Structural Problem Repair

Modelers often face the situation where a GAMS model successfully compiles but delivers an improper answer. Models can be set up which solvers report back as infeasible or unbounded. Worse yet solutions can be optimal, but unrealistic. Typically such failures are caused by omissions, implementation errors in equation structure and/or faulty calculations. Here we discuss checking model structure before solving. The next chapter covers post solution checking.

Presolution checking can be done manually or by computer. There are three types of computerized presolution analysis techniques that may be used with GAMS. Those are the ones included inside GAMS, those inside the GAMS solvers, and those that can be used using GAMSCHK (McCarl). One may also use a GAMS compatible version of ANALYZE (Greenberg) but this will not be described herein.

In presenting this material, note that we are not trying to teach modeling. Generally, models fail either because the modeler has not properly implemented the conceptional model or because the conceptual model is fundamentally flawed. In this book we are addressing the first, rather than the second cause. Readers wishing treatment of how to set up conceptual models should review modeling books such as Williams; McCarl and Spreen; Bradley, Hax, and Magnanti; or Wagner; Hillier and Lieberman.

8.1 Adopting an Example

Our discussion will be facilitated by the use of a common example model. This model depicts a multi-plant, furniture producing, firm which makes tables and chairs in turn selling dining sets. There are two qualities of furniture that are made: fancy and functional. In furniture production, the firm uses small and large lathes, carvers, labor and a table top fitter. The production processes which can be used are: a) the normal process; b) a process using the maximum amount of the large lathe; and c) one

using maximum small lathe time. The model also depicts transportation where the tables or chairs produced at a plant can be shipped to other plants. The GAMS formulation is in Table 8.1, and a tableau in Figure 8.1.

The above example is well formed and is not particularly good for the discussion of finding out when things are drastically wrong. As a consequence we make several modifications to thoroughly mess the model up. In particular, referencing the model in Table 8.1, we will

a) change the objective function sign of the transportation cost term in line 96
b) remove the conditions based on the ACTIVITY parameter which tells whether a plant makes a product by commenting the $ term out in lines 94, 103 and 117
c) drop the condition on nonzero transport costs in lines 97, 110 and 113
d) drop the coefficient indicating term that inventory on hand is used when transporting goods out from a plant by zeroing the term starting in line 109
e) change the resource endowments to negative numbers by placing a minus in front of the coefficient in line 106

The consequent model is listed in Table 8.2. This model is both infeasible and unbounded and will be used to show the function of a number of the model structure checks.

8.2 Examining Model Structure Within GAMS

GAMS directly checks or allows checks of model structure through: a) the GAMS internal checks for obvious model structural flaws; b) use of the LIMROW/LIMCOL options to display structural features; or c) computation of matrix coefficients into parameters and formula based analyses of those data or displays thereof.

8.2.1 GAMS Internal Structural Checks

As illustrated in the Chapter 7 execution errors section, GAMS checks the model as it is generated for proper model structure. The internal checks are limited in scope and consist of examination of whether an upper bound for a variable is less than the lower bound or whether a empty equation is set equal to a nonzero constant. These are flagged as execution errors. Finding, interpreting

and dealing with messages indicating such errors is discussed in Chapter 7.

8.2.2 Using LIMROW and LIMCOL to Look at a Model

Users can examine empirical model structure using the GAMS LIMROW and LIMCOL options. In particular, one can set the LIMROW and LIMCOL parameters large enough so that the LST file includes the target variables and/or equations. For our example model, suppose we set LIMROW equal to two and LIMCOL equal to three. In that case, we get the output in Table 8.3. This portrays the first two equations in each equation block skipping the rest. Similarly, the first three variables are listed for each variable block. Now suppose we look for flaws. Note, in Table 8.3 the resource equations are flawed since a sum of positive coefficients times non negative variables is required to be less than or equal to a negative RHS. In addition, in the variable listing the TRNSPORT variables can be seen to be revenue producing in the objective function but do not use inventory.

Unfortunately, LIMROW or LIMCOL are crude model diagnosis tools. They generate a lot of output. But, that output is not always complete so needed items may not appear. In particular, if the equation or variable desired contains a lot of cases then either a large value of LIMROW/LIMCOL is needed or careful item ordering must be employed so that the desired items appear first.

The ordering of LIMROW/LIMCOL displays merits further discussion. GAMS orders that output in accordance with the order in which the set definition of the item is defined as well as the ordering of set members and variable/equation declarations. First, let us look at the effects of set membership ordering. Suppose we alter the order of the PLANT set in line 16 so that PLANT2 appears before PLANT1. In this case Table 8.4 gives the resultant LIMROW output where PLANT2 data appears first with PLANT1 data omitted. All displays involving the PLANT set will now have elements associated with PLANT2 appearing first. Similarly, the ordering of the equations as they appear in the LIMROW output depends on the order they are listed in the EQUATIONS declaration. For example,

suppose we modify lines 86-88 so rather than listing the OBJT row first we list PLANTPROD first. In turn in the output the PLANTPROD elements would appear first. In general, in the LIMROW/LIMCOL displays the output is ordered according to the following rules:

1) The equation and variable blocks are ordered according to their order of appearance in the model declarations. Thus, if items are desired in a particular order, then alter the variable and equation definition order (as in lines 80-88).

2) The sets indexing the equations and variables, control the order in which entries in a block are output. Suppose we have an item like the MAKE variable block in the example which has four indexing sets(X(I,J,K,L)). In that case the display will first vary the fourth subscript then the third, second, and first. Thus if one wishes to see cases of the second subscript (J) together then alter the item definition in the model so the 2nd subscript is in the fourth location(X(I,K,L,J)).

3) When a set is referenced the order in which the set elements appear is determined by the order in which they are listed in the set statements with one proviso as discussed below. Thus, if a set contains the words PLANT1, PLANT2 then all elements involving PLANT1 will be listed before those for PLANT2.

4) The exact order in which set elements are listed is not always controlled by the set definitions. The real controlling factor is the order in which unique set element names appear. In the example, if the word PLANT2 was interjected in the PRODUCT set, then regardless of the element order in the PLANT set, PLANT2 would always be listed first, since the PLANT2 set element name was seen first. The best strategy to use is to use unique set element names so that one always gets the proper order. Equivalently, one could also make sure the PLANT1 name was seen before the PLANT2 name in the program.

By reordering the variable and equation names as well as the set member ordering one can control the elements that appear first. For example, note how the reordering of the PLANT set in line 16, altered the listing order in Table 8-4 when compared to Table 8-3.

The LIMROW/LIMCOL options are rather blunt instruments when it comes to model structural examination. In a model with 10 equation and variable blocks an examination of one element from each block would require generation of at least 10 times as much information. Further, for large models where a particular equation falls in say the 500th case for an equation block it is easy to generate many

megabytes of LST file. A few quests for a model structural difficulties needed in this huge hay stack of information shows both the desirability of the chapter five SMALL TO LARGE argument and the desirability of a more pointed instrument such as DISPLAYCR in GAMSCHK.

8.2.3 Data Calculation Based Structural Checks

One may also examine structure by crafting special calculations. Suppose a user wished to examine a submatrix of model coefficients. For example, suppose we wished to examine the production/resource part of the model that we wish to do a test for cases where we potentially have production without any resource usage. We would implement this by inserting the GAMS statements in Table 8.4 after line 122. In this particular case we set up a parameter (AMATRIX) which has the dimension of the MAKE variables for the last four dimensions, and the dimensions of the RESOURCE equations for the first two. In turn, we compute AMATRIX just as the RESOURCE equations are computed and display it. An example of this is found in of Table 8.5 with the output in Table 8.6. Notice this allows us to review of the calculation of each of the numbers in this matrix and make sure they are appropriate and would allow much more complex calculations to be portrayed. We also introduce in lines 129-132 a computation of cases where we have missing A matrix elements for the model. In this particular case we see if for a particular furniture item type and method where there is some resource use and if there is none (i.e. resource use is less than or equal to zero) we set the parameter BADAMATRIX equal to 1 and in turn display. This display appears in Table 8.6 and shows that for PLANT2 we are missing data for normal production of Tables and for both plants we are missing data for the maximum small and large lathe production of tables. (However, we should expect this because we didn't define any data so this may not be a case where there is a bad setup for our problem). This is indicative of a more general approach when one wants to examine a submatrix one

one can compute the coefficients for that submatrix into an internal GAMS array and then display that array and examine what the matrix looks like and if desirable compute items based on conditions about whether the coefficients are appropriate.

8.2.3 Data Calculation Based Structural Checks

One may also examine structure by crafting special calculations. Suppose a user wished to examine a submatrix of model coefficients. For example, suppose we wished to examine the production/resource part of the model that we wish to do a test for cases where we potentially have production without any resource usage. We would implement this by inserting the GAMS statements in Table 8.2 after line 122. In this particular case we set up a parameter (AMATRIX) which has the dimension of the MAKE variables for the last four dimensions, and the dimensions of the RESOURCE equations for the first two. In turn, we compute AMATRIX just as the RESOURCE equations are computed and display it. An example of this is found in of Table 8.5 with the output in Table 8.6. Notice this allows us to review of the calculation of each of the numbers in this matrix and make sure they are appropriate and would allow much more complex calculations to be portrayed. We also introduce in lines 129-132 a computation of cases where we have missing A matrix elements for the model. In this particular case we see if for a particular furniture item type and method where there is some resource use and if there is none (i.e. resource use is less than or equal to zero) we set the parameter BADAMATRIX equal to 1 and in turn display. This display appears in Table 8.6 and shows that for PLANT2 we are missing data for normal production of Tables and for both plants we are missing data for the maximum small and large lathe production of tables. (However, we should expect this because we didn't define any data so this may not be a case where there is a bad setup for our problem). This is indicative of a more general approach when one wants to examine a submatrix one

one can compute the coefficients for that submatrix into an internal GAMS array and then display that array and examine what the matrix looks like and if desirable compute items based on conditions about whether the coefficients are appropriate.

8.3 Examining Structure With Solver Features

Some of the solvers accessible through GAMS have modules designed to improve solver performance which try to simplify problems before a solution process is applied. These modules are called presolves and are resident in at least OSL and CPLEX. These features are described in the solver manuals that are distributed when one buys solvers and in discussions such as those in Leunberger. These features are automatic and generally should be used. On the other hand they are not always desirable and may need to be suppressed.

A presolve tries to reduce solution time by making the problem simpler. This is done by automatically reformulating the problem removing variables and constraints, converting constraints on single variables into bounds and tightening bounds on integer variables along with other procedures. In the process of doing this presolves can find the problem is infeasible or unbounded. Such structural checks are free "byproducts" of the presolver which can help with diagnosing problems in a model, although they are not a consistent source of aid but something that happens on occasion which can be useful.

Potential presolve users should be aware of one aspect of their design which may cause misleading results.. In particular, most presolves do not use tolerances. Thus, if a presolve finds a constraint is infeasible by ten to the minus 30th, it will call that constraint infeasible even though the constraint may really be feasible and round off error is causing the infeasibility conclusion. Thus if a presolve has declared a model infeasible or unbounded it may be advantageous to suppress the presolve

and submit the problem to the solver anyhow.

The results of these presolve procedures is best illustrated by example. The above thoroughly messed up problem (Table 8.2) was run through OSL, CPLEX, MINOS5 and BDMLP.

8.3.1 CPLEX

When the default version of CPLEX (where PRESOLVE is active) was run, GAMS aborted with errors as reflected in the output abstracted in Table 8.7, Panel A. This output shows the CPLEX presolve found a problem (i.e., stat is "No Solution Exists") but the LST file does not give us any further meaningful information. This is not entirely unsurprising since CPLEX PRESOLVE is not designed for model diagnosis but rather for quickly finding out if a model is flawed. Nevertheless, experience indicates that the CPLEX PRESOLVE is of little use in finding infeasibilities and should be suppressed in such cases.

In turn CPLEX was run with PRESOLVE suppressed yielding the output abstracted in Table 8.7, Panel B. This shows there are infeasible rows in the RESOUREQ equations, as indicated by the INFES marks.

CPLEX also allows us to run with the IIS feature which generates an "irreducible infeasible set." In this case we get the output in Table 8.7, Panel C which in turn is followed by solver output like that in Panel B. The Panel C output identifies the interaction between some of the RESOUREQ equations and the MAKE variables as an infeasibility cause.

8.3.2 OSL

The OSL PRESOLVE was more informative. As shown in Table 8.8, Panel A, OSL PRESOLVE showed infeasibilities and listed the involved equations/variables. Running with the OSL PRESOLVE suppressed generated the information in Panel B which is essentially identical to the

CPLEX output in Table 8.7, Panel B.

8.3.3 Solvers without Presolve or with Presolve Suppressed

When one uses most solvers without presolves or suppresses presolve a common type of output arises. In such a case we get output like that given by CPLEX or OSL without the PRESOLVE as abstracted in Panel B of Tables 8.7 and 8.8. This output contains an identification of items which cannot be made feasible denoted by the flag INFES in the LST file. Note however differing items were marked INFES (some of the MAKE variables were included by the other solvers).

8.3.4 An Unbounded Example

We also ran an unbounded model through the solvers. This was done by taking the model in Table 8.1 with the modification in line 109 in Table 8.2. When we remove the requirement that we need to withdraw supply when transporting allowing a cheap supply of goods. In turn all the solvers reported an unbounded solution containing .LST file elements such as abstracted in Table 8.9. The code UNBND identifies the particular variable that was found to be unbounded. The variable marked UNBND differed across solvers. In some cases the solver identified the TRANSPORT variables and other cases the SELL variables. Both are unbounded under the modification that we made. The NOPT indicator is also relevant as it shows other variables which are not yet optimal in terms of their reduced costs and may be unbounded.

8.3.5 Solver Summary

When using GAMS there are cases where the solvers can detect a problem and identify where it has occurred. Unfortunately, the location is not always identified. Generally the solvers do a better job in terms of identifying sources of infeasibilities than they do unboundedness. CPLEX also accurately identifies a set of the infeasibilities when the presolve is suppressed and the IIS option used. We

recommend reliance only on the IIS finder for infeasibility detection. Generally, we feel modelers will be able to more easily find of problems using GAMSCHK as discussed in the next section and the post solution approaches in the next chapter. This is not surprising as the solvers are fundamentally designed to solve, not do structural checking.

8.4 Presolution Structural Checking with GAMSCHK

Presolution structural checking within GAMS and the GAMS solvers is not always effective. All structural information is contained in the LIMROW/ LIMCOL output, but may require sifting through a large amount of output. Similarly, solver information is almost always partial and often inaccurate. GAMSCHK is designed to assist in presolution structural examination of models.

GAMSCHK contains several different types of procedures for presolution analysis. In particular one may have GAMSCHK:

a) check the structure of the model for obvious structural flaws (using ANALYSIS).

b) create a schematic of the model matrix (using PICTURE)

c) generate a more aggregate variable and equation level block schematic coupled with structural checks (using BLOCKPIC).

d) list equation and variable block characteristics and perform a block level structural check (using BLOCKLIST).

e) display selected equation and/or variables (using DISPLAYCR).

f) portray aggregate characteristics of variables and equations (using MATCHIT)

These functions will be illustrated below.

8.4.1 Using Analysis

Probably 90% of the GAMS models that are initially infeasible, unbounded or yield strange

solutions are that way because of inadvertent errors in algebraic structure and/or data calculations. Simple checks can be applied to examine a model for such mistakes. Suppose we have the following model with one constraint and one variable:

$$\begin{aligned} \text{Max} \quad & cX \\ & aX \leq b \\ & X \geq 0 \end{aligned}$$

where c, a, and b are exogenous scalars. If c is negative while a and b are positive then X must be zero. More generally, if a maximization model contains a nonnegative variable with a negative objective coefficient and all positive resource usages in less than or equal to constraints then the optimum variable value will be zero. Similarly, if c is positive but a is negative the problem is unbounded and if a is positive but b negative the problem is infeasible. This is suggestive of a set of model structural checks. Table 8.10,11 give a set of checks considering additional variable sign restrictions and constraint types. Checks can also be implemented based on bounds. In particular given the constraint $\sum_i a_{ij} X_j \leq b_i$

this constraint will be infeasible if $\sum_{j \in a_{ij} \succ 0} a_{ij} Lb_j + \sum_{j \in a_{ij} \prec 0} a_{ij} Ub_j \geq b_i$ where Lb_j and Ub_j are lower and

upper bounds for $\mathbf{X}_j$ infeasible. Table 8.12 gives additional checks in this manner.

These checks have been implemented in the GAMSCHK ANALYSIS module. ANALYSIS checks an empirical GAMS model for the presence of cases in the Tables. For example, when the messed up model ir run through ANALYSIS, the output in Table 8.13 arises. This output identifies that: a) the TRANSPORT variables are unbounded; b) the RESOUREQ equations are infeasible; c) the

RESOUREQ equation for top capacity at PLANT2 causes all the variables in it to be zero.

We recommend use of the ANALYSIS procedure on newly developed models as it quickly identifies structural problems allows one to correct them.

8.4.2 Using a Picture

Historically a common techniques for model structural examination has been a “picture”. A picture portrays the overall model as a schematic where each row and column of the picture represents each individual equation and variable in the model. A picture routine has been written and implemented in GAMSCHK. One invokes it using the GCK file PICTURE possibly followed by variable and equation selections. An example picture of our basic model is in Table 8.14 while an example picture of the messed up model is in Table 8.15.

GAMSCHK pictures consist of four parts. The first part (Table 8.14, Panel A) defines the coefficient codes used in the schematic. Negative entries within the matrix are coded as numerical entries while positive entries are coded as alphabetic characters. Zeros are coded as blanks. Furthermore, the further the alphabetic character is from A the larger the positive number while larger digits are used to portray larger absolute value negative numbers.

The picture also contains codes for the variable and equation names. This is necessary since a GAMS allows up to 10 subscripts to be used on a variable, each of which can be 10 characters long , thus the item names can be as long as 121 characters. Panels C and D give a definition of the codes used for each variable and equation.

The Panel B part of picture can be used for structural examination. This output represents the complete programming model matrix. It is useful in this context to compare Table 8.14 to the tableau in Figure 8.1. Notice we have a 1 to 1 correspondence between the locations of the numbers and symbols

in the model. The dense block under MAKE variables 3-8 and RESOUREQ equations 1-4 and represents the resource usages that are associated with PLANT1 chair manufacturing. Similarly the transportation variables have plus and minus one entries in the product balances as depicted as the "C"'s and "3's" in the picture in the PLANTPROD equations under the TRANSPORT variables.

As mentioned above a coding scheme is employed for the variables and equations. Let us briefly define that scheme. When a variable, for example the MAKE variable, has more than 10 entries the name is strung out horizontally with variable numbers entered. Thus, the MAKE variable with a 8 under it in Table 8.15, Panel B is the eighth MAKE variable of 24. The dictionary shows that MAKE variable 8 makes fancy chairs at plant 1 using the normal lathe process. However, when there are less than 10 entries for a variable, as for SELL, then the variable name is listed vertically with a number below it. Thus, we have SELL variables 1-4 which again more extensively defined in the dictionary. The equation codes are essentially the same with equations with nine or less and more than ten cases listed differently. Again the equation dictionary (Panel D) identifies the full names.

Such a picture allows identification of structural problems. For example in the second picture (Table 8.15) the RESOUREQ equations clearly are shown as requiring positive coefficients to be less than or equal to a negative right-hand side. This causes infeasibility. Similarly the TRANSPORT variables are shown to have negative (revenue producing)[1] coefficients in the objective function and negatives in the $\leq$ equations which will be unbounded.

The picture routine can generate a large amount of output. For example, in the pictures used

[1] GAMS because of its objective function equation structure may reverse the objective function signs, thus one must be careful to understand the signs before interpreting the picture..

herein we get 33 variables and 60 equations per page. The picture output for a model with 500 variables and 300 equations would be 91 pages long and difficult to deal with. (Note: One can expand width and use a small font). The BLOCKPIC procedure discussed in the next section is designed for looking at such large models.

PICTURE has been designed to portray model subcomponents. In particular, to picture the subcomponent involving the RESOUREQ equations and the MAKE variables one would use the GCK file:

```
PICTURE
VARIABLES
MAKE
EQUATIONS
RES*
```

The resultant picture is listed in Table 8.16.

Pictures the model structural interrelationships between variables and equations. However, pictures can be very large but one can restrict attention to sub-components or turn to the more aggregate picture generated by BLOCKPIC. In addition, we recommend that one work from SMALL TO LARGE. In that setting the picture can be used in examining the small variants of a model which should have most features in common with the larger true models.

8.4.3 Using BLOCKPIC

BLOCKPIC is a more aggregate tool for examining model structure. This creates a picture as well as other output which deals with the variables and equations by block. Thus, much larger models can be handled. However for the output to be meaningful the coefficients in the blocks must be relatively homogeneous (i.e., it is nice for all coefficients in the intersection of an equation and a variable

block to be of the same sign).

BLOCKPIC was applied to the messed up example using a GCK file containing the keyword BLOCKPIC. In turn the output in Tables 8.17 (Panels A-F) was generated. Therein Panel A gives an overall model view giving variable block names, equation block names, equation types, and variable types. The signs of the coefficients at the intersection of each variable and equation block are identified with a "+" if all are positive, a "-" if all are negative and a "m" if they are mixed. Panel A sows all the MAKE variables have positive coefficients in the objective function (OBJT), and RESOUREQ equations, while all coefficients in the PLANTPROD equations are negative. Also, note the variable type at the bottom is "+" for variables that are greater than or equal to zero; "-" for variables that are less than or equal to zero and "u" for variables that are unrestricted in sign (free).

The utility of the Panel A information involves more complex models. Table 8.18 contains the Panel A information for the ASM example that was used in Chapter 5. This picture summarizes the overall structure of a model with several hundred variables and constraints showing broad structural characteristics. Note, the same aggregate picture occurs for the full version of ASM which has tens of thousands of variables and multiple thousands of equations. The second and third picture variants are given in Table 8.17, Panel B and C. Here counts of elements appearing in each block intersection are given in total or on average. This shows there are 24 MAKE variables, 16 (panel B) or 0.667 on average (Panel C) of which have positive coefficients in OBJT. Further in those 24 variables, there are 56 in total, or 2.333 on average, positive coefficients in the RESOUREQ equation and 24 in total, or -1.0 on average, negative coefficients in the PLANTPROD equations. Similarly, in Panel B we see for the PLANTPROD equation there are 24 negatives under the MAKE variables, 16 negatives under TRANSPORT, 8 positives under SELL and this equation is less than or equal to zero, with similar

information in Panel C . A count of the number of equations in a block is displayed. These picture variants also supports diagnosis of model problems. For example, the RESOURCEQ equation shows that positive coefficient times greater than or equal to zero variables less than or equal to a negative. This error will be detected in the block level analysis output.

Panel D presents block level scaling information. This shows the maximum and minimum values of coefficients in a block. Thus it shows for example, under the SELL variable in the OBJT the maximum absolute coefficient 850 while the minimum is 400. In the PLANTPROD equation the maximum coefficient is a 6 while the minimum is a one. This scaling information will be used in the latter discussion.

Finally, Panels E and F report the results of block level structural diagnosis tests. Here the checks in Table 8.10 and 8.11 are applied at the block level indicating when problems exist for all equations or variables in a block. In the messed up example BLOCKPIC identifies the errors with respect to the TRANSPORT variables and the RESOUREQ equations.

The BLOCKPIC level analysis is particularly useful since when an problem is identified in all equations, or variables in a block this usually points to major data or equation structure problems. This analysis can also be generated without the pictures by BLOCKLIST as identified in the next section.

8.4.4 Using BLOCKLIST

BLOCKLIST displays the characteristics of variable and column blocks and runs a structural analysis. It is invoked by using a GCK file containing the keyword BLOCKLIST. The output for our messed up example is that given in Table 8.19. Therein for each variable block the display includes the block name, the sign restriction on the variables, how many cases are defined, how many of variables contain nonlinear terms, as well as counts of the number of positive, negative and nonlinear terms. Also

the maximum and minimum absolute values of the but also get counts of the number of positive and negative right-hand sides. We also get block level structural analysis as an BLOCKPIC using the same tests as discussed under the ANALYSIS procedure.

8.4.5 Using DISPLAYCR

The GAMS LIMROW and LIMCOL options are not easy to target on the specific structural items one wishes to examine. DISPLAYCR displays user selected variables and equations. Users can name variables and/or equations for display as well as choose all variables that appear in a particular equation or all equations in which a variable appears. One can also display coefficients at the intersection of selected equations and variables. Items are selected through a GCK such as

```
DISPLAYCR
VARIABLES
TR*
EQUATIONS
RES*
```

in turn generating the output in Table 8.20. This output allows structural examination in essentially the same manner as discussed in the LIMROW/LIMCOL section above. Several other illustrations show the potential uses of DISPLAYCR.

A) Use of the GCK file

```
DISPLAYCR
INEQUATIONS
PLANTPROD (PLANT1)
```

Causes display of all variables that appear in the PLANTPROD equations for the

PLANT1 case.

B) Modifying the last line above to PLANTPROD (PLANT1, TABLES) further restricts

the variables to only those that are in the PLANTPROD equations for the PLANT1 and TABLES cases. One could also use

```
DISPLAYCR
INVARIABLES
MAKE (PLANT2, *, *, FANCY)
```

c) Use of the commands

```
DISPLAYCR
VARIABLES
MAKE
EQUATIONS
PLANTPROD
INTERSECT
```

yields a display of the coefficients at the intersection of the MAKE variables with the PLANTPROD equations. Through this one can display submatrices in the model.

8.4.6 Using MATCHIT

Users may wish to make lists of items that are present and their aggregate characteristics. MATCHIT creates such lists and can be used in two modes. First, one count the number of variables and/or equations matching a particular profile. Second, one can request lists of each item which does match and their characteristics. MATCHIT is invoked using an GCK file like

```
MATCHIT
VARIABLES
MAKE(Plant 2)
LIST VARIABLES
TRNSPORT
EQUATIONS
RESOUREQ
LISTEQUATIONS
PLANTPROD(PLANT1)
```

The items selected under a variables or equations command without the list modifier are totaled across all matching items. When the list modifier is present each item is listed. The characteristics summarized

include a) whether or not nonlinear terms are present; b) total number of coefficients; c)number of positive, negative, and nonlinear coefficients; d) maximum, and minimum, absolute values of coefficients and e) under the aggregate display number of items matched This MATCHIT information is designed to facilitate scaling and general inquiries on number of items variables in a model.

8.4.7 GAMSCHK and Nonlinear Terms

One feature worthy of mention when discussing GAMSCHK and the GAMS LIMROW, LIMCOL outputs is the handling of nonlinear terms. In particular, the nonlinear coefficients in the displays are Taylor series expansions around the current solution point. This current solution point is either the starting point the user originally provided in the GAMS program, or the point that was found during the last solve executed. Thus, if the model has not been solved, the starting point will be the base point for the Taylor series expansion. Otherwise the last solution will be used. If a starting point is not provided,, then a number based on the upper and lower bounds is used for the starting point. If bounds are not provided for some of the nonlinear variables then the starting point used is usually zero. This means that the first LIMROW LIMCOL or GAMSCHK display can be misleading as many the variables may well expanded around zero.

This has major implications for GAMSCHK users. The nonlinear terms are only point depictions of nonlinear terms. Namely, the Taylor series expansions around the current starting or solution point are not global values. Thus, procedures like ANALYSIS may be misleading in that they may conclude that there are flaws in the problem but the flaw may be due to nonlinear coefficients being expanded around a poor starting point. The flaws may not even be present because when the nonlinear terms expanded around a more relevant point, then a markedly different coefficient may arise which may even have an entirely different sign. The GAMSCHK output contains three asterisks in the

coefficient by coefficient displays to identify where nonlinear terms appear whereas some of the other features will count how many nonlinear terms are present. If one is solving a model and wishes to have current information on the value of the nonlinear terms a second solution needs to be done causing the nonlinear terms to be expanded about the current solution point. This might be done by using for example the following sequence

```
option NLP = minos5;
solve mymodel using NLP Maximizing return;
option NLP = GAMSchk;
solve mymodel using NLP Maximizing return;
```

This would result in GAMSCHK output based on nonlinear terms expanded around the solution point from the last solve. Thus, the displays would depict the values for the nonlinear terms that the solver was considering when it terminated. This is important when trying to find modeling errors in nonlinear models.

These points are probably best illustrated by example. Consider the following problem (nonlinp.gms):

```
3  variables z
4  positive variables
5     x1
6     x2
7  equations
8     r1
9     r2
10    r3;
11  r1..  z=e= 2*(x1-5)*(x1-5) + (x2-10)*(x2-10);
12  r2..  x1+x2 =g= 10;
13  r3..  (x1**2 - 3*x1+4) + x2**0.5 =l=100;
16  model nonlin /all/
17  option nlp=gamschk
18  solve nonlin using nlp minimizing z;
19  solve nonlin using nlp minimizing z;
```

In this model the variables X1 and X2 enter the first and third equations in a nonlinear fashion. GAMS

selects the starting point for these variables to be zero since no bounds are specified. In turn the aggregate block picture looks as follows

```
              |           |     R
              |     X  X  |     H
              |  Z  1  2  |     S
 -------------------------------
 R1           |  +  +  +  |  E  0
 R2           |     +  +  |  G  +
 R3           |     -     |  L  +
 -------------------------------
 Variable Typ |  u  +  +
```

This picture portrays X1 and X2 with positive coefficients in the objective function (R1). These are clearly only local values as for example having X1 equaling 5 would lead to an effective objective function coefficient of zero. Similarly, X1 is depicted with a negative coefficient in the third constraint but as X1 becomes larger then the coefficients in that constraint would become positive. These local values cause the ANALYSIS to identify errors and issue warnings. In particular, ANALYSIS concludes that the first constraint would cause all variables in it to be zero while the variables X1 and X2 are identified as unbounded and the third constraint is called redundant.

The situation is significantly changed by the second execution of a solve where the Taylor expansions are updated. Namely, the block picture becomes:

```
              |           |     R
              |     X  X  |     H
              |  Z  1  2  |     S
 -------------------------------
 R1           |  +        |  E  0
 R2           |     +  +  |  G  +
 R3           |     +  +  |  L  +
 -------------------------------
 Variable Typ |  u  +  +
```

Here the coefficients for the two variables in the objective function have become zero whereas the coefficient for X1 in the third equation has become positive, as has the coefficient for X2. Analysis now comes up with different conclusions. In particular, the only identified problem is that the first equation is said to cause all variables in it to have to be zero. Again this shows that ANALYSIS can generate misleading conclusions because it operates based on local values for the nonlinear terms. The user gets an indication of this since when ANALYSIS executes it invokes DISPLAYCR to provide information on the exact appearance of the equations and variables at which it is looking. The output from that follows:

```
----## EQU R1
 Z                                                    1.0000
 X1                                               *** 0.00000E+00
 X2                                               *** 0.00000E+00
         =E=                                          0.00000E+00
```

The message below also appears when analysis is dealing with nonlinear terms.

> **** There are nonlinear terms here, so message
> may only refer to a local phenomena

Here notice that the coefficients on X1 and X2 in R1 - the objective function equation - are marked with three asterisks indicating that these terms are nonlinear Taylor series expansions.

This reinforces the main point. The coefficients that are employed in GAMSCHK or the GAMS LIMROW LIMCOL displays are first order Taylor series expansions around the current point. This may lead GAMSCHK to misleading conclusions about the structure of model. These points may also be more accurately depicted by causing them to be re-expanded around for solution point so that one may see the exact parameters that the solver was considering at the time of termination.

8.4.8 GAMSCHK Summary

GAMSCHK facilitates two types of presolution structural checking. First, automatic discovery of block level model structural errors is done within ANALYSIS, BLOCKPIC, and BLOCKLIST. Second, model structure can be portrayed with pictures, displays, item counts etc possible depending on user choice. On caution, as discussed in the last section the presence of nonlinear terms can lead to misleading results since only local Taylor series expansions around the current solution point are considered.

8.5 Overall Summary for Structural Checking

Structural checking can be done via GAMS, GAMS solvers or GAMSCHK. One can also verify the equations algebraically and manually. We have written GAMSCHK and distribute through the web page agrinet.tamu.edu/mccarl to aid modelers in the quest for finding structural problems. Unfortunately, the discovery of structural errors is not always simple and does require a combination of problem specific knowledge, careful modeling, numerical investigations of the model, and repeated solution using the tools discussed in this and the next chapter.

Nonlinear programs provide special challenges. Unfortunately, the structural checking supported by GAMSCHK can be misleading. In particular, the coefficients for the values of nonlinear terms that all are displayed are entirely based upon local Taylor series expansions. GAMSCHK has no way of knowing about the global values of these terms. This is discussed in section 8.4.7 above.

Table 8.1 GAMS Input for Basic Example

```
 9  *     SECTION A       SET DEFINITION
11   SET  PRODUCT       PRODUCTS             /TABLES, CHAIRS/
12        TYPE          TYPES OF PRODUCT     /FUNCT ,FANCY/
13        RESOURCE      TYPES OF RESOURCES
14                         /SMLLATHE,LRGLATHE,CARVER,LABOR,TOP/
15        METHOD        PRODUCTION METHODS   /NORMAL,MAXSML,MAXLRG/
16        PLANT         DIFFERENT PLANTS     /PLANT1, PLANT2/;
17     ALIAS(PLANT,PLANTS);
18  *     SECTION B       DATA DEFINITION
20  TABLE PRODCOST(PRODUCT,METHOD,TYPE) PRODUCTION COST
22                                     FUNCT         FANCY
23           CHAIRS.NORMAL              15             25
24           CHAIRS.MAXSML              16             26
25           CHAIRS.MAXLRG              17             27
26           TABLES.NORMAL              80            100;
28  TABLE RES(RESOURCE,PRODUCT,TYPE,METHOD) RESOURCE USE
30           CHAIRS.FUNCT.NORMAL CHAIRS.FUNCT.MAXSML CHAIRS.FUNCT.MAXLRG
31  SMLLATHE          8                  13                     2
32  LRGLATHE          5                   2                    13
33  CARVER            4                   4                     4
34  LABOR            10                  11                    11
35   +       CHAIRS.FANCY.NORMAL CHAIRS.FANCY.MAXSML CHAIRS.FANCY.MAXLRG
36  SMLLATHE         12                  17                     5
37  LRGLATHE          7                   3                    15
38  CARVER           10                  10                    10
39  LABOR             8                   8                     8
40  +        TABLES.FUNCT.NORMAL TABLES.FANCY.NORMAL
41     LABOR          3                   5
42     TOP            1                   1   ;
44  TABLE TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS) TRANSPORT COST
46                   PLANT1.PLANT2   PLANT2.PLANT1
47      CHAIRS.FUNCT       5               5
48      TABLES.FUNCT      14
49      CHAIRS.FANCY       5               5
50      TABLES.FANCY      18                       ;
53   TABLE PRICE(PLANT,TYPE) PRICE OF SETS
54                             FUNCT          FANCY
55           PLANT1             400            800
56           PLANT2             425            850
58   TABLE RESORAVAIL(RESOURCE,PLANT) RESOURCES AVAILABLE
59                   PLANT1      PLANT2
60       TOP           500
61       SMLLATHE     1100         1400
62       LRGLATHE      880          900
63       CARVER        500         1200
64       LABOR        1750         1250      ;
66   TABLE PRODPERSET(PRODUCT,TYPE)  PRODUCTS PER SET
68             FANCY    FUNCT
69    CHAIRS    6        4
70    TABLES    1        1
72   TABLE ACTIVITY(PLANT,PRODUCT,METHOD) TELLS IF PLANT MAKES PRODUCT
73              TABLES.NORMAL  CHAIRS.(NORMAL,MAXSML,MAXLRG)
```

Table 8.1 GAMS Input for Basic Example(continued)

```
   PLANT1     1                    1
   PLANT2                           1

*   SECTION     C      MODEL  DEFINITION
 POSITIVE VARIABLES
      MAKE(PLANT,PRODUCT,METHOD,TYPE)     NUMBER OF ITEMS MADE
      TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS) NUMBER OF ITEMS TRANSPORTED
      SELL(PLANT,TYPE)                    NUMBER OF SETS SOLD;
 VARIABLES
      NETINCOME                           PROFIT;
 EQUATIONS
      OBJT                                OBJECTIVE FUNCTION ( PROFIT )
      RESOUREQ(PLANT,RESOURCE)            RESOURCES AVAILABLE
      PLANTPROD(PLANT,PRODUCT,TYPE)       PRODUCT BALANCE FOR A PLANT;

OBJT..      NETINCOME =E=
   SUM((PLANT,TYPE),
         PRICE(PLANT,TYPE) * SELL(PLANT,TYPE))
 - SUM((PLANT,PRODUCT,METHOD,TYPE)
        $ACTIVITY(PLANT,PRODUCT,METHOD)
       ,MAKE(PLANT,PRODUCT,METHOD,TYPE)*PRODCOST(PRODUCT,METHOD,TYPE))
 - SUM((PRODUCT,TYPE,PLANT,PLANTS)
         $TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
        ,TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
        *TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS));

  RESOUREQ(PLANT,RESOURCE)..
        SUM((PRODUCT,TYPE,METHOD)
             $ACTIVITY(PLANT,PRODUCT,METHOD)
             ,RES(RESOURCE,PRODUCT,TYPE,METHOD)
            *MAKE(PLANT,PRODUCT,METHOD,TYPE) )
            =l= RESORAVAIL(RESOURCE,PLANT) ;

  PLANTPROD(PLANT,PRODUCT,TYPE)..
          SUM(PLANTS
                $TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
                ,TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS))
          - SUM(PLANTS
                $TRANSCOST(PRODUCT,TYPE,PLANTS,PLANT)
               ,TRNSPORT(PRODUCT,TYPE,PLANTS,PLANT))
          + SELL(PLANT,TYPE)*PRODPERSET(PRODUCT,TYPE)
          =L= SUM(METHOD
                 $ACTIVITY(PLANT,PRODUCT,METHOD)
                     ,MAKE(PLANT,PRODUCT,METHOD,TYPE));
 MODEL FIRM /ALL/;
 OPTION SOLPRINT = ON ;
* SECTION D    SOLVE THE PROBLEM
 OPTION LP=GAMSCHK
 SOLVE FIRM USING LP MAXIMIZING NETINCOME;
```

Figure 8.1 Tableau of Example Model

			Manufacturing														Transport						Sales of Dining Set				RHS
			PLANT1								PLANT2						Tables		Chairs								
			Tables		Chairs						Chairs						Fun	Fan	Funct		Fancy						
			Normal		Normal		Maxsml		Mxlrg		Normal		Maxsml		Maxlrg		PLl	PL1	P1	P2	p1	p22	PLANT1		PLANT2		
			Fun	Fan	Fun	Fan	Fun	Fan	Fun	Fan	Fun	Fan	Fun	Fan	Fun	Fan	PL2	PL2	P2	p1	p2	p11	Fun	Fan	Fun	Fan	
Objective			-80	-100	-15	-25	-16	-26	-17	-27	-15	-25	-16	-26	-17	-27	-14	-18	-5	-5	-5	-5	400	800	-425	-850	Max
R	p	Smllathe			8	12	13	17	2	5																	≤ 1100
e	l	Lrglathe			5	7	2	3	13	15																	≤ 880
s	n	Carver			4	10	4	10	4	10																	≤ 500
o	t	Labor	3	5	10	8	11	8	11	8																	≤ 1750
u	1	Top fit	1	1																							≤ 500
r	p	Smllathe									8	12	13	17	2	5											≤ 1400
c	1	Lrglathe									5	7	2	3	13	15											≤ 900
e	t	Carver									4	10	4	10	4	10											≤ 1200
s	2	Labor									10	8	11	8	11	8											≤ 1250
	p	T Fu	-1														1						1				≤ 0
P	1	b Fa		-1														1						1			≤ 0
r	n	–																									
o	t	C Fu			-1		-1		-1										1	-1			4				≤ 0
d	1	h Fa				-1		-1		-1											1	-1		6			≤ 0
b																											
a	p	T Fu															-1								1		≤ 0
l	1	b Fa																-1								1	≤ 0
n	t	C Fu									-1		-1		-1				-1	1					4		≤ 0
c	2	h Fa										-1		-1		-1					-1	1				6	≤ 0

Table 8.2 GAMS Input for Throughly Messed Up Example

```
 9  *     SECTION A       SET DEFINITION
11   SET  PRODUCT       PRODUCTS             /TABLES, CHAIRS/
12        TYPE          TYPES OF PRODUCT     /FUNCT ,FANCY/
13        RESOURCE      TYPES OF RESOURCES
14                         /SMLLATHE,LRGLATHE,CARVER,LABOR,TOP/
15        METHOD        PRODUCTION METHODS   /NORMAL,MAXSML,MAXLRG/
16        PLANT         DIFFERENT PLANTS     /PLANT1, PLANT2/;
17     ALIAS(PLANT,PLANTS);
18  *     SECTION B       DATA DEFINITION
20  TABLE PRODCOST(PRODUCT,METHOD,TYPE) PRODUCTION COST
22                                      FUNCT         FANCY
23            CHAIRS.NORMAL              15            25
24            CHAIRS.MAXSML              16            26
25            CHAIRS.MAXLRG              17            27
26            TABLES.NORMAL              80           100;
28  TABLE RES(RESOURCE,PRODUCT,TYPE,METHOD) RESOURCE USE
30            CHAIRS.FUNCT.NORMAL CHAIRS.FUNCT.MAXSML CHAIRS.FUNCT.MAXLRG
31  SMLLATHE          8                  13                     2
32  LRGLATHE          5                   2                    13
33  CARVER            4                   4                     4
34  LABOR            10                  11                    11
35   +       CHAIRS.FANCY.NORMAL CHAIRS.FANCY.MAXSML CHAIRS.FANCY.MAXLRG
36  SMLLATHE         12                  17                     5
37  LRGLATHE          7                   3                    15
38  CARVER           10                  10                    10
39  LABOR             8                   8                     8
40  +       TABLES.FUNCT.NORMAL TABLES.FANCY.NORMAL
41     LABOR          3                   5
42     TOP            1                   1   ;
44  TABLE TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS) TRANSPORT COST
46                    PLANT1.PLANT2   PLANT2.PLANT1
47      CHAIRS.FUNCT        5                5
48      TABLES.FUNCT       14
49      CHAIRS.FANCY        5                5
50      TABLES.FANCY       18                        ;
53   TABLE PRICE(PLANT,TYPE) PRICE OF SETS
54                             FUNCT          FANCY
55           PLANT1             400            800
56           PLANT2             425            850
58   TABLE RESORAVAIL(RESOURCE,PLANT) RESOURCES AVAILABLE
59                    PLANT1      PLANT2
60       TOP            500
61       SMLLATHE      1100         1400
62       LRGLATHE       880          900
63       CARVER         500         1200
64       LABOR         1750         1250      ;
66   TABLE PRODPERSET(PRODUCT,TYPE)  PRODUCTS PER SET
68              FANCY    FUNCT
69    CHAIRS     6        4
70    TABLES     1        1
72   TABLE ACTIVITY(PLANT,PRODUCT,METHOD) TELLS IF PLANT MAKES PRODUCT
73              TABLES.NORMAL  CHAIRS.(NORMAL,MAXSML,MAXLRG)
74     PLANT1       1                    1
75     PLANT2                            1
76
```

Table 8.2 GAMS Input for Throughly Messed Up Example (continued)

```
 77  *   SECTION     C      MODEL  DEFINITION
 79   POSITIVE VARIABLES
 80        MAKE(PLANT,PRODUCT,METHOD,TYPE)     NUMBER OF ITEMS MADE
 81        TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS) NUMBER OF ITEMS TRANSPORTED
 82        SELL(PLANT,TYPE)                    NUMBER OF SETS SOLD;
 83   VARIABLES
 84        NETINCOME                           PROFIT;
 85   EQUATIONS
 86        OBJT                                OBJECTIVE FUNCTION ( PROFIT )
 87        RESOUREQ(PLANT,RESOURCE)            RESOURCES AVAILABLE
 88        PLANTPROD(PLANT,PRODUCT,TYPE)       PRODUCT BALANCE FOR A PLANT;
 90  OBJT..      NETINCOME =E=
 91     SUM((PLANT,TYPE),
 92           PRICE(PLANT,TYPE) * SELL(PLANT,TYPE))
 93    - SUM((PLANT,PRODUCT,METHOD,TYPE)
 94  *        $ACTIVITY(PLANT,PRODUCT,METHOD)
 95          ,MAKE(PLANT,PRODUCT,METHOD,TYPE)*PRODCOST(PRODUCT,METHOD,TYPE))
 96    + SUM((PRODUCT,TYPE,PLANT,PLANTS)
 97  *         $TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
 98           ,TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
 99           *TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS));
100
101    RESOUREQ(PLANT,RESOURCE)..
102           SUM((PRODUCT,TYPE,METHOD)
103  *              $ACTIVITY(PLANT,PRODUCT,METHOD)
104                ,RES(RESOURCE,PRODUCT,TYPE,METHOD)
105               *MAKE(PLANT,PRODUCT,METHOD,TYPE) )
106               =l= -RESORAVAIL(RESOURCE,PLANT) ;
107
108    PLANTPROD(PLANT,PRODUCT,TYPE)..
109             0*SUM(PLANTS
110  *                 $TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
111                        ,TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS))
112             - SUM(PLANTS
113  *                 $TRANSCOST(PRODUCT,TYPE,PLANTS,PLANT)
114                   ,TRNSPORT(PRODUCT,TYPE,PLANTS,PLANT))
115             + SELL(PLANT,TYPE)*PRODPERSET(PRODUCT,TYPE)
116             =L= SUM(METHOD
117  *                  $ACTIVITY(PLANT,PRODUCT,METHOD)
118                     ,MAKE(PLANT,PRODUCT,METHOD,TYPE));
120   MODEL FIRM /ALL/;
121   OPTION SOLPRINT = ON ;
123  * SECTION D    SOLVE THE PROBLEM
124   OPTION LP=GAMSCHK
125   SOLVE FIRM USING LP MAXIMIZING NETINCOME;
```

Table 8.3 LIMROW AND LIMCOL Output

```
---- OBJT        =E=  OBJECTIVE FUNCTION ( PROFIT )
OBJT..  80*MAKE(PLANT1,TABLES,NORMAL,FUNCT) + 100*MAKE(PLANT1,TABLES,NORMAL,FANCY)
      + 15*MAKE(PLANT1,CHAIRS,NORMAL,FUNCT) + 25*MAKE(PLANT1,CHAIRS,NORMAL,FANCY)
      + 16*MAKE(PLANT1,CHAIRS,MAXSML,FUNCT) + 26*MAKE(PLANT1,CHAIRS,MAXSML,FANCY)
      + 17*MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT) + 27*MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)
      + 80*MAKE(PLANT2,TABLES,NORMAL,FUNCT) + 100*MAKE(PLANT2,TABLES,NORMAL,FANCY)
      + 15*MAKE(PLANT2,CHAIRS,NORMAL,FUNCT) + 25*MAKE(PLANT2,CHAIRS,NORMAL,FANCY)
      + 16*MAKE(PLANT2,CHAIRS,MAXSML,FUNCT) + 26*MAKE(PLANT2,CHAIRS,MAXSML,FANCY)
      + 17*MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT) + 27*MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)
      - 14*TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2) - 18*TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
      - 5*TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2) - 5*TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)
      - 5*TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2) - 5*TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)
      - 400*SELL(PLANT1,FUNCT) - 800*SELL(PLANT1,FANCY) - 425*SELL(PLANT2,FUNCT)
      - 850*SELL(PLANT2,FANCY) + NETINCOME =E= 0 ; (LHS = 0)

---- RESOUREQ    =L=  RESOURCES AVAILABLE

RESOUREQ(PLANT1,SMLLATHE)..  8*MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)
      + 12*MAKE(PLANT1,CHAIRS,NORMAL,FANCY) + 13*MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)
      + 17*MAKE(PLANT1,CHAIRS,MAXSML,FANCY) + 2*MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)
      + 5*MAKE(PLANT1,CHAIRS,MAXLRG,FANCY) =L= -1100 ; (LHS = 0, INFIES = 1100 ***)

RESOUREQ(PLANT1,LRGLATHE)..  5*MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)
      + 7*MAKE(PLANT1,CHAIRS,NORMAL,FANCY) + 2*MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)
      + 3*MAKE(PLANT1,CHAIRS,MAXSML,FANCY) + 13*MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)
      + 15*MAKE(PLANT1,CHAIRS,MAXLRG,FANCY) =L= -880 ; (LHS = 0, INFIES=880 ***)

REMAINING 8 ENTRIES SKIPPED

---- PLANTPROD   =L=  PRODUCT BALANCE FOR A PLANT

PLANTPROD(PLANT1,TABLES,FUNCT)..  - MAKE(PLANT1,TABLES,NORMAL,FUNCT)
      - MAKE(PLANT1,TABLES,MAXSML,FUNCT) - MAKE(PLANT1,TABLES,MAXLRG,FUNCT)
      - TRNSPORT(TABLES,FUNCT,PLANT1,PLANT1) - TRNSPORT(TABLES,FUNCT,PLANT2,PLANT1)
      + SELL(PLANT1,FUNCT) =L= 0 ; (LHS = 0)

PLANTPROD(PLANT1,TABLES,FANCY)..  - MAKE(PLANT1,TABLES,NORMAL,FANCY)
      - MAKE(PLANT1,TABLES,MAXSML,FANCY) - MAKE(PLANT1,TABLES,MAXLRG,FANCY)
      - TRNSPORT(TABLES,FANCY,PLANT1,PLANT1) - TRNSPORT(TABLES,FANCY,PLANT2,PLANT1)
      + SELL(PLANT1,FANCY) =L= 0 ; (LHS = 0)

REMAINING 6 ENTRIES SKIPPED
---- MAKE NUMBER OF ITEMS MADE
MAKE(PLANT1,TABLES,NORMAL,FUNCT)
                (.LO, .L, .UP = 0, 0, +INF)
       80       OBJT
```

Table 8.3 LIMROW AND LIMCOL Output(continued)

```
        3       RESOUREQ(PLANT1,LABOR)
        1       RESOUREQ(PLANT1,TOP)
       -1       PLANTPROD(PLANT1,TABLES,FUNCT)

MAKE(PLANT1,TABLES,NORMAL,FANCY)
                (.LO, .L, .UP = 0, 0, +INF)
      100       OBJT
        5       RESOUREQ(PLANT1,LABOR)
        1       RESOUREQ(PLANT1,TOP)
       -1       PLANTPROD(PLANT1,TABLES,FANCY)

MAKE(PLANT1,TABLES,MAXSML,FUNCT)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       PLANTPROD(PLANT1,TABLES,FUNCT)
REMAINING 21 ENTRIES SKIPPED

---- TRNSPORT   NUMBER OF ITEMS TRANSPORTED
TRNSPORT(TABLES,FUNCT,PLANT1,PLANT1)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       PLANTPROD(PLANT1,TABLES,FUNCT)

TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
                (.LO, .L, .UP = 0, 0, +INF)
      -14       OBJT
       -1       PLANTPROD(PLANT2,TABLES,FUNCT)

TRNSPORT(TABLES,FUNCT,PLANT2,PLANT1)
                (.LO, .L, .UP = 0, 0, +INF)
       -1       PLANTPROD(PLANT1,TABLES,FUNCT)
REMAINING 13 ENTRIES SKIPPED

---- SELL       NUMBER OF SETS SOLD
SELL(PLANT1,FUNCT)
                (.LO, .L, .UP = 0, 0, +INF)
     -400       OBJT
        1       PLANTPROD(PLANT1,TABLES,FUNCT)
        4       PLANTPROD(PLANT1,CHAIRS,FUNCT)

SELL(PLANT1,FANCY)
                (.LO, .L, .UP = 0, 0, +INF)
     -800       OBJT
        1       PLANTPROD(PLANT1,TABLES,FANCY)
        6       PLANTPROD(PLANT1,CHAIRS,FANCY)

SELL(PLANT2,FUNCT)
                (.LO, .L, .UP = 0, 0, +INF)
     -425       OBJT
        1       PLANTPROD(PLANT2,TABLES,FUNCT)
        4       PLANTPROD(PLANT2,CHAIRS,FUNCT)
REMAINING  ENTRIES SKIPPED

---- NETINCOME  PROFIT

NETINCOME
                (.LO, .L, .UP = -INF, 0, +INF)
        1       OBJT
```

Table 8.4 LIMROW Output After Set Reordering

```
---- OBJT        =E=  OBJECTIVE FUNCTION ( PROFIT )

OBJT..  80*MAKE(PLANT2,TABLES,NORMAL,FUNCT) + 100*MAKE(PLANT2,TABLES,NORMAL,FANCY)
```

```
      + 15*MAKE(PLANT2,CHAIRS,NORMAL,FUNCT) + 25*MAKE(PLANT2,CHAIRS,NORMAL,FANCY)
      + 16*MAKE(PLANT2,CHAIRS,MAXSML,FUNCT) + 26*MAKE(PLANT2,CHAIRS,MAXSML,FANCY)
      + 17*MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT) + 27*MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)
      + 80*MAKE(PLANT1,TABLES,NORMAL,FUNCT) + 100*MAKE(PLANT1,TABLES,NORMAL,FANCY)
      + 15*MAKE(PLANT1,CHAIRS,NORMAL,FUNCT) + 25*MAKE(PLANT1,CHAIRS,NORMAL,FANCY)
      + 16*MAKE(PLANT1,CHAIRS,MAXSML,FUNCT) + 26*MAKE(PLANT1,CHAIRS,MAXSML,FANCY)
      + 17*MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT) + 27*MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)
      - 14*TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2) - 18*TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
      - 5*TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1) - 5*TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
      - 5*TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1) - 5*TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)
      - 425*SELL(PLANT2,FUNCT) - 850*SELL(PLANT2,FANCY) - 400*SELL(PLANT1,FUNCT)
      - 800*SELL(PLANT1,FANCY) + NETINCOME =E= 0 ; (LHS = 0)

---- RESOUREQ    =L=  RESOURCES AVAILABLE

RESOUREQ(PLANT2,SMLLATHE)..  8*MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)
      + 12*MAKE(PLANT2,CHAIRS,NORMAL,FANCY) + 13*MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)
      + 17*MAKE(PLANT2,CHAIRS,MAXSML,FANCY) + 2*MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)
      + 5*MAKE(PLANT2,CHAIRS,MAXLRG,FANCY) =L= -1400 ; (LHS = 0 ***)

RESOUREQ(PLANT2,LRGLATHE)..  5*MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)
      + 7*MAKE(PLANT2,CHAIRS,NORMAL,FANCY) + 2*MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)
      + 3*MAKE(PLANT2,CHAIRS,MAXSML,FANCY) + 13*MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)
      + 15*MAKE(PLANT2,CHAIRS,MAXLRG,FANCY) =L= -900 ; (LHS = 0 ***)

REMAINING 8 ENTRIES SKIPPED

---- PLANTPROD   =L=  PRODUCT BALANCE FOR A PLANT

PLANTPROD(PLANT2,TABLES,FUNCT)..  - MAKE(PLANT2,TABLES,NORMAL,FUNCT)
      - MAKE(PLANT2,TABLES,MAXSML,FUNCT) - MAKE(PLANT2,TABLES,MAXLRG,FUNCT)
      - TRNSPORT(TABLES,FUNCT,PLANT2,PLANT2) - TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
      + SELL(PLANT2,FUNCT) =L= 0 ; (LHS = 0)

PLANTPROD(PLANT2,TABLES,FANCY)..  - MAKE(PLANT2,TABLES,NORMAL,FANCY)
      - MAKE(PLANT2,TABLES,MAXSML,FANCY) - MAKE(PLANT2,TABLES,MAXLRG,FANCY)
      - TRNSPORT(TABLES,FANCY,PLANT2,PLANT2) - TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
      + SELL(PLANT2,FANCY) =L= 0 ; (LHS = 0)

REMAINING 6 ENTRIES SKIPPED
```

Table 8.5 Calculations to Find Matrix Errors Using Basic GAMS

```
123  parameter amatrix(PLANT,RESOURCE,plant,PRODUCT,TYPE,METHOD) amatrix for model;
124
125  amatrix(PLANT,RESOURCE,plant,PRODUCT,TYPE,METHOD)$ACTIVITY(PLANT,PRODUCT,METHOD)=
126                RES(RESOURCE,PRODUCT,TYPE,METHOD);
127  option amatrix:0:2:4;display amatrix;
128
129  parameter badamatrix(PLANT,PRODUCT,TYPE,METHOD) missing items for variables;
130
131  badamatrix(PLANT,PRODUCT,TYPE,METHOD)=1$(sum(resource,
132      amatrix(PLANT,RESOURCE,plant,PRODUCT,TYPE,METHOD) ) le 0);
133  option badamatrix:0:3:1;display badamatrix;
```

Table 8.6 Displays of Calculations to Find Matrix Errors

Panel A - Display of AMatrix

---- 127 PARAMETER AMATRIX amatrix for model

	PLANT1 TABLES FUNCT NORMAL	PLANT1 TABLES FANCY NORMAL	PLANT1 CHAIRS FUNCT NORMAL	PLANT1 CHAIRS FUNCT MAXSML	PLANT1 CHAIRS FUNCT MAXLRG	PLANT1 CHAIRS FANCY NORMAL	PLANT1 CHAIRS FANCY MAXSML	PLANT1 CHAIRS FANCY MAXLRG	PLANT2 CHAIRS FUNCT NORMAL	PLANT2 CHAIRS FUNCT MAXSML	PLANT2 CHAIRS FUNCT MAXLRG	PLANT2 CHAIRS FANCY NORMAL	PLANT2 CHAIRS FANCY MAXSML	PLANT2 CHAIRS FANCY MAXLRG
PLANT1.SMLLATHE			8	13	2	12	17	5						
PLANT1.LRGLATHE			5	2	13	7	3	15						
PLANT1.CARVER			4	4	4	10	10	10						
PLANT1.LABOR	3	5	10	11	11	8	8	8						
PLANT1.TOP	1	1												
PLANT2.SMLLATHE									8	13	2	12	17	5
PLANT2.LRGLATHE									5	2	13	7	3	15
PLANT2.CARVER									4	4	4	10	10	10
PLANT2.LABOR									10	11	11	8	8	8

---- 133 PARAMETER BADAMATRIX missing items for variables

	NORMAL	MAXSML	MAXLRG
PLANT1.TABLES.FUNCT		1	1
PLANT1.TABLES.FANCY		1	1
PLANT2.TABLES.FUNCT	1	1	1
PLANT2.TABLES.FANCY	1	1	1

Table 8.7 Abstracted CPLEX Output for Messed Up Problem

Panel A Results from Default CPLEX Version

```
**** SOLVER STATUS     ERROR SYSTEM FAILURE
**** MODEL STATUS      ERROR NO SOLUTION
Starting CPLEX...
Column 'x26' set to infinite upper bound.
Presolve Time =    0.00 sec.
No solution exists.
*** Error in reporting the solution
**** USER ERROR(S) ENCOUNTERED
```

Panel B Results from CPLEX with PreSolve Suppressed

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      4 INFEASIBLE

Problem is infeasible.

---- EQU RESOUREQ    RESOURCES AVAILABLE
                     LOWER     LEVEL     UPPER    MARGINAL
PLANT1.SMLLATHE       -INF        .    -1100.000    -0.059 INFES
PLANT1.LRGLATHE       -INF        .     -880.000    -0.067 INFES
PLANT1.CARVER         -INF        .     -500.000    -0.100 INFES
PLANT1.LABOR          -INF        .    -1750.000    -0.091 INFES
PLANT1.TOP            -INF        .     -500.000    -1.000 INFES
PLANT2.SMLLATHE       -INF        .    -1400.000    -0.059 INFES
PLANT2.LRGLATHE       -INF        .     -900.000    -0.067 INFES
PLANT2.CARVER         -INF        .    -1200.000    -0.100 INFES
PLANT2.LABOR          -INF        .    -1250.000    -0.091 INFES
PLANT2.TOP            -INF        .          .          .
```

Panel C Results from CPLEX with PreSolve Suppressed and IIS Enabled

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      4 INFEASIBLE
Problem is infeasible.
IIS found. An IIS is a set equations and variable bounds (ie a submodel) which
is infeasible but becomes feasible if any one equation or variable bound
is dropped. A problem may contain several independant IISs but only one
will be found per run.

Number of equations in the IIS:  1.
   upper: RESOUREQ(PLANT2,LABOR) < -113.636

Number of variables in the IIS:  8.
   lower: MAKE(PLANT2,TABLES,NORMAL,FUNCT) > 0
   lower: MAKE(PLANT2,TABLES,NORMAL,FANCY) > 0
   lower: MAKE(PLANT2,CHAIRS,NORMAL,FUNCT) > 0
   lower: MAKE(PLANT2,CHAIRS,NORMAL,FANCY) > 0
   lower: MAKE(PLANT2,CHAIRS,MAXSML,FUNCT) > 0
   lower: MAKE(PLANT2,CHAIRS,MAXSML,FANCY) > 0
   lower: MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT) > 0
   lower: MAKE(PLANT2,CHAIRS,MAXLRG,FANCY) > 0
```

Table 8.8 Abstracted OSL Output for Messed Up Problem

Panel A Results under default setup

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      4 INFEASIBILITY

**** PRESOLVE detected that model is infeasible

**** ERRORS(S) IN EQUATION RESOUREQ(PLANT1,SMLLATHE)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT1,LRGLATHE)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT1,CARVER)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT1,LABOR)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT1,TOP)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT2,SMLLATHE)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT2,LRGLATHE)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT2,CARVER)
**** ERRORS(S) IN EQUATION RESOUREQ(PLANT2,LABOR)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,TABLES,NORMAL,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,TABLES,NORMAL,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,NORMAL,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,MAXSML,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,TABLES,NORMAL,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,TABLES,NORMAL,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,NORMAL,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,MAXSML,FANCY)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)
**** ERRORS(S) IN VARIABLE MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)
```

Panel B Results with PreSolve Supressed

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      4 INFEASIBLE
**** OBJECTIVE VALUE                0.0000
---- EQU RESOUREQ    RESOURCES AVAILABLE
                     LOWER     LEVEL     UPPER    MARGINAL
PLANT1.SMLLATHE       -INF       .    -1100.000       .    INFES
PLANT1.LRGLATHE       -INF       .     -880.000       .    INFES
PLANT1.CARVER         -INF       .     -500.000       .    INFES
PLANT1.LABOR          -INF       .    -1750.000       .    INFES
PLANT1.TOP            -INF       .     -500.000       .    INFES
PLANT2.SMLLATHE       -INF       .    -1400.000       .    INFES
PLANT2.LRGLATHE       -INF       .     -900.000       .    INFES
PLANT2.CARVER         -INF       .    -1200.000       .    INFES
PLANT2.LABOR          -INF       .    -1250.000       .    INFES
PLANT2.TOP            -INF       .          .          .
**** REPORT SUMMARY :         0     NONOPT
                              9 INFEASIBLE (INFES)
                              0  UNBOUNDED
```

Table 8.9 Typical Output with Unbounded Model

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      3 UNBOUNDED
**** OBJECTIVE VALUE                0.0000

 Model is unbounded

**** ERRORS(S) IN VARIABLE TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
     1 INSTANCE OF - Unbounded variable
---- VAR TRNSPORT    NUMBER OF ITEMS TRANSPORTED
                              LOWER     LEVEL     UPPER    MARGINAL

TABLES.FUNCT.PLANT1.PLANT2      .         .        +INF       .
TABLES.FANCY.PLANT1.PLANT2      .         .        +INF       .
CHAIRS.FUNCT.PLANT1.PLANT2      .         .        +INF     97.750 UNBND
CHAIRS.FUNCT.PLANT2.PLANT1      .         .        +INF     -5.000  NOPT
CHAIRS.FANCY.PLANT1.PLANT2      .         .        +INF       .
CHAIRS.FANCY.PLANT2.PLANT1      .         .        +INF    111.667
```

Table 8.10 Conditions under which Analysis will Advise of Potential Difficulty for Equations

Type of Constraint	Sign of Coefficient on Nonnegative Variables		Sign of Coefficient on Nonpositive Variables		Sign of Coefficient on Unrestricted Variables		Sign of RHS	Type of PC[a]	Example[b]
	+	-	+	-	+	-			
≤	≥ 0[c]	0	0	≥ 0[c]	0	0	0	Zero Variables - Case 1	x ≤ 0 [f] -y ≤ 0 x - y ≤ 0
	≥ 0	0	0	≥0	0	0	-	Infeasible -PS Case 2	x ≤ - k -y ≤ - k x - y ≤ - k
	0	≥ 0	≥ 0	0	0	0	+ or 0[d]	Redundant - PS Case 3	-x ≤ + k y ≤ + k -x + y ≤ + k
0	≥ 0	0	0	≥ 0	0	0	0	Zero Variables - Case 1	x = 0 -y = 0 x - y = 0
	0	≥ 0	≥ 0	0	0	0	0	Zero Variables - Case 1	-x = 0 y = 0 -x + y = 0
	≥	0	0	≥	0	0	-	Infeasible -PS Case 2	x = -k -y = -k
	0	≥	≥	0	0	0	+	Infeasible -PS Case 2	-x = k y = k
	0	0	0	0	+[f]	-[f]	0	Zero Variable	x=0
≥	≥ 0	0	0	≥ 0	0	0	- or 0[d]	Redundant - PS Case 3	x ≥ - k -y ≥ - k x - y ≥ - k
	0	≥ 0	≥ 0	0	0	0	0 or +	Infeasible -PS Case 2	-x ≥ + k y ≥ + k -x + y ≥ + k
	0	≥0	≥0	0	0	0	0	Zero Variables - Case 1	-x ≥ 0 y ≥ 0 -x + y ≥ 0

[a] The PS cases indicate, because the variables in this equation follow this pattern, that:

1. The variables appearing with nonzeros in this equation are forced to equal zero.
2. This equation can never be satisfied and is obviously infeasible.
3. This equation is redundant. The nonnegativity conditions are a stronger restriction.

[b] In the examples let x = non-negative variables
y = non-positive variables

[c] In this case (and a number of the cases below) one or the other set of coefficients would have to be nonempty.

[d] In this case all zeros in the matrix would generate a warning provided the RHS was nonzero.

[e] This and the entries below give examples of the problem covered by each warning. Namely, in the first case examining only the nonnegative variables suppose all the non-negative variables have signs ≥ 0 but the right-hand-side is zero. Thus, we have X ≥ 0 and X ≤ 0 which implies X = 0. A warning is generated in that case.

[f] Only one coefficient is allowed.

Table 8.11 Conditions under which Analysis will Advise about Potential Difficulties for Variables in a Maximum Problem.

Type of Variable	Objective function coefficient sign	Sign of a_{ij}'s in ≥ rows		Number of a_{ij}'s in = rows		Number of a_{ij}'s in ≤ rows		PS[a]	Examples
		+	-	+	-	+	-		
Nonnegative	+	≥ 0	0	0	0	0	≥ 0	Unbounded Variable case 1	max x[b] x ≥ a -x ≤ b
	-	0	≥ 0	0	0	≥ 0	0	Zero optimal solution case 2	max -x -x ≥ a x ≤ b
	0	≥0	0	0	0	0	≥0	Variable Relaxes constraint case 3	max ox x ≥a -x ≤ b
	0	≥0	0	≥ 0[c]	≥ 0[c]	0	≥0	Variable Relaxes constraint case 4	max ox x ≥a -x ≤ b DZ+x = g
Nonpositive	+	≥ 0	0	0	0	0	≥ 0	Zero optimal solution case 2	max y y ≥ a -y ≤ b
	-	0	≥ 0	0	0	≥ 0	0	Unbounded Variable case 1	max -y -y ≥ a y ≤ b
	0	0	≥ 0	0	0	≥ 0	0	Variable Relaxes constraint case 3	max oy -y ≥a y ≤ b
	0	0	≥ 0	≥ 0[c]	≥ 0[c]	≥ 0	0	Variable Relaxes constraint case 4	max oy -y ≥a y ≤ b DZ+y = g
Unrestricted	+/-	0	0	0	0	0	0	Unbounded Variable case 1	max ± z

[a] PS cases are:
The variables which satisfy this condition are:
1) Obviously unbounded as they can be increased to infinity contributing to the objective function while satisfying the constraints.
2) Obviously zero since they consume constraint resources and have a cost in the objective function.
3) Warning this variable relaxes all constraints in which it appears
4) Warning this variable relaxes all the equality constraints in which it appears in one direction

[b] This example shows x has a positive term in the objective and can be increased without ever violating any constraints so x is unbounded.

[c] Only one coefficient can be present in the equality rows

[a] Types of warnings are:
The variables which satisfy this condition are:
1) Obviously unbounded as they can be increased to infinity contributing to the objective function while satisfying the constraints.
2) Obviously zero since they consume constraint resources and have a cost in the objective function.

[b] This example shows x has a positive term in the objective and can be increased without ever violating any constraints so x is unbounded.

Table 8-12 Conditions for Potential Infeasibility or Redundancy in Equations Based on Bounds of Variables

	TYPE OF CONSTRAINT		PS
	≤ b	≥ b	
SUM OF THE SMALLEST VALUE[a]	> b	---	INFEASIBLE
	---	> b	REDUNDANT
SUM OF THE LARGEST VALUE[b]	---	< b	INFEASIBILE
	< b	---	REDUNDANT

Note:

a. Suppose X_j is bounded as follows, LB_j (lower bound) ≤ X_j ≤UB_j (upper bound), and the sum is either >b or <b, then this will be the smallest value which could happen in that sum. If the constraint is ≤b, then if S>b, we know that this constraint will never be satisfied. In the constraint is ≥b, then if S>b, we know that this constraint will not limit any possible X value. Hence, it is redundant.

b. Suppose X_j is bounded as follows, LB_j (lower bound) ≤ X_j ≤UB_j (upper bound), and we have the sum which is either >b or <b, then this will be the largest value which could happen in that sum. If the constraint is ≤b, if L<b, we know that this constraint will not limit any possible X value. Hence, it is redundant. In the constraint is ≥b, then if L<b, we know that this constraint will never be satisfied.

c.Thanks to Paul Preckel for bringing these tests to the authors' attention.

Table 8.13 GAMSCHK Analysis Output for Throughly Messed Up Example

```
 ----#### Executing ANALYSIS

   ----### Analysis of Variables (treating nonlinear terms at their starting p

**** ERROR These variables are unbounded -- they
           have a desirable objective function coeffic
           all 0 or + coefficients in the =G= rows
           all 0 or - coefficients in the =L= rows
           and no coefficients in the =E= rows

 ##  TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
     TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)
     TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)

----### Analysis of Equations (treating nonlinear terms at their starting poin

 **** ERROR This =L= constr. causes an infeasible model
               since the nonnegative variables present
                        have only 0 or + coefficients
                     the nonpositive variables present
                        have only 0 or - coefficents
                     the unrestricted variables
                        have only zero coefficients
                     and the RHS is negative

 ##  RESOUREQ(PLANT1,SMLLATHE)
     RESOUREQ(PLANT1,LRGLATHE)
     RESOUREQ(PLANT1,CARVER)
     RESOUREQ(PLANT1,LABOR)
     RESOUREQ(PLANT1,TOP)
     RESOUREQ(PLANT2,SMLLATHE)
     RESOUREQ(PLANT2,LRGLATHE)
     RESOUREQ(PLANT2,CARVER)
     RESOUREQ(PLANT2,LABOR)
  **** Warning  This =L= constraint causes all
               variables in it to have a zero solution value
               since the nonnegative variables present
                        have only 0 or + coefficients
                     the nonpositive variables present
                        have only 0 or - coefficents
                     the unrestricted variables
                        have only zero coefficients
                     and the rhs is zero.

 ##  RESOUREQ(PLANT2,TOP)
```

Table 8.14 Picture of Basic Example

Panel B - Model Schematic

```
                                T T T T T T S S S S N     R
           |                    R R R R R R E E E E E     H
           |                    N N N N N N L L L L T     S     P     N
           |                    S S S S S S L L L L I           O     E     R
           |                    P P P P P P           N     C     S     G     O
           |                    O O O O O O           C     O     I A   A A   W
           | M A K E - M A K E - M A K E R R R R R R           O     E     T I   T I   C
           |                    T T T T T T           M     F     I J     J   N
           |                                          E     F     V ,   V ,   T
           |                                                S     E S   E S   S
           |                  1 1 1 1 1
           | 1 2 3 4 5 6 7 8 9 0 1 2 3 4 1 2 3 4 5 6 1 2 3 4 1
            ---------------------------------------------------
     OBJT 1| E F E E E E E E E E E E E E D D D D 6 6 6 6 C   = 0     2 1     4   2 5
  RESOUREQ 1|     D E E E E D D                               < G       6     0     6
  RESOUREQ 2|     D D D D E E                                 < F       6     0     6
  RESOUREQ 3|     D E D E D E                                 < F       6     0     6
  RESOUREQ 4| D D E D E D E D                                 < G       8     0     8
  RESOUREQ 5| C C                                             < F       2     0     2
  RESOUREQ 6|                 D E E E D D                      < G       6     0     6
  RESOUREQ 7|                 D D D D E E                      < F       6     0     6
  RESOUREQ 8|                 D E D E D E                      < G       6     0     6
  RESOUREQ 9|                 E D E D E D                      < G       6     0     6
 PLANTPROD 1| 3                           C           C       < 0       2     1     3
 PLANTPROD 2|   3                           C           C     < 0       2     1     3
 PLANTPROD 3|     3   3   3                   C 3     D       < 0       2     4     6
 PLANTPROD 4|       3   3   3                       C 3   D   < 0       2     4     6
 PLANTPROD 5|                           3               C     < 0       1     1     2
 PLANTPROD 6|                             3               C   < 0       1     1     2
 PLANTPROD 7|                 3   3   3       3 C         D   < 0       2     4     6
 PLANTPROD 8|                   3   3   3         3 C       D < 0       2     4     6
            ----------------------------------------------------
 POSITIVE   | 3   5   5   5   5   5   5   2   2   2   2   2   1
 COLUMN CTS |   3   5   5   5   5   5   5   2   2   2   2   2
 NEGATIVE   | 1   1   1   1   1   1   1   1   1   1   1   1   0
 COLUMN CTS |   1   1   1   1   1   1   1   1   1   1   1   1
 COLUMN     | 4   6   6   6   6   6   6   3   3   3   3   3   1
 COUNTS     |   4   6   6   6   6   6   6   3   3   3   3   3
            ----------------------------------------------------
```

PANEL A - COEFFICIENT CODES

LOWER BOUND (INCLUSIVE)	CODE	UPPER BOUND (LESS THAN)
1000.00000	G	+INFINITY
100.00000	F	1000.00000
10.00000	E	100.00000
1.00000	D	10.00000
1.00000	C	1.00000
0.50000	B	1.00000
0.00000	A	0.50000
0.00000	0	0.00000
-0.50000	1	0.00000
-1.00000	2	-0.50000
-1.00000	3	-1.00000
-10.00000	4	-1.00000
-100.00000	5	-10.00000
-1000.00000	6	-100.00000
-INFINITY	7	-1000.00000

Panel C - ### Dictionary of Variables

```
            M   1: MAKE(PLANT1,TABLES,NORMAL,FUNCT)
            A   2: MAKE(PLANT1,TABLES,NORMAL,FANCY)
            K   3: MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)
            E   4: MAKE(PLANT1,CHAIRS,NORMAL,FANCY)
            -   5: MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)
            M   6: MAKE(PLANT1,CHAIRS,MAXSML,FANCY)
            A   7: MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)
            K   8: MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)
            E   9: MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)
            -  10: MAKE(PLANT2,CHAIRS,NORMAL,FANCY)
            M  11: MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)
            A  12: MAKE(PLANT2,CHAIRS,MAXSML,FANCY)
            K  13: MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)
            E  14: MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)
  TRNSPORT     1: TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
  TRNSPORT     2: TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
  TRNSPORT     3: TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
  TRNSPORT     4: TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)
  TRNSPORT     5: TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)
  TRNSPORT     6: TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)

  SELL         1: SELL(PLANT1,FUNCT)
  SELL         2: SELL(PLANT1,FANCY)
  SELL         3: SELL(PLANT2,FUNCT)
  SELL         4: SELL(PLANT2,FANCY)
  NETINCOME    1: NETINCOME

 Panel D ----### Dictionary of Equations

OBJT         1: OBJT
RESOUREQ     1: RESOUREQ(PLANT1,SMLLATHE)
RESOUREQ     2: RESOUREQ(PLANT1,LRGLATHE)
RESOUREQ     3: RESOUREQ(PLANT1,CARVER)
RESOUREQ     4: RESOUREQ(PLANT1,LABOR)
RESOUREQ     5: RESOUREQ(PLANT1,TOP)
RESOUREQ     6: RESOUREQ(PLANT2,SMLLATHE)
RESOUREQ     7: RESOUREQ(PLANT2,LRGLATHE)
RESOUREQ     8: RESOUREQ(PLANT2,CARVER)
RESOUREQ     9: RESOUREQ(PLANT2,LABOR)
PLANTPROD    1: PLANTPROD(PLANT1,TABLES,FUNCT)
PLANTPROD    2: PLANTPROD(PLANT1,TABLES,FANCY)
PLANTPROD    3: PLANTPROD(PLANT1,CHAIRS,FUNCT)
PLANTPROD    4: PLANTPROD(PLANT1,CHAIRS,FANCY)
PLANTPROD    5: PLANTPROD(PLANT2,TABLES,FUNCT)
PLANTPROD    6: PLANTPROD(PLANT2,TABLES,FANCY)
PLANTPROD    7: PLANTPROD(PLANT2,CHAIRS,FUNCT)
PLANTPROD    8: PLANTPROD(PLANT2,CHAIRS,FANCY)
```

Table 8.15 Picture of Thoroughly Messed Up Example

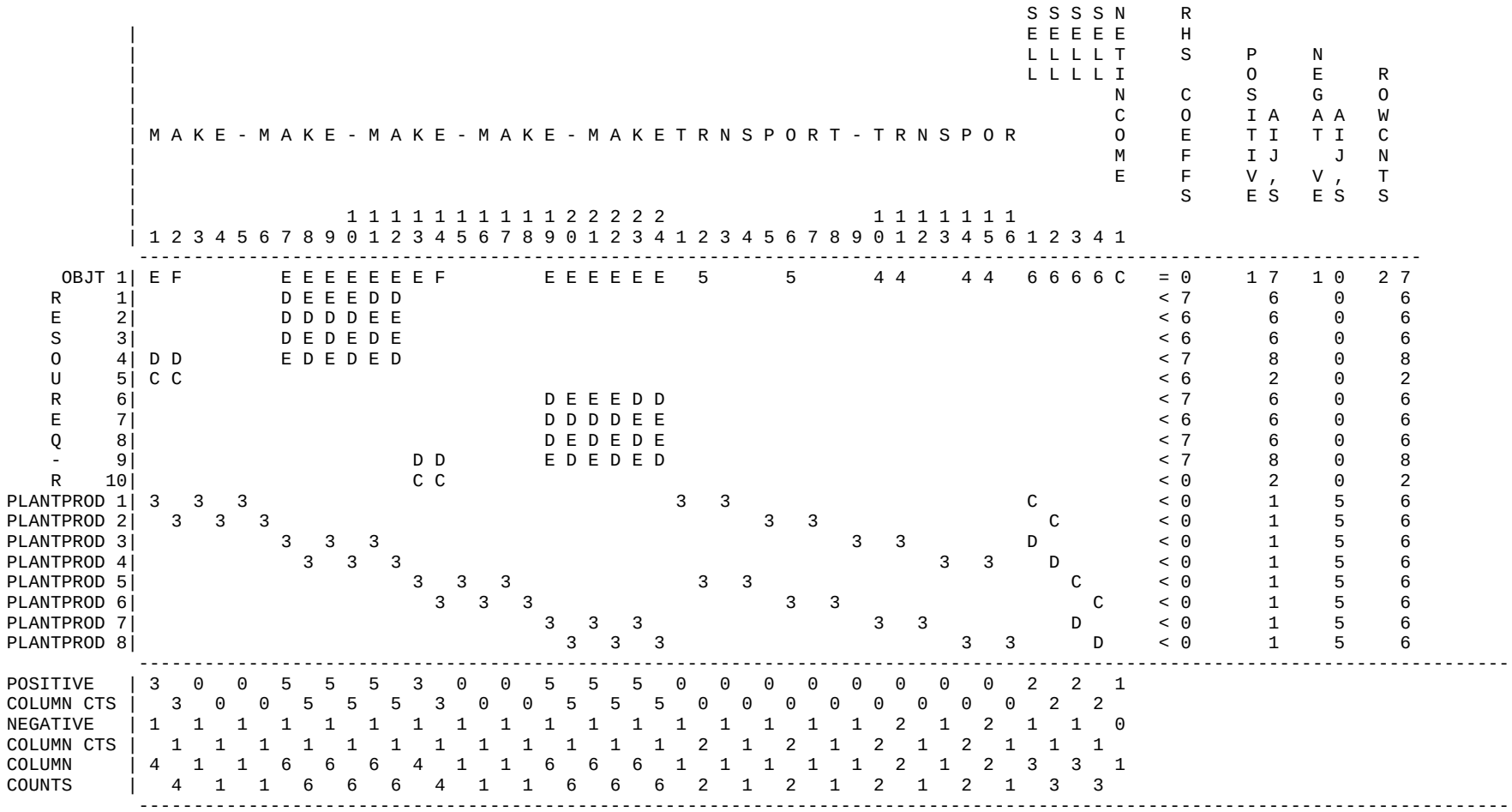


Table 8.14 Picture of Thoroughly Messed Up Example (continued)

```
Panel C----### Dictionary of Variables
           M   1: MAKE(PLANT1,TABLES,NORMAL,FUNCT)
           A   2: MAKE(PLANT1,TABLES,NORMAL,FANCY)
           K   3: MAKE(PLANT1,TABLES,MAXSML,FUNCT)
           E   4: MAKE(PLANT1,TABLES,MAXSML,FANCY)
           -   5: MAKE(PLANT1,TABLES,MAXLRG,FUNCT)
           M   6: MAKE(PLANT1,TABLES,MAXLRG,FANCY)
           A   7: MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)
           K   8: MAKE(PLANT1,CHAIRS,NORMAL,FANCY)
           E   9: MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)
           -  10: MAKE(PLANT1,CHAIRS,MAXSML,FANCY)
           M  11: MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)
           A  12: MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)
           K  13: MAKE(PLANT2,TABLES,NORMAL,FUNCT)
           E  14: MAKE(PLANT2,TABLES,NORMAL,FANCY)
           -  15: MAKE(PLANT2,TABLES,MAXSML,FUNCT)
           M  16: MAKE(PLANT2,TABLES,MAXSML,FANCY)
           A  17: MAKE(PLANT2,TABLES,MAXLRG,FUNCT)
           K  18: MAKE(PLANT2,TABLES,MAXLRG,FANCY)
           E  19: MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)
           -  20: MAKE(PLANT2,CHAIRS,NORMAL,FANCY)
           M  21: MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)
           A  22: MAKE(PLANT2,CHAIRS,MAXSML,FANCY)
           K  23: MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)
           E  24: MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)
           T   1: TRNSPORT(TABLES,FUNCT,PLANT1,PLANT1)
           R   2: TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
           N   3: TRNSPORT(TABLES,FUNCT,PLANT2,PLANT1)
           S   4: TRNSPORT(TABLES,FUNCT,PLANT2,PLANT2)
           P   5: TRNSPORT(TABLES,FANCY,PLANT1,PLANT1)
           O   6: TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
           R   7: TRNSPORT(TABLES,FANCY,PLANT2,PLANT1)
           T   8: TRNSPORT(TABLES,FANCY,PLANT2,PLANT2)
           -   9: TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT1)
           T  10: TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
           R  11: TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)
           N  12: TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT2)
           S  13: TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT1)
           P  14: TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)
           O  15: TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)
           R  16: TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT2)
  SELL          1: SELL(PLANT1,FUNCT)
  SELL          2: SELL(PLANT1,FANCY)
  SELL          3: SELL(PLANT2,FUNCT)
  SELL          4: SELL(PLANT2,FANCY)
  NETINCOME     1: NETINCOME

  Panel D----### Dictionary of Equations
    OBJT          1: OBJT
             R    1: RESOUREQ(PLANT1,SMLLATHE)
             E    2: RESOUREQ(PLANT1,LRGLATHE)
             S    3: RESOUREQ(PLANT1,CARVER)
             O    4: RESOUREQ(PLANT1,LABOR)
             U    5: RESOUREQ(PLANT1,TOP)
             R    6: RESOUREQ(PLANT2,SMLLATHE)
             E    7: RESOUREQ(PLANT2,LRGLATHE)
             Q    8: RESOUREQ(PLANT2,CARVER)
             -    9: RESOUREQ(PLANT2,LABOR)
             R   10: RESOUREQ(PLANT2,TOP)
    PLANTPROD     1: PLANTPROD(PLANT1,TABLES,FUNCT)
    PLANTPROD     2: PLANTPROD(PLANT1,TABLES,FANCY)
    PLANTPROD     3: PLANTPROD(PLANT1,CHAIRS,FUNCT)
    PLANTPROD     4: PLANTPROD(PLANT1,CHAIRS,FANCY)
    PLANTPROD     5: PLANTPROD(PLANT2,TABLES,FUNCT)
    PLANTPROD     6: PLANTPROD(PLANT2,TABLES,FANCY)
    PLANTPROD     7: PLANTPROD(PLANT2,CHAIRS,FUNCT)
    PLANTPROD     8: PLANTPROD(PLANT2,CHAIRS,FANCY)
```

Table 8.16 PICTURE for Selected Submatrix

																										RHS	POS	NEG	ROW
		M	A	K	E	-	M	A	K	E	-	M	A	K	E	-	M	A	K	E	-	M	A	K	E	COEF	POSITIV	NEGATIV	ROWCNTS
											1	1	1	1	1	1	1	1	2	2	2	2	2						
		1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4				
REQUR	1							D	E	E	E	D	D													< 7	6	0	6
	2							D	D	D	D	E	E													< 6	6	0	6
	3							D	E	D	E	D	E													< 6	6	0	6
	4	D	D					E	D	E	D	E	D													< 7	8	0	8
	5	C	C																							< 6	2	0	2
	6																		D	E	E	E	D	D		< 7	6	0	6
	7																		D	D	D	D	E	E		< 6	6	0	6
	8																		D	E	D	E	D	E		< 7	6	0	6
	9													D	D				E	D	E	D	E	D		< 7	8	0	8
	10													C	C											< 0	2	0	2
POSITIVE		2		0		0		4		4		4		2		0		0		4		4		4					
COLUMN CTS			2		0		0		4		4		4		2		0		0		4		4		4				
NEGATIVE		0		0		0		0		0		0		0		0		0		0		0		0					
COLUMN CTS			0		0		0		0		0		0		0		0		0		0		0		0				
COLUMN COUNTS		2		0		0		4		4		4		2		0		0		4		4		4					
			2		0		0		4		4		4		2		0		0		4		4		4				

Table 8.17 BLOCKPIC Output

Panel A. Aggregate Block Picture

	MAKE	PORT	SELL	CONSUME		RHS
OBJT	+	-	-	+	E	0
RESOUREQ	+				L	-
PLANTPROD	-	-	+		L	0
Variable Typ	+	+	+	u		

Panel B. Picture Giving Number of Coefficients by Block -- Strip 1

	MAKE	TRNSPORT	SELL	CONSUME		RHS	Coeff Cnts	# of Eqns
OBJT	16+			1+	E	17+	1	
		6-	4-			10-		
RESOUREQ	56+				L	56+	10	
						9-		
PLANTPROD			8+		L	8+	8	
	24-	16-				40-		
Coeff Cnts	72+		8+	1+		81+		
	24-	22-	4-			9- 50-		
# of Vars	24	16	4	1				
Variable Typ	>=0	>=0	>=0	<0>				

Panel C. Picture Giving Average Number of Coefficients by Column Block

```
            |                                                  C       #
            |                                 N                f
            |              T                  E                s       o
            |              R                  T                        f
            |              N                  I                P
            |              S                  N                e
            |      M       P       S          C                r       E
            |      A       O       E          O          R     E       q
            |      K       R       L          M          H     q       n
            |      E       T       L          E          S     u       s
 ---------------------------------------------------------------------------
 OBJT       |  0.667+                        1+ | E             17+      1
            |          0.375-      1-           |                10-
 RESOUREQ   |  2.333+                           | L               5+     10
            |                                   |        9-
 PLANTPROD  |                        2+         | L               1+      8
            |      1-      1-                   |                 5-
 ---------------------------------------------------------------------------
 Cfs PerVar |      3+              2+      1+ |
            |      1-  1.375-      1-         |
 # of  Vars |     24      16       4       1
  Var Type  |    >=0     >=0     >=0     <0>
 ---------------------------------------------------------------------------
```

Panel D. Scaling Data - Maximum & Minimum Coefficients by Block

```
              |                                          R         E
              |                                 N        H         q
              |                  T              E        S         u
              |                  R              T
              |                  N              I        M         M
              |                  S              N        a         a
              |        M         P         S    C        x         x
              |        A         O         E    O        M         M
              |        K         R         L    M        i         i
              |        E         T         L    E        n         n
 ---------------------------------------------------------------------------
 OBJT       Max|       100        18       850        1               850
            Min|        15         5       400        1                 1
 RESOUREQ   Max|        17                                1750        17
            Min|         1                                 500         1
 PLANTPROD  Max|         1         1         6                         6
            Min|         1         1         1                         1
 ---------------------------------------------------------------------------
  Total Var Max|       100        18       850        1      1750
            Min|         1         1         1        1       500
```

Panel E Analysis of Variables (treating nonlinear terms at their starting points)

```
**** ERROR These variables are unbounded -- they
          have a desirable objective function coeffic
          all 0 or + coefficients in the =G= rows
          all 0 or - coefficients in the =L= rows
          and no coefficients in the =E= rows
 ##  TRNSPORT
```

Panel F Analysis of Equations (treating nonlinear terms at their starting points)

```
 **** ERROR This =L= constr. causes an infeasible model
              since the nonnegative variables present have only 0 or + coefficients
                    the nonpositive variables present have only 0 or - coefficents
                    the unrestricted variables have only zero coefficients
                    and the RHS is negative
 ##  RESOUREQ
```

Table 8.18 BLOCKPIC Aggregate Block Picture -- ASM Example

	DEMAND	IMPORT	EXPORT	DEMAND	CROPBUDGET	LVSTBUDGET	LANDSUPPLY	AUMSPUB	AUMSPRIV	PROCESS	WATERFRX	WATERVAR	FAMILYLIM	HIRELIM	NATMIXR	CCLOANPR	DEFPRODN	PRD5092	DIVERTPRD	ARTS	CSPS	TOLR	TWID		RHS
OBJT	-	+	-	-	m	+	+	+	+	+	+	+	+	+		-	-	-		+	+			E	+
MAXLAND					+	+																		L	+
MINLAND					+																			G	0
LAND					+	+	-																	L	0
AUMSR						+		-	-															L	0
PUBAUMS								+																L	+
WATERR					+						-	-												L	0
FIX											+													L	+
LABOR					+								-	-										L	0
FAMILYLIM													+											L	+
HIRELIM														+										L	+
PRIMARYBAL	+	-	+		-	m				+						+	-			-				L	0
SECONDBAL				+		+				-														L	0
MIXREG					+										-								+	E	0
MIXREGTOT					+										-									E	0
MIXNAT						+										-						+		E	0
FRMPROG					-												+							L	0
P5092																		+						L	0
DIVERT																			+					L	0
Variable Typ	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	u	+	+		

Table 8.19 BLOCKLIST Output

```
 ----### List of Variable Block Characteristics

 Note Max and Min do not include Objective Row

 Variable   Sign  Numb Numb Pos  Neg  Nonl  Maximum   Minimum
   Block    Res   Vars Nonl Coef Coef Coef  Absolute  Absolute
 MAKE       >=0    24    0   72   24    0    17.00     1.000
 TRNSPORT   >=0    16    0    0   22    0    1.000     1.000
 SELL       >=0     4    0    8    4    0    6.000     1.000
 NETINCOME  <0>     1    0    1    0    0   0.0000E+000.0000E+00


 ----### List of Equation Block Characteristics

 Note Max and Min do not include RHS and Objective variable

 Equation   Type  Numb Numb Pos  Neg  Nonl Pos  Neg  Maximum  Minimum
   Block    Res   Eqns Nonl Coef Coef Coef RHS  RHS  Absolute  Absolute
 OBJT       =E=     1    0   17   10    0    0    0   850.0     5.000
 RESOUREQ   =L=    10    0   56    0    0    0    9   17.00     1.000
 PLANTPROD  =L=     8    0    8   40    0    0    0   6.000     1.000

   ----### Analysis of Variables ( nonlinear terms at current point)



**** ERROR These variables are unbounded -- they
           have a desirable objective function coeffic
           all 0 or + coefficients in the =G= rows
           all 0 or - coefficients in the =L= rows
           and no coefficients in the =E= rows
 ##  TRNSPORT

 ----### Analysis of Equations

 **** ERROR This =L= constr. causes an infeasible model
              since the nonnegative variables present
  have only 0 or + coefficients
                    the nonpositive variables present
  have only 0 or - coefficents
                    the unrestricted variables
  have only zero coefficients
                    and the RHS is negative
 ##  RESOUREQ
```

TABLE 8.20 ABSTRACT DISPLAYCR OUTPUT(REPLACE SECOND HALF)

```
----###    DISPLAYING EQUATIONS

 ##  RESOUREQ(PLANT1,SMLLATHE)
 MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)                          8.0000
 MAKE(PLANT1,CHAIRS,NORMAL,FANCY)                          12.000
 MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)                          13.000
 MAKE(PLANT1,CHAIRS,MAXSML,FANCY)                          17.000
 MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)                          2.0000
 MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)                          5.0000
         =L=                                              -1100.0
 ##  RESOUREQ(PLANT1,LRGLATHE)
 MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)                          5.0000
 MAKE(PLANT1,CHAIRS,NORMAL,FANCY)                          7.0000
 MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)                          2.0000
 MAKE(PLANT1,CHAIRS,MAXSML,FANCY)                          3.0000
 MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)                          13.000
 MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)                          15.000
         =L=                                              -880.00
 ##  RESOUREQ(PLANT1,CARVER)
 MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)                          4.0000
 MAKE(PLANT1,CHAIRS,NORMAL,FANCY)                          10.000
 MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)                          4.0000
 MAKE(PLANT1,CHAIRS,MAXSML,FANCY)                          10.000
 MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)                          4.0000
 MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)                          10.000
         =L=                                              -500.00
 ##  RESOUREQ(PLANT2,LABOR)
 MAKE(PLANT2,TABLES,NORMAL,FUNCT)                          3.0000
 MAKE(PLANT2,TABLES,NORMAL,FANCY)                          5.0000
 MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)                          10.000
 MAKE(PLANT2,CHAIRS,NORMAL,FANCY)                          8.0000
 MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)                          11.000
 MAKE(PLANT2,CHAIRS,MAXSML,FANCY)                          8.0000
 MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)                          11.000
 MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)                          8.0000
         =L=                                              -1250.0
 ##  RESOUREQ(PLANT2,TOP)
 MAKE(PLANT2,TABLES,NORMAL,FUNCT)                          1.0000
 MAKE(PLANT2,TABLES,NORMAL,FANCY)                          1.0000
         =L=
0.00000E+00
```

TABLE 8.21 MATCHIT OUTPUT

```
----#### Executing MATCHIT
 Note Max and Min do not include Obj row coef
                                    Is  Tot Pos Neg Nln  Minimum   Maximum
  ----### Requested Variables       Non Cof Cof Cof Cof  Absolute  Absolute

 ----## VAR TRNSPORT

 TRNSPORT(TABLES,FUNCT,PLANT1,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)   0   2   0   2   0   1.000      1.000
 TRNSPORT(TABLES,FUNCT,PLANT2,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(TABLES,FUNCT,PLANT2,PLANT2)   0   1   0   1   0   1.000      1.000
 TRNSPORT(TABLES,FANCY,PLANT1,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)   0   2   0   2   0   1.000      1.000
 TRNSPORT(TABLES,FANCY,PLANT2,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(TABLES,FANCY,PLANT2,PLANT2)   0   1   0   1   0   1.000      1.000
 TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)   0   2   0   2   0   1.000      1.000
 TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)   0   2   0   2   0   1.000      1.000
 TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT2)   0   1   0   1   0   1.000      1.000
 TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT1)   0   1   0   1   0   1.000      1.000
 TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)   0   2   0   2   0   1.000      1.000
 TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)   0   2   0   2   0   1.000      1.000
 TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT2)   0   1   0   1   0   1.000      1.000



                                  Numb  Numb  Total  Pos   Neg  Nonln
 ----### Variable Request         Varia Nonln  Coef  Coef  Coef  Coef

 MAKE(PLANT2)                      12     0    48    36    12     0
 TRNSPORT                          16     0    22     0    22     0

 Note Max and Min do not include Rhs and Obj var coef

                                    Is  Tot Pos Neg Nln  Minimum   Maximum
  ----### Requested Equations       Non Cof Cof Cof Cof  Absolute  Absolute

 ----## EQU RESOUREQ

 RESOUREQ(PLANT1,SMLLATHE)           0   7   7   0   0   2.000      17.00
 RESOUREQ(PLANT1,LRGLATHE)           0   7   7   0   0   2.000      15.00
 RESOUREQ(PLANT1,CARVER)             0   7   7   0   0   3.000      10.00
 RESOUREQ(PLANT1,LABOR)              0   9   9   0   0   1.000      11.00
 RESOUREQ(PLANT1,TOP)                0   3   3   0   0   1.000      8.000
 RESOUREQ(PLANT2,SMLLATHE)           0   7   7   0   0   2.000      17.00
 RESOUREQ(PLANT2,LRGLATHE)           0   7   7   0   0   2.000      15.00
 RESOUREQ(PLANT2,CARVER)             0   7   7   0   0   3.000      10.00
 RESOUREQ(PLANT2,LABOR)              0   9   9   0   0   1.000      11.00
 RESOUREQ(PLANT2,TOP)                0   3   2   1   0   1.000      1.000

 ----## EQU PLANTPROD

 PLANTPROD(PLANT1,TABLES,FUNCT)      0   7   1   6   0   1.000      1.000
 PLANTPROD(PLANT1,TABLES,FANCY)      0   7   1   6   0   1.000      1.000
 PLANTPROD(PLANT1,CHAIRS,FUNCT)      0   7   1   6   0   1.000      4.000
 PLANTPROD(PLANT1,CHAIRS,FANCY)      0   7   1   6   0   1.000      6.000
 PLANTPROD(PLANT2,TABLES,FUNCT)      0   7   1   6   0   1.000      1.000
 PLANTPROD(PLANT2,TABLES,FANCY)      0   7   1   6   0   1.000      1.000
 PLANTPROD(PLANT2,CHAIRS,FUNCT)      0   7   1   6   0   1.000      4.000
 PLANTPROD(PLANT2,CHAIRS,FANCY)      0   6   1   5   0   1.000      6.000

                                  Numb  Numb  Total Pos   Neg   Nonln
  ----### Equation Request        Equat Nonln Coef  Coef  Coef  Coef

 RESOUREQ                          10     0    66    65     1     0
 PLANTPROD(PLANT1)                  4     0    28     4    24     0
 *RES*                             19     0   149    91    58     0
```

Chapter 9 Post Solution Model Analysis

Unfortunately, regardless of how much care is taken when formulating and checking models, cases occur when the solver terminates with a bad solution. The model may be found to be unbounded, infeasible, or worst yet optimal, but with an unrealistic answer. In these cases, one begins the sometimes arduous task of figuring out what is wrong. There are tools provided in GAMS and in GAMSCHK to help in this process. This chapter covers approaches to such problems using those tools.

The discussion is organized into three parts addressing models which terminate as: 1) infeasible; 2) unbounded; or 3) optimal, but with an unrealistic solution.

9.1 Correction of Models Which are Infeasible

Infeasibility is always a possible outcome when solving models. Linear programming solvers handle infeasibility by going through a two or three step solution approach. First, there may be some presolution calculations which may determine a model cannot be made feasible (as done by the OSL and CPLEX PRESOLVE options). Second, there is usually a Phase I operation wherein the sum of a set of implicitly added artificial variables is minimized. During this phase, the problem is artificially rendered feasible. Third, if the artificial variable values are all driven to zero, then the problem is declared feasible and the solver turns to the normal (Phase II) simplex method proceeding toward optimally.

However, the problem may not go into phase II and be declared infeasible if the sum of the artificial variables cannot be driven to zero. In such cases, the information content of the differs between solvers and may not be very helpful. Rarely does this output give enough information to directly diagnose and fix the cause of the infeasibility. Cases may even occur

where a PRESOLVE discovers infeasibility and the message is given that the problem has no solution then the solver quits giving back no usable information to GAMS which in turn terminates. Thus, the LST file may contain little information. More commonly, the solution part of the LST file contains particular variables or equations tagged as infeasible with the marker INFES (as in Table 8.7, Panel B). In addition when using ANALYSIS, some presolves and the IIS procedures in CPLEX, there may be an identification of restrictions involved with the infeasibility. Unfortunately, ANALYSIS and the presolves cannot always detect infeasibility causes (Use of Chennicks IIS will always diagnose at least part of the problem). Thus, many modelers will sometimes have to contend with a model that is infeasible without a great deal of information on what is causing that condition.

9.1.1 Causes of Infeasible Models

Causes of infeasibility are not always easily identified. Solvers may report a particular equation as infeasible in cases where an entirely different equation is the cause. Consider the following example (infe.gms),

$$
\begin{array}{lrcrcl}
\text{Max} & 50X_1 & + & 50X_2 & & \\
\text{s.t.} & X_1 & + & X_2 & \leq & 50 \\
 & 50X_1 & + & X_2 & \leq & 65 \\
 & X_1 & & & \geq & 20 \\
 & X_1 & , & X_2 & \geq & 0
\end{array}
$$

In this example, the interaction between the constraint $X_1 \geq 20$, the constraint immediately above it, and the nonnegativity condition on X_2 render the model infeasible. There may be several potential explanations as to why the infeasibility is present. The 65 on the right hand side of the

second constraint may be a data entry error, perhaps a number in excess of 1000 was intended. Similarly, the 50 for X_1 in the second constraint may be an error with a number more like 0.50 or a negative entry intended. Third, the limit requiring $X_1 \geq 20$ may be misspecified with the RHS really intended to be 0.20. Fourth, perhaps the X_2 variable should have been allowed to be negative. Fifth, there could be multiple errors involving several of the above cases. Runs with OSL, CPLEX, BDMLP and MINOS5 resulted in the marking of either the $X_1 \geq 20$ or the second constraint as the infeasible item. This may or may not be a proper identification of the problem causing mistake. Generally, infeasibilities occur because of the interaction of multiple variables and equation restrictions. In more complex models a set of 50 constraints could be involved. Thus, we need procedures to find the involved set of variable and equation restrictions. In turn we can look for the root cause of the infeasibility in the model subcomponent.

9.1.2 Finding Causes of Infeasibility -- Basic Theory

There are two approaches we will recommend for finding causes of an infeasibility. The fist approach relies on “artificial” variables . The second is due to Chinneck and finds an “irreducible infeasible system” - IIS. We will only briefly cover the IIS approach as it requires special software and the only GAMS solver that incorporates it is CPLEX.

The concept of an artificial variable is introduced in virtually every course or book which treats linear programming algorithms. An artificial variable is a new variable added to an equation to guarantee that the equation is “artificially” feasible. However, in order to attain real feasibility, the model must be modified to provide an incentive to remove the artificial variables from the solution. There are two ways such incentives are entered: through the “Big M” penalty method or through the “Phase I/ Phase II” optimization approach.

Suppose we review the so called Big M method. In that case we augment the above model with an artificial variable as follows (infeart.gms):

$$
\begin{array}{lrcrcrcl}
\text{Max} & 50X_1 & + & 50X_2 & - & 1000000000A & & \\
 & X_1 & + & X_2 & & & \leq & 50 \\
 & 50X_1 & + & X_2 & & & \leq & 65 \\
 & X_1 & & & + & A & \geq & 20 \\
 & X_1, & & X_2, & & A & \geq & 0
\end{array}
$$

with A being the artificial variable. Such a variable would be entered for each model equation which could not be satisfied by setting all the X's to zero (i.e., $X_1 = 0$ is not feasible in the $X_1 \geq 20$ constraint). Each artificial has a very large, undesirable objective function coefficient (the so called "Big M value") and an entry in the potential infeasible equation which allows its satisfaction. Such a variable "artificially" allows satisfaction of the equations, but at a very high cost. Because of this large cost A is very unattractive, since the objective function cost of setting A to 20 are far in excess of any possible objective function value involving the X's. Thus, if at all possible the solution process would drive A from the solution. However in the solution to this problem the artificial cannot be driven from the basis. In turn the solver would indicate the problem is infeasible. The Big M method is not used in solvers because of difficulties in determining the appropriate magnitude of the objective function penalty and the potential introduction of numerical instability.

Solvers virtually without exception employ the Phase I/Phase II solution method where the artificial variables are implicitly (and automatically) added, then an auxiliary objective function is used which minimizes the sum of the artificial variables during Phase I (possibly with some

weight attached to the real objective function). Once a feasible solution is found the solver drops the artificials and optimizes the real objective function.

Under either case the general approach to solving potentially infeasible problems is to introduce variables which "artificially" guarantee a feasible solution, then alter the objective function to drive those variables out. Problems wherein the artificials cannot be driven to zero are reported as infeasible.

9.1.3 Diagnosing Infeasible Solutions

Our contention is that we can use the information from a solution with artificials present to diagnose the cause of the infeasibility. Suppose we illustrate this point by example.

A GAMS formulation of the above problem including the artificial in both Big M and Phase I implementations is given in Table 9.1, Panel A(infeart.gms). The resultant Big M solution is given in Panel B, while Panel C contains the Phase I solution. In both cases, the artificial variable is in the basis having a non-zero level of 18.7 indicating the model is infeasible.

The question then is: So what? When a linear programming is solved, the optimum solution contains a number of output items which are influenced by the objective function parameters from the basic variables. In particular the shadow prices, reduced costs and objective function value are all a function of the objective function coefficients of the basic variables as follows:

$$u=C_B B^{-1}$$

$$Z_j-C_j = C_B B^{-1}A_j - C_j$$

$$z= C_B B^{-1} b$$

where u is the vector of shadow prices for the model constraint equations also called the

marginals in gams;

C_B is the objective function coefficients for the basic variables ;

B^{-1} is the inverse of the basis matrix;

Z_j-C_j is the reduced costs for the jth nonbasic variable also called the marginals for the variables in GAMS ;

A_j is the coefficients in the constraint equations associated with the jth nonbasic variable;

C_j is the objective function coefficient associated with the jth nonbasic variable; and

b is the vector of right hand side coefficients.

The above formula for u is interpretable as the marginal rate of change in the objective function when the right hand side on the constraint is altered (see McCarl and Spreen or Bazzara, Jarvis and Sherali). The of Z_j-C_j formula estimates the amount the objective function will change when the nonbasic variable takes on a nonzero value. This can be interpreted as the shadow price on the nonnegativity condition for the variable and thus the value of allowing the variable to take on a negative value thereby relaxing the constraint. The objective function value is to simply a number.

Now when artificial is in the basis, then relaxation of some of the right hand sides will cause that artificial to become smaller. These constraints would have artificially large shadow prices since in those cases right hand side relaxation greatly reduce the size of the optimal objective function since the artificials generally form a large proportion of the optimal objective function value. Similarly the reduced costs will be influenced by the presence of the artificial. The solution marginal information also would similarly identify the constraints and variable

bounds associated with the infeasibility under a Phase I approach. In particular, the Phase I shadow prices and reduced costs apply to the sum of the infeasibilities telling how much relaxing the right hand sites or variable bounds would reduce the sum of the infeasibilities.

Thus the marginals (or in non GAMS terminology the shadow prices or dual variables and reduced costs) reveal the constraints which are party to the infeasibility. Under the Big M formulation the marginals for the variable X_2 as well as the marginals for the second and third constraints are very large showing data revisions therein would alter the solution level of the artificial. Under the Phase I solution, the marginals are non-zero in the same places indicating that alterations therein would reduce the sum of the infeasibilities. Thus, the solution contains the signal that there is something wrong in the interaction of X_2, the second constraint, and the third constraint. In turn, a modeler would examine these items and the other associated variables to fix the infeasibility. Hopefully, then the underlying error would be found.

The above material indicates two ways of finding the cause of infeasibilities. First, if the solution reports back the Phase I shadow prices, then examine the variables and constraints associated with those to find the cause of the infeasibility. Otherwise, set up the model with artificials present, solve it and if the artificials appear in the solution, look for distorted shadow prices and marginals to find the causal set of equations. One word of caution, this will always identify some of the infeasibility causes, but in the face of a nonunique dual solution caused by degeneracy or alternative optimals may not reveal them all. Thus, multiple applications of the procedure may be needed.

9.1.4 Details on Artificial Variable Approach to Resolving Infeasibility

After one is taught about artificial variables, often the assertion is made that solvers automatically add them and thus artificials are of no further concern. Unfortunately, when solvers reach an infeasibility, the LST file does no always contain the Phase I marginal information. In particular, some solvers give marginals solely from the original objective, not the Phase I objective function. Thus, we recommend that those looking for infeasibility causes (in the absence of specialized infeasibility finding procedures such as IIS) set up their own Big M or Phase I based problem, solve it and then use the marginals to find the infeasibility causing set.

The following gives the steps in a Big M based, artificial variable based approach for finding infeasibility causes.

Step 1	Identify the relevant equations and/or variable bounds for which artificials are needed to be added (details about this in next section)
Step 2	Add artificial variables to those equations and bounds. These artificials each have a Big M penalty in the objective function and an entry in a single constraint.
Step 3	Solve the model
Step 4	Examine the model solution. Where the marginals (the reduced costs for the variables and the shadow prices for the equations) are distorted by the presence of the artificials, identify those as the variables and equations to be examined for the cause of infeasibility
Step 5	Fix the model and repeat the process if needed

There are several questions inherent in the above procedure. In particular: Where should artificial variables be added?; How should the artificial variables be structured?; and How does one find a

"distorted" marginal? Each is discussed below.

9.1.4.1 Where Should Artificial Variables be Added?

The places where artificial variables should be added can be determined in several ways. One could look at the model solution and enter artificials in the equations and/or variable bounds marked by the solver as infeasible. However, while this sometimes points to proper places, it does not always do such. The approach advocated here is to add artificials in all possible infeasible locations.

Programming models will only be infeasible when setting all the decision variables equal to zero is not feasible. This occurs when: a) the interval between variable upper and lower bounds does not include zero; or b) equations appear which are not satisfied when all variables are set to zero. The equation cases which are not satisfied when the variables equal zero are:

1) Less than or equal to constraints with a negative right side i.e. $x \leq -1$
2) Greater than or equal to constraints with a positive right side. i.e. $x \geq 1$
3) Equality constraints with a nonzero right side i.e. $x = 1$ or $x = -1$

In addition, when the interval between lower and upper bounds on a variable does not include zero then those bounds need to be converted to constraints with artificials added. This will occur when:

1) the lower bound is positive, or
2) the upper bound is negative

The ADVISORY and NONOPT -- IDENTIFY procedures in GAMSCHK have been written to create a list of all occurrences of these five cases. The output from these procedures for the messed up model contained in Table 8.2, appears in Table 9.2. In that output the

RESOUREQ equations are identified as the potential places where infeasibility may occur. These are the places where the artificial variables are needed.

9.1.4.2 Entering Artificial Variables in GAMS

Once one has found where the artificial variables need to be added one still has to address the questions: How they should be added? and What should they look like? The following general rules address this question. A new variable should be defined for each equation with the same dimension as the infeasible cases in the equation. Thus, if artificials are added to an equation defined with dimensions like RESOUREQ(PLANT,RESOURCE), then one should define a variable like ARTRESOUR(PLANT,RESOURCE). Strictly speaking, such terms only need to be added in the cases identified above. However, given the power of GAMS it may be easier to add them to all cases for a named equation. The next question is how should the artificials be structured. Artificials should be entered with a coefficient of plus or minus one in the potentially infeasible equations. The sign of the coefficient depends on equation type. The artificials should have one coefficient in just a single equation (along with the objective function entry) and the coefficient in the equations should be a:

a) negative one in =L= (≤) equations with negative RHS

b) plus one in =G= (≥) equations with positive RHS

c) plus or minus one with being the same as that of the RHS in =E= (=) equations

For example in our messed up problem we would add the artificial variables with negative one coefficients to the RESOUREQ equations so the resultant equation appears as follows.

```
RESOUREQ(PLANT,RESOURCE)..
   SUM((PRODUCT,TYPE,METHOD)
        ,RES(RESOURCE,PRODUCT,TYPE,METHOD)
        *MAKE(PLANT,PRODUCT,METHOD,TYPE) )
        -ARTRESOUR(PLANT,RESOURCE)
```

```
        =l= -RESORAVAIL(RESOURCE,PLANT) ;
```

Note, one cannot add an artificial into the GAMS upper and lower bound defined with the variablename.LO, .UP or .FX syntax. Thus, one needs to convert positive LO, negative UP and nonzero FX bounds into equations and then add the artificials. The LO, UP and FX conditions need to be converted =G=, to =L= and =E= equations respectively. The above rules on signs can then be used.

One will also need to enter a large penalty as the objective function coefficient for the artificials. The coefficient will be negative in a maximization context and positive in a minimization problem. The magnitude of this penalty is entirely problem dependent and can cause numerical problems in the solver. All that can be said in general is that the penalty should dwarf the other objective function coefficients and should be large enough so that the artificial is driven to zero in any feasible model. In our example we enter this term to the objective function as follows:

```
OBJT..      NETINCOME =E=
              SUM((PLANT,TYPE),
                    PRICE(PLANT,TYPE) * SELL(PLANT,TYPE))
                  - SUM((PLANT,PRODUCT,METHOD,TYPE)
                  , MAKE(PLANT,PRODUCT,METHOD,TYPE)
            * PRODCOST(PRODUCT,METHOD,TYPE))
      - SUM((PRODUCT,TYPE,PLANT,PLANTS)
                  , TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
                  * TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS))
      - 1000000000*sum((PLANT,RESOURCE),
                       ARTRESOUR(PLANT,RESOURCE))  ;
```

where the -1000000000 is the large penalty.

Alternatively if numerical problems are plaguing the solution with the artificials entered, one can define a new objective function to be maximized or zero out the old terms then proceed. In the case above to minimize the sum of the infeasibilities one would alter the objective to be:

```
OBJNEW.       SUMINFES =E=
              sum((PLANT,RESOURCE),
                        ARTRESOUR(PLANT,RESOURCE))  ;
```

and alter the problem so it had the new variable SUMINFES and minimized that variable.

Similarly, one could just multiply the original objective by zero and make the sum of the artificials into the effective objective function then solve the model as normal with the original SOLVE statement as follows:

```
OBJT..        NETINCOME =E=
              0*[
                   SUM((PLANT,TYPE),
                      PRICE(PLANT,TYPE) * SELL(PLANT,TYPE))
                    - SUM((PLANT,PRODUCT,METHOD,TYPE)
                    , MAKE(PLANT,PRODUCT,METHOD,TYPE)
              * PRODCOST(PRODUCT,METHOD,TYPE))
       - SUM((PRODUCT,TYPE,PLANT,PLANTS)
                    , TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
                    * TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS))
                ]
       - sum((PLANT,RESOURCE),
                        ARTRESOUR(PLANT,RESOURCE))  ;
```

9.1.4.3 How Are Distorted Marginals Identified?

The next question involves finding the distorted marginals. Under the BIG M method one reviews the output in the GAMS LST file as in Table 9.1, Panel B looking for marginals with large absolute values while under the phase I method one would find non zero marginals. However, in models with thousands of variables and equations this information can be well hidden. The GAMSCHK procedure NONOPT has been written to help in this quest. All items with marginals larger in absolute value than 10 to a filter value are output as potential causes of the infeasibility when NONOPT is run on a feasible problem.

Suppose we illustrate this. Let us cause the example from Chapter 8 to be infeasible by taking the model as structured in Table 8.1 and negating the right hand side on the RESOUREQ

equation in line106. Now adding the artificials exactly as they were specified above (i.e. in the GAMS instructions for the BIG M implementation in Section 9.1.4.2) and running the solver, we get an optimal solution with nonzero artificials and a large negative objective value (see the solution in Table 9.3, Panel A). In turn NONOPT gives the messages in Table 9.3, Panel B. That output identifies some of lower bounds on the MAKE variables (their nonnegativity restrictions) and the RESOUREQ equations as likely contributors to the infeasibility. In turn one could examine those items to find the infeasibility cause.

Usage of NONOPT for these purposes requires that the marginal filter value be set in conjunction with the size of the penalties on the artificials. The default filter is 6, thus all marginals greater in absolute value than 10^6 will be reported. We recommend that for the artificial penalties be greater than 10 to the filter in absolute value by at least a factor of 1000 (that is why we use such a large number). One can change this filter by using the GAMSCHK option file employing the MARGFILT option. If one is using the phase 1 approach then the marginal filter can be set small say to -5 (Try this with the phase I formulation in blockarti.gms).

9.1.5 Using IIS

An alternative method for finding infeasibility causes involves using Chinneck's IIS. The I IS procedure invokes infeasibility finding techniques which discover sets of constraints and bounds on variables (including nonnegativity conditions) which cause a problem to be infeasible. Namely, the IIS procedure discovers an "irreducible infeasible set"(IIS). An IIS set consists of equations and/or variable bounds which are collectively infeasible, but become feasible if any one of them is deleted from the set. These sets are discovered by an algorithmic procedure which temporarily relaxes constraints and variable bounds either by dropping them or by adding artificial

variables until the IISs are found. We will not go further into the exact procedures employed by the IIS implementation. The mathematical programming theory basis and algorithmic implementation details are presented in a number of papers by Chinneck and various co-author's. The procedure is carried out by CPLEX on request in an automatic fashion.

Suppose, we provide an example which illustrates IIS use and output. In particular, suppose we use the infeasible problem from section 9.1. An initial solution of this model reveals it to be infeasible. Then we solve the problem again with CPLEX with the IIS procedure active. In turn the IIS procedure will identify the model constraints related to the infeasibility. This requires that we provide an options file which suppresses the presolve and invokes IIS (one should suppress the presolve as it can cause CPLEX to terminate without running the IIS). The options file we use in this case is:

```
presolve 0
iis 1
```

We also enter commands into our problem file (infecp.gms) that request CPLEX be used and make the options file active. The file submitted then is:

```
variable           objmax;
positive variables x1, x2,A
equations          obj,  r1,  r2,   r3;
obj.. objmax =e= 50*x1 +50*x2  ;
r1..               x1 +   x2    =L=  50;
r2..             50*x1 +   x2    =L=  65;
r3..               x1            =G=  20;
model infe /all/;
infe.optfile=1;
option lp=cplex;
solve infe using lp maximizing objmax;
```

Following solution the following IIS output appears in the LST file:

```
Problem infeasible.
IIS found.

An IIS is a set equations and variables (ie a submodel)which
is infeasible but becomes feasible if any one equation or
variable bound is dropped.

A problem may contain several independent IISs but only one
will be found per run.

Number of equations in the IIS:  2.
     upper: R2 < 65
     lower: R3 > 20
Number of variables in the IIS:  1.
     lower: X2 > 0
```

This output shows that the IIS procedure discovered an IIS set consisting of three elements: the second and third constraint equations along with the nonnegativity condition on X2. Thus if a) the second equation was dropped ($50X1+X2 \leq 65$), b) the third equation was dropped ($X1 \geq 20$) or c) the nonnegativity condition on X2 ($X2 \geq 0$); then the problem would become feasible. Ordinarily, the third condition is one modelers do not typically think about as the IIS sets almost always contain an identification of variables which if allowed to become negative relax the infeasible constraint equaions and permit feasibility.

One characteristic of the IIS approach is that it will only discover one set of infeasible equations at a time. To illustrate this, we apply the IIS to the problem from chapter 8 as used in the artificial variable section above (9.1.4.3) where we negate the right hand sides on the RESOURCE constraints. This modification yields a problem containing several IISs, since each of the individual equations is separately infeasible. A run of the IIS yielded the output below.

```
Number of equations in the IIS:  1.
     upper: RESOUREQ(PLANT2,LABOR) < -1250
```

```
Number of variables in the IIS:  6.
     lower: MAKE(PLANT2,CHAIRS,NORMAL,FUNCT) > 0
     lower: MAKE(PLANT2,CHAIRS,NORMAL,FANCY) > 0
     lower: MAKE(PLANT2,CHAIRS,MAXSML,FUNCT) > 0
     lower: MAKE(PLANT2,CHAIRS,MAXSML,FANCY) > 0
     lower: MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT) > 0
     lower: MAKE(PLANT2,CHAIRS,MAXLRG,FANCY) > 0
```

In that output IIS identified one of the RESOURCE equations as being in the infeasibility causing set along with the nonnegativity conditions for six MAKE variables appearing in the equation. Under those circumstances, if one went back and only fixed the first RESOURCE equation, then one would go through several more applications of the IIS procedure before the full set of infeasibilities was uncovered. This contrasts with the artificial variable technique where all of the infeasible equations would be identified in one pass. However with the artificial variable technique there would not be the rigorous identification of a set of constraints such that the removal of any one would render that part of the problem feasible. Rather, there would be a collective identification of all the items involved with the infeasibility which could possibly be in separate IISs but are identified as one big collective set.

This shows the general type of performance one might expect from IIS usage. Namely, the IIS will examine an infeasible problem and find a subset of equations such that the removal of any one renders that problem subset feasible. Typically that subset will involve constraint equations along with variable nonnegativity conditions. Given this IIS identification, one can examine the identified equations and variable nonnegativity conditions and either repair the model or figure other ways to make the problem formulation feasible. The subset of equations in the IIS will not necessarily contain all of the causes of the infeasibility within the total problem structure. Namely, when there are multiple infeasible equation sets then multiple applications of IIS maybe necessary

to find all the problems. However often with modeling in GAMS an IIS identification may lead one to discover block level equation specification problems fixing a number of IISs at once. This would have most likely occurred in the example involving the RESOURCE equation as one would have found the negative right hand sign problem and fixed all of the occurrences at once. Thus, IIS use while conveniently finding a small set of equations may take more than one pass to fully identify all the problematic equations. The artificial variable technique discussed above will find all the occurrences in one pass provided degeneracy has not occurred due to redundancy in the infeasible equations

9.1.6 General Procedure for Finding Infeasibility Causes

Let us summarize a general procedure for finding infeasibility causes.

Step 1 Identify that the model is infeasible or could be infeasible.

Step 2(a) Use CPLEX IIS to find an infeasible set, then go to Step 7

Step 2(b) Identify the places where artificial variables are needed to guarantee feasibility. There are three ways that one can identify these places.

i) Use GAMSCHK using the ADVISORY procedure and have it generate a list of places. Equivalently run NONOPT in IDENTIFY mode, but note this involves a solve and that infeasible models may cause GAMS execution errors. In turn, the equations and variable bounds identified as potentially infeasible are the candidates for artificials

ii) Take the output from one of the PRESOLVE routines in the solvers such as OSL or CPLEX and identify the potential

infeasibilities listed for the addition of artificial variables.

iii) Take the solution output from a solver LST file and pick out the equations and/or variables flagged with INFES. These can also be found by looking at the NONOPT output when NONOPT is run on an infeasible model.

All three steps above identify places where artificial variables need to be added. We believe only the first will identify a set of artificials which if added will result in an "artificially" feasible model which will not need additional artificials added later.

Step 3 Add artificial variables to the identified places. Mechanically the artificial variables are added according to the rules above. Insure that the absolute value of the objective function penalty is at least a thousand times the largest expected marginal.

Step 4 Solve the model.

Step 5 If the model is not feasible, then go back to Step 2(b) and add more artificials. If an optimal solution is found, go on to Step 6.

Step 6 The feasible solution is "artificially feasible" and is not a valid feasible solution if any of the artificial variables are nonzero. Find out if this is the case and if so look for large distorted shadow prices. This may be done manually or through the use of GAMSCHK using the NONOPT procedure. If using NONOPT, make sure the artificial penalties and the MARGFILT option value are set in an appropriate relative manner. Identify those items with large marginals as contributors to the

infeasibility.

Step 7 Investigate the identified items, fix the problem and repeat the above procedure if needed. If you cannot find the infeasibility cause, gain further insight by using the tools discussed in the section on finding the cause for an unrealistic optimal solution in the last part of this chapter.

9.1.7 Infeasibility and Non Linear Programs

Much like the discussion in chapter 8 indicated nonlinear programming problems merit special treatment. This is not because a different technique is to be used, but rather because of the expanded possibility of numerical problems and the importance of starting points. Fundamentally, one discovers the causes of infeasibility in an NLP in the same fashion as one would in an LP. Artificial variables are entered in all constraints that are not feasible at the starting point (note in an LP one looks for equations that are not satisfied when everything is zero however in an NLP one should do this evaluation around the starting point for the variables which have had starting points specified for them and zeros for the other variables). GAMSCHK in NONOPT or ADVISORY mode will identify the constraints which are not feasible taking the starting points into account. Then would one would modify the objective function using either the Big M or Phase I approach as discussed in section 9.4.2 (numerical difficulties often favor use of the Phase I form in NLP cases). One then hunts for the large or nonzero shadow prices as discussed in section 9.4.1.3(see the example problems nlpinfes.gms which is simply an infeasible NLP, nlpinfpi.gms- which is a phase I implementation for that problem- and nlpinfar.gms- a big M implementation for that problem).

However, when dealing with NLP's there is one key difference , namely the possibility of

numerical problems. Once a set of potentially problematic constraints has been identified it is worthwhile evaluating any constraints which contain nonlinear terms to see how much they are infeasible by and whether a change in solution tolerance or a slight loosening of the constraints is desirable. One can also obtain this information by looking at the optimal values of the artificial variables. It can also be worthwhile to investigate whether the constraints with nonlinear terms form a convex constraint set (if this is not true, numerical problems are highly likely). Users may wish to temporarily remove constraints with nonlinear terms and make sure the rest of the problem is feasible, then slowly add back in the nonlinear constraints. Manipulation of solver tolerances and or use of alternative nonlinear solvers may also be in order.

9.1.8 Infeasibility and Mixed Integer Models

Yet another special infeasibility case involves MIPs. Because of integer restrictions models may be infeasible for which an optimal LP solution can be found. Consider the following example (MIPINFES.gms)

```
  1  variables z;
     2  integer variables x1,x2;
3  equations r1,r2,r3,r4;
4  r1.. z=e=3*x1+2*x2;
5  r2..  x1+x2=l=10;
6  r3..  x1=g=4.1;
7  r4..  x2=g=5.1;
8  model infemip /all/
9  solve infemip maximizing z using mip;
```

Solving this problem in OSL causes it to crash while CPLEX concluded normally but said that the problem was integer infeasible. The reason that the problem is integer infeasible is found in the interaction of constraints r2, r3, and r4 along with the integer restrictions on x1 and x2. The only way that solution then use for to x1 and x2 can satisfy r3 and r4 while being integer is to take on

the values of 5 and 6 respectively which makes their sum equal to 11 while r2 causes that to be infeasible. Adding artificial variables and solving causes partial, but not full discovery of the problem. Namely, if we add artificials to r3 and r4 (mipinfar.gms) as follows

```
 1  variables z;
 2  integer variables x1,x2
 3  positive variables a1,a2
 4  equations r1,r2,r3,r4;
 5  r1.. z=e=3*x1+2*x2-1000000*(a1+a2);
 6  r2..  x1+x2=l=10;
 7  r3..  x1+a1=g=4.1;
 8  r4..  x2+a2=g=5.1;
 9  model infemip /all/
10  solve infemip maximizing z using mip;
```

we get a solution as follows:

```
                          LOWER     LEVEL     UPPER    MARGINAL

---- EQU R1                 .         .         .         1.000
---- EQU R2               -INF     10.000    10.000     3.000
---- EQU R3               4.100     5.000     +INF         .
---- EQU R4               5.100     5.100     +INF   -1.000E+6

                          LOWER     LEVEL     UPPER    MARGINAL

---- VAR Z                -INF  -1.000E+5     +INF         .
---- VAR X1                 .       5.000   100.000      EPS
---- VAR X2                 .       5.000   100.000 1.0000E+6
---- VAR A1                 .         .       +INF   -1.000E+6
---- VAR A2                 .       0.100     +INF         .
```

with the artificial variable a2 in it a shadow prices on the r4 constraint as well as a large marginals for the reduced cost of x2. Note that shadow prices are not assigned to the r2 and r3 constraints in conjunction with the r4 constraint cause of the infeasibility problem. To understand why this happens one needs to recognize how an MIP is solved. Generally, MIPs are solved with branch and bound algorithms. During solution that algorithm imposes constraints on the integer variables holding them at integer levels. The constraints r2 and r3 in all likelihood would be superseded by

constraints added during the branch and bound solution process that held x1 greater than or equal to 5 and x2 greater than or equal to 6. Thus, those constraints would have been redundant and non binding. As a consequence shadow prices would not appear on those constraints at the time that the problem was judged infeasible and would not be reported back to the user. Note however in our solution lead to get a large marginal on the x2 variable which indicates that it is being held at a bound in the final optimal solution i.e. apparently during the last branch and bound iteration a lower bound has been imposed on x2 holding it to be greater than or equal to 6. Thus, the marginal information steel provide to least a partial road map pointing to the cause of the difficulty in the problem. Namely, it shows the infeasibility is involved with the interaction of x2 and r4 at least a pointing out to us to investigate those two items.

In general, in the MIP arena the marginal information is less reliable because it's less well defined due to the noncontinuity of the solution region and the way that solvers add constraints as they work. However, adding artificial variables still aids in problem diagnosis providing information on at least a partial subset of the constraints to investigate. Further, the shadow prices in an MIP context become more informative as more and more continuous variables appear in the problem, particularly when the infeasibility is not caused by constraints that are only affecting the pure integer variables..

9.2 Correction of Models Which are Unbounded

LP problems occasionally yield unbounded solutions. One can try to use the tools reviewed in the last chapter to find the problem. However, if the model is still unbounded, one can modify the model and solve it to gain information on the causes. This is done through the imposition of "artificially" large bounds. Linear programming solvers discover unboundedness

when they find a variable which is attractive to make larger, but find that the variable may be increased without limit. This may occur at any stage of the Phase II iterations. Thus, while only one unbounded variable will be reported, there may be numerous other variables which have not been examined and could be unbounded. Unfortunately, the LST file does not generally give enough information to diagnose and fix the cause of the unboundedness and pre-solves rarely find such problems. Commonly, the solution report contains an instance where a particular item is tagged as unbounded (with the marker UNBND) as in Table 8.9, but there will also be other variables marked as nonoptimal (NOPT) which may or may not be unbounded. Finally, note that use of ANALYSIS and correction of all identified problems does not indicate the model will be bounded. Thus, most modelers will occasionally contend with models that are unbounded and will need to discover what is causing that condition.

9.2.1 Solvers and Unbounded Models

Causes of unbounded models are not always easily identified. Solvers may report a particular variable as unbounded when in reality an entirely different variable and interactions between variables imposed within constraints is the real cause. Consider the following example:

$$
\begin{aligned}
\text{Max} \quad & 5X_1 - 3X_2 + 3X_3 \\
& X_1 - X_2 + X_3 \leq 0 \\
& \qquad\quad X_3 \leq 20 \\
& X_1 \,, \quad X_2 \,, \quad X_3 \geq 0
\end{aligned}
$$

Here the unboundedness is caused by the interrelationship between X_1 and X_2. In particular, since X_1 is profit generating but requires X_2 and the net profit contribution of both is positive and since there are no constraints limiting X_1 or X_2, then this model is unbounded. There may be several

potential explanations as to why the unboundedness is present. The profit contributions may not be volume independent and some form of diminishing revenue or increasing cost as sales increase may be omitted. Second, there may be omitted constraints on X_1 or X_2. Third, there may be multiple errors involving the above cases. Runs with OSL, CPLEX, BDMLP and MINOS resulted in the marking of X_2 as the unbounded item. This may or may not be a proper identification of the problem causing mistake. The mistake may be on the X_1 side. Usually unboundedness occurs because of the interaction of multiple variables and constraints, not just the one variable that the solver happens to mark. In a more complex model, potentially a set of 50 variables and constraints could be involved. Thus, we need to find the involved set of variables and equations and then look for the root cause of the unboundedness. How then does one go about discovering this? Again, model modifications may be necessary.

9.2.2 Finding Causes of Unboundedness -- Basic Theory

Linear programming theory shows that infeasible primal problems usually lead to unbounded dual problems. Since the dual of the dual is the primal, we may attack unboundedness by using a "dual" artificial variable approach. Duality theory shows the equivalent action to introducing a dual artificial variable is to enter a large primal upper bound. Thus, we bound the variables to be less than or equal to some very large number like 10^{10}. The consequent model will be bounded, but the solution may have variable values which are quite large. Suppose in the example we bound the X_1 and X_3 variables since they contribute revenue to the objective function.

$$
\begin{array}{lrrrrrrr}
\text{Max} & 5X_1 & - & 3X_2 & + & 3X_3 & & \\
 & X_1 & - & X_2 & + & X_3 & \leq & 0 \\
 & & & & & X_3 & \leq & 20 \\
 & X_1 & & & & & \leq & 1000000000 \\
 & & & & & X_3 & \leq & 1000000000 \\
 & X_1 & , & X_2 & , & X_3 & \geq & 0
\end{array}
$$

Note we are making the problem "artificially" bounded. If it is truly unbounded, then we should expect that the solution will show X_1 and X_2 taking on large values which are far larger than any anticipated "non artificial" value. However, when unboundedness is not present the large upper bound constraints should be redundant with no affect on the solution.

Summarizing, the approach to solving potentially unbounded problems is to add large bounds guaranteeing a bounded solution. When the solution shows the large bounds are binding constraints, then one knows the problem is really unbounded. In such a case the solution information can be used to diagnose the cause of the unboundedness.

We illustrate this by setting up a GAMS model of our above example (called UNBD). When we solve UNBD without the bound constraint active, the solver reports that the model is unbounded placing a flag on X_2 showing that as it enters the solution the model becomes unbounded (Table 9.4, Panel A). If on the other hand, we add a large upper bound on X_1 , we then obtain an optimal solution as in Table 9.4, Panel C. In the Panel C solution, both X_1 and X_2 have taken on quite large values but not X_3. So what? The solution tells us what is wrong in the model through the variable levels. The levels for both X_1 and X_2 are distorted while X_3 is

unaffected. Thus, the modeler would receive signals that the unboundedness involves the interaction of the X_1 and X_2. In turn, we would examine these variables and any binding equations relating them to fix the unboundedness.

The above material indicates a way of finding the cause of unboundedness. Namely, set up the model with large bounds present, solve it and look for distorted levels to find the causal set of variables and equations. One word of caution, this will always identify some of the unboundedness causes, but in the face of a nonunique primal solution caused by degeneracy or alternative optimals may not reveal them all. Thus, multiple applications of the procedure may be needed.

9.2.3 Details on Large Bound Approach to Resolving Unboundedness

The following gives the steps for finding unboundedness causes.

Step 1	Identify the relevant variables for which artificially large bounds need to be added.
Step 2	Add bounds to those variables.
Step 3	Solve the model.
Step 4	Examine the model solution. When variable and equation solution levels are found which are excessively large, identify those as the variables and equations to be examined for the cause of infeasibility.
Step 5	Fix the model and repeat the process if needed.

There are several questions inherent in the above procedure. In particular, which items need bounds? What type of bounds should be entered? How does one find an excessively large level? Each is discussed below.

9.2.3.1 Where Do We Add Large Bounds?

The places where bounds are required can be determined in several ways. One could look at the model solution and just add bounds on the variables marked by the solver as unbounded or nonoptimal. However, while this rather readily points to proper places in the example model, it does not always do such. One approach that can be used is to add bounds to all potentially unbounded variables.

Linear programming models are unbounded when the solver finds the objective function can be improved by altering the value of a variable, but finds that variable is not limited by a constraint. Thus, to identify all potentially unbounded variables then one has to find all variables which contribute to the objective function, but are not directly bounded. Such cases in a maximization context involve

a) nonnegative variables with positive objective coefficients and no upper bound

b) nonpositive variables with negative objective coefficients and no lower bound

c) unrestricted or free variables with positive objective coefficients and no upper bound

d) unrestricted or free variables with negative objective coefficients and no lower bound

These cases identify a larger than necessary set since the restrictions imposed by the constraint set are not considered. However, more complex tests would be needed to factor in those constraints. The ADVISORY and NONOPT procedures in GAMSCHK have been written to create a list of all occurrences of these cases. Use of ADVISORY does not require the model be solved or while NONOPT in IDENTIFY mode only work after a model solve. Note part of

the Table 9.5 output gives an example of this case showing the SELL variables need bounds to guarantee the model will be bounded.

GAMS permits an alternative technique for bounding the problem. Namely, one can go provide a large upper bound on the variable to be maximized or if the problem is minimization problem a large negative lower bound. Attributes of this technique will be discussed in section 9.2.3.4 below.

9.2.3.2 Entering Bounds in GAMS

Once one has found where the bounds need to be added, one still has to address the question of how they should be added and what bound level should be used. The following general rules apply. A bound should be defined for each variable block with the same dimension as the block. Thus, if a variable is defined like SELL(PLANT,TYPE), where the SELL objective function coefficients increase the objective function as the variable value increases, then define a bound.

```
SELL.UP(PLANT,TYPE)=1000000000;
```

On the other hand, if SELL can be negative and the objective function increases as the variable decreases then define,

```
SELL.LO(PLANT,TYPE)= -1000000000.
```

The numerical magnitude of the bound is entirely problem dependent. All that can be said in general is that it should dwarf the largest reasonably anticipated "real" variable level, but should be less than 10^{31} which is considered infinite in GAMS.

The shortcut approach can also be used where one bounds the variable being optimized. For example, if the solve statement minimizes the variable COST then one can be found the

problem simply by entering in the statement

COST.LO = -1000000000;

9.2.3.3 How Do I Find Distorted Levels?

The next question involves finding the distorted levels. The simple aspect of this is that one can simple review the output as in Table 9.4, Panel B and find the levels with large exponents. The more complex aspect is that in a model with thousands of variables and equations this information can be well hidden. The GAMSCHK NONOPT procedure has been written to help in this quest. All items with levels in an optimal solution which are larger in absolute value than 10 to a filter value are output as potential causes of the unboundedness.

Suppose we illustrate using the messed up example from Chapter 8. We also cause the example to be unbounded by taking the model as structured in Table 8.1 then zero the transportation supply usages in line 109 of Table 8.1 and add bounds to the SELL variables as in the above discussion. In this case when we run the solver we get an optimal solution but some of the SELL variables are unrealistically large (Table 9.5). In turn NONOPT yields the output in Table 9.6. NONOPT has named some of the SELL and TRNSPORT variables as items which have excessively large levels and are likely contributors to the unboundedness. In turn one could examine those items and the constraints linking them (possibly using DISPLAYCR, PICTURE, BLOCKPIC or POSTOPT) to find the unboundedness cause.

Usage of the NONOPT for these purposes requires that the LEVELFILT value be set in conjunction with the size of the artificially large bounds. The default LEVELFILT is 6 thus all levels greater in absolute value than 10^6 will be reported. We recommend that the absolute value of the bounds be greater than 10 to LEVELFILT by at least a 1000 (that is why we use such a

large number). One can change this filter by using the GAMSCHK option file and altering LEVELFILT.

9.2.3.4 Comparing the Bounding Techniques

As mentioned in sections 9.2.3.2 and 9.2.3.3 there are actually two unboundedness techniques that can be used. In particular, one can bound the unboundedness by providing bounds on multiple individual variables which contribute to the desirability of the objective function (for example, those that are profitable in a maximization problem) or can simply bound the single variable which is being optimized in the problem. There can be substantial differences in the value the information generated by the techniques, thus a comparison of their characteristics is desirable. The two most distinguishing characteristics of the techniques involve simplicity of use and completeness of information. First, the technique where one simply bounds the variable being optimized is simpler one simply adds one bound statement without having to think through which variables are desirable to the objective function and then add multiple bound statements on those. Second, simplicity has its costs fancy information content in solution may be less under the simpler bound technique. Namely when a unbounded model is solved and there is more than one set of variables causing the unboundedness, the use of the single bound will only reveal one unbounded case at a time. To illustrate this let us return to our example and rather than imposing upper bounds on all the SELL variables let us just impose an bound on the NETINCOME variable(blockbn2.gms). In that case a run of GAMSCHK with NONOPT generates the following output

```
----### THESE VARIABLES MAY BE UNBOUNDED
        Since their levels are so large

  TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)                    level    1246883.
  TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)                    level    7481297.
```

```
   SELL(PLANT2,FANCY)                                          level    1246883.
```

whereas a run of the model use with bounds placed on the profitable for each individual sell variables and yields the following output

```
----### THESE VARIABLES MAY BE UNBOUNDED
         Since their levels are so large

   TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)                      level   0.1000000E+10
   TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)                      level   0.1000000E+10
   TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)                      level   0.4000000E+10
   TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)                      level   0.6000000E+10
   SELL(PLANT2,FUNCT)                                        level   0.1000000E+10
   SELL(PLANT2,FANCY)                                        level   0.1000000E+10
```

comparing the outputs the fact the more extensive bounding technique generates information on more cases. Much like in the case of the IIS above the trade-off is between the more than expensive model examination and bound imposition exercise needed to fully bound all the profitable variables in the model vs. the simple, one bound statement needed approach which might take multiple solution passes to fully identify all the problems in the model.

9.2.4 General Procedure for Finding Unboundedness Causes

We then now outline our procedure for finding the causes of unboundedness.

Step 1 Identify that the model is unbounded or could be unbounded

Step 2 Identify where to add artificially large bounds. There are four ways to identify these places

- a Manually identify all variables which contribute to the objective function and are not bounded
- b Use the GAMSCHK ADVISORY or NONOPT-IDENTIFY procedure. In turn, the equations and variables listed as potentially unbounded are the candidates.

c Pick out the equations and/or variables flagged with UNBND and NOPT in the solver LST file. These can also be found by running NONOPT on an unbounded model without the IDENTIFY keyword.

d. Add an upper bound to the variable being maximized or a lower bound to the variable being minimized if one wants to take the simple one bound approach.

All four ways identify places where artificially large bounds need to be added. We believe only the first and last will guarantee that the subsequent model will be artificially bounded.

Step 4 Add the artificially large bounds. Mechanically the bounds are added according to the rules above. Insure that the absolute bound value is at least a thousand times the largest absolute expected variable level.

Step 5 Solve the model.

Step 6 If the model is still unbounded go back to Step 2 and add more bounds. If an optimal solution is found, pass on to Step 7.

Step 7 The feasible solution is “artificially bounded” and not a valid bounded solution if any of the artificially large bounds are binding. Find out if this is the case and if so look for large distorted variable and equation levels. This may be done manually through the use of GAMSCHK using the NONOPT procedure. If using NONOPT make sure the artificially large bounds and the LEVELFILT option value are set in an appropriate relative

manner. Identify those items with large levels as contributors to the unboundedness.

Step 8 Investigate those items and any binding constraints which interrelate them. Fix the problem and repeat the above procedure if needed. If you cannot find the unboundedness cause, then use the tools discussed in the section on finding the cause for unrealistic optimal solutions.

9.2.5 NLPs, MIPs and Unboundedness

In the sections above we presented material that indicated that infeasibility diagnosis involves a somewhat different set of issues dealing with mixed integer or nonlinear programs. When dealing with the unbounded case there are really not fundamentally different approaches to use with dealing with either mixed integer programs or nonlinear programs. One proceeds adds above in either case. However in the nonlinear programming case one also has to consider has two additional issues: objective function form and solver numerical properties. In terms of objective function form, nonlinear programming theory requires a concave objective function for the attainment of global optimality in maximization problems and a convex objective function in the case of minimization problems. The if a nonlinear programming model is judged unbounded, then one should investigate the objective function convexity/concavity characteristics. Nonlinear programming books such as Hadley or Bazarra and Shetty cover this issue.

Second when a nonlinear programming model is judged unbounded and one can be running into numerical problems. In particular, issues such scaling, starting points, tolerances and other numerical issues can be a the problem If the nonlinear model is malfunctioning one might send the problem to CONOPT to see if it is rejected for scaling and otherwise investigate the

starting point, scaling and numerical tolerances. The bounding technique above has been shown in the authors work. to be useful but on occasions has been subject to numerical problems which needed to be resolved before proceeding.

9.3 Duality and A Single Artificial

The alternative approach to unboundedness discovery were one simply enters a single bound on the objective function coupled with duality theory suggests an alternative artificial variable approach which some may wish to employee. Namely, it is possible the add just a single artificial variable which is in that the dual of the single bound on the overall objective function. This artificial variable is added in the following manner. Add the artificial variable into the objective function with either a big M penalty or in the phase 1 case just a plus one coefficient. The artificial also enter the constraints which are not feasible at the all zero (or starting point in NLPs) solution. It will have a negative one coefficient in less than or equal to constraints which have a negative right hand side; a positive one coefficient in greater than or equal to constraints which have a positive right hand side; and a coefficient of one with the sign the same as the sign of the right hand side in equality constraints which have a nonzero right hand. Such an artificial is added in the example blockar3.gms. There an application of NONOPT in GAMSCHK shows the solution contains the message

```
----### THESE EQUATIONS MAY PARTIALLY CAUSE INFEASIBLE MODEL
          Since their marginals are so large
   RESOUREQ(PLANT1,LABOR)                  marg    0.1000000E+08
```

along an identification of eight of the MAKE variables as also having large marginals. On the other hand running the model with the full set of artificials present in each of the possibly infeasible equations yields.

```
----### THESE EQUATIONS MAY PARTIALLY CAUSE INFEASIBLE MODEL
         Since their marginals are so large
   RESOUREQ(PLANT1,SMLLATHE)              marg    0.1000000E+10
   RESOUREQ(PLANT1,LRGLATHE)              marg    0.1000000E+10
   RESOUREQ(PLANT1,CARVER)                marg    0.1000000E+10
   RESOUREQ(PLANT1,LABOR)                 marg    0.1000000E+10
   RESOUREQ(PLANT1,TOP)                   marg    0.1000000E+10
   RESOUREQ(PLANT2,SMLLATHE)              marg    0.1000000E+10
   RESOUREQ(PLANT2,LRGLATHE)              marg    0.1000000E+10
   RESOUREQ(PLANT2,CARVER)                marg    0.1000000E+10
   RESOUREQ(PLANT2,LABOR)                 marg    0.1000000E+10
```

along with identification of fourteen of the MAKE variables as also having large marginals. Thus, when using the single composite artificial one finds the most infeasible case, not all the infeasible cases. Consequently, one would have to interactively solve the model, fix any cases found, then resolve the model fix any additional cases found and so until all the infeasibility cases had been resolved.

9.4 Unrealistic Optimal Solutions

Unfortunately, an investigation of some optimal solutions quickly shows that the solution is ridiculous. The presence of an optimal solution means a problem has a mathematically consistent optimum. However, mathematically consistency does not necessarily imply that the solution will be realistic in terms of the entity being modeled. Usually unrealistic solutions are caused by improper specifications within a model. For example, one may inadvertently omit constraints, leave out certain coefficients, improperly calculate coefficients and/or make algebraic errors in the GAMS equation specifications. Ultimately such errors are found by examining the empirical model comparing the empirical structure and coefficients with expectations about a proper empirical model. In turn, one usually finds and corrects errors. This section covers ways to search for such errors. However, we will not be any means exhaustive as error finding is always problem specific.

Improper model solutions exhibit incorrect valuation and/or allocation results. Valuation difficulties arise in the reduced cost and shadow price information. Valuation information is determined by setting the reduced costs to zero for basic variables. Allocation difficulties arise when the slack or decision variable values are unrealistic. The values of these items are formed through the satisfaction of the constraints. Thus, to find unrealistic solutions, one investigates either the valuation information associated with the reduced costs or the allocation calculations inherent in the primal constraints. Two techniques are presented below, one for the investigation of reduced costs, which we call "budgeting"; and another for the reconstruction of the constraint calculations, which we call "row summing."

9.4.1 Budgeting

A model analysis technique useful when dealing with unrealistic optimal solutions involves reconstruction of the reduced costs. We will call this procedure budgeting and GAMSCHK supports it through the POSTOPT procedure when variables are selected. Budgeting can be used for both linear and nonlinear problems. We will cover the theory for an LP but the principles are the same in the NLP arena.

9.4.1.1 Theoretical Background for Budgeting

In any linear programming problem the optimal solution arises when the reduced costs of the basic variables are equal to zero and the reduced costs of the non-basic variables are greater than or equal to zero. The formula for the reduced cost of all variables is:

$$\sum_i u_i \, a_{ij} - c_j \geq 0 \qquad \text{for all } j$$

In this equation, u_i is the shadow price from the solution, a_{ij} is the amount of the ith resource used in producing the jth commodity and, c_j the direct objective function contribution of the jth

variable. This equation shows that the sum of the shadow prices times the resource usage is greater than or equal to the objective function contribution for each and every variable. Further in a linear programming solution the reduced cost of the basic variables is equal to zero, so for basic variables the equation above reduces to:

$$\sum_i u_i \, a_{ij} - c_j = 0 \qquad \text{for } j \text{ basic}$$

or in matrix terms

$$UB - C_B = 0$$

where the matrix B gives the elements of the constraint matrix for the basic columns, and C_B is the vector of objective function coefficients for the basic variable values. In turn this equation can be rewritten as:

$$UB = C_B$$

and given that B is invertible one can solve for U

$$U = C_B B^{-1}$$

This equation shows how shadow price values are computed. Namely, the shadow price values are a function of the coefficients of the variables in the linear programming model which use those resources. What this means is that the shadow prices take on their values from the basic variables. Thus, in order to examine incorrect evaluation information one needs to examine the reduced costs of the basic variables, we do this through the budgeting technique.

9.4.1.2 The Budgeting Technique

The budgeting technique is simply an explosion of the equation in

$$\sum_i u_i \, a_{ij} - c_j \geq 0 \qquad \text{for all } j.$$

What we do in order to budget is create a table that is formatted as follows:

Budget for Variable j

Equation Name	a_{ij}	Shadow Price	$u_i a_{ij}$
i=1	a_{1j}	u_1	$u_1 a_{1j}$
i=2	a_{2j}	u_2	$u_2 a_{2j}$
⋮	⋮	⋮	⋮
i=m	a_{mj}	u_m	$u_m a_{mj}$
Sum	--	--	$\Sigma_i u_i a_{ij}$
Objective	--	--	$-c_j$
Reduced Cost	--	--	$\Sigma_i u_i c_{ij} - c_j$

Here we list each equation in which the variable to be budgeted has a coefficient, as well as the shadow prices for those equations. In turn, we multiply them together, sum across the equations, subtract off the objective function coefficient c_j and then form the reduced cost. Collectively this should exactly reproduce the calculations in the model that determine the variable to enter during simplex iterations and the calculations inherent in the formation of the shadow prices. The procedure also yields the optimal reduced costs which give costs of forcing a nonbasic variables in the solution. The utility of this information is that one may examine

1. why particular variables are in the solution that one feels should not be;
2. why particular variables are held at a zero solution level one feels they should be nonzero; and
3. why resources are worth particular amounts in terms of their optimal marginal or shadow price values.

This is probably best illustrated by the solution example, thus consider the model shown in Table 9.7.

This model contains variables for buying miscellaneous inputs, selling corn, soybeans, and

pork; and producing corn, soybeans, and hogs. The objective is maximized subject to equations constraining land, and supply-demand balances on pork, soybeans, corn, and miscellaneous inputs. The miscellaneous equation item is specified in dollars and therefore enters the objective function at a per unit cost of $1 while supplying a dollar's worth of miscellaneous inputs. Corn is sold for $2.50 per unit, soybeans for $6 per unit, and pork for $.50 per unit. Corn production incurs $75 in direct production costs and $125 in miscellaneous inputs while using one acre of land and yielding 120 bushels of corn. Soybean production costs $50 in direct production costs and another $50 in miscellaneous inputs while using an acre of land and yielding 50 bushels of soybeans. Hog production has no direct costs, uses $20 in miscellaneous inputs, and requires 20 bushels of corn. An unrealistically large yield in the hog activity has been entered (1000 pounds per hog). This example "error" will be sought by the budgeting technique.

The optimal solution to this model is shown in Table 9.8. The optimal value of the objective function is $1,608,000. This solution includes several symptoms that there is something wrong. For example, 3,600,000 pounds of pork are sold, the reduced cost on selling soybeans is $49.6 a bushel, the shadow price on land is $2,680 and the shadow price of corn is $24 a bushel. Budgeting investigates the shadow prices and reduced costs in an effort to discover model misspecifications.

Budgeting involves construction of an extensive version of the reduced cost calculations which in turn are examined for plausibility. The variable budgeted first could be chosen because: a) it is nonbasic when intuition suggests it should be basic; b) it has an unrealistically high reduced cost; or c) it uses a resource which appears to be improperly valued. In the example, suppose we budget soybean sales because of its high reduced cost. To budget, write a row for each nonzero

coefficient (a_{ij}) under the chosen variable, with associated shadow prices (u_i) and $a_{ij}u_i$ the product, then finally subtracting cost. The budget generated by GAMSCHK POSTOPT for soybean sales is shown in Table 9.9, Panel A (Note in that table that the objective row does not always appear as the last entry, rather its placement depends on the order of the GAMS equation specifications).

Mechanically the budget examines the value of resource usage in the equations in which the variable has nonzero coefficients. In the case of the soybean sales variable, there is a non-zero coefficient in the soybean production constraint along with an objective function coefficient. The shadow price on soybean production of $55.60 while the commodity sells for $6.00. Thus, the direct revenue to sales is $6.00 while the opportunity cost of the soybeans used is $55.60. In this case the $49.60 reduced cost arises since the bushel of soybeans is only sold for $6.00 while the $55.60 opportunity cost means the firm would lose $49.60 .bushel of soybeans sold in. Thus, the reason for the high reduced cost for soybean sales is the internal worth (the shadow price) of soybeans. Empirically this value is about 10 times what it should be. Thus, we need to investigate the distorted soybean shadow price.

The cause of a distorted shadow price for a resource always lies in the basic (or super basic) variables that use or produce that resource. In this case, in the basic variable involved with soybeans is soybean production. Thus, we budget the soybean production variable. That variable has non-zero coefficients in the land, soybean balance and miscellaneous input balance constraints. The budget (Table 9.9, Panel C) shows that one acre of soybeans uses $2,680 worth of land and yields 50 bushels of soybean, each valued at the opportunity for shadow price of $55.60. Also, 50 units of miscellaneous inputs are used which, valued at the shadow price of $1, cost $50. Summing these terms, the marginal contribution of soybean production, ignoring its direct costs,

is \$50. Its direct cost (c_j=50) is then subtracted yielding a \$0 reduced cost for this basic variable. These data show that the \$55.60 soybean value arises because the imputed value of soybeans needs to balance off against the high imputed value of land - the \$2,680 shadow price. The question then becomes why is land this valuable.

Again, shadow prices are derived from basic variables which use those resources. Thus, the high land shadow price must arise from the reduced costs of some other basic variable which utilizes land. The only other land using basic variable is corn production. We then budget the corn production variable (Table 9.9, Panel B). Note that while one acre of corn production uses \$2,680 of land, it receives \$2,880 from the value of the corn sold. Thus, the reason for the \$2,680 cost of land is the \$2,880 value of the corn. Institutional knowledge indicates the 120 bushels per acre corn yield is reasonable, but the \$24 corn shadow price is not.

Thus, the question becomes, "Why is the corn shadow price so high?" Again, this is generated by a basic variable which utilizes corn. The only basic corn consuming variable is hog production. The budget for hog production shown in Table 9.9, Panel D. These computations show that zero reduced cost for this activity requires that 20 bushels of corn be valued at \$24/unit. The cause of the \$24 a bushel value for corn is an unrealistic value of pork produced (\$500). The erroneous 1000 lb. coefficient for pork production per hog would then be discovered. A revised value of the pork yield per hog would alter the model, making the solution more realistic.

9.4.1.3 Budget Summary

The above procedure gives the general indication of how the budgeting technique is used. Namely, one finds an unreasonable shadow price or reduced cost and then examines the model

data in to see why that result might occur. In linear programming models the shadow prices are always set so the difference between the marginal revenue for producing the product and its marginal direct objective function cost is equated by the shadow prices. For example, in the soybean production activity the miscellaneous input costs in objective function coefficient have fixed prices so the only place to attribute the differences between those items is in the soybean shadow price in the land shadow price. Since the land shadow price took on a large value, then in a rigorous all of that value was allocated into the soybean price leading to a very high soybean opportunity cost. Similarly, when the reduced cost were formed for the hog production variable in the high yield of pork times the price yielded a high level of marginal revenue that was allocated to the only factor of production which did not have a fixed shadow price i.e., corn.

The general procedure then is to go through this residual accounting, possibly in a deductive step wise situation, until one finds the unrealistically large revenue and cost that is causing the shadow prices to be distorted. In turn, one then repairs the model and solves it again and if needed repeats the process.

The budgeting technique is useful in a number of settings. Through its use, one may examine the reduced costs of nonbasic variables to see what is keeping them out of solution. The soybean sales variable budget provides such an example. Budgeting may also be used to investigate the magnitude of shadow prices. Shadow prices arise from a residual accounting framework where, after the fixed revenues and costs are considered, the residual income is attributed to the unpriced resources. The example again shows such a case. Finally, budgeting can be used to deal with infeasible solutions from phase 1 of a two-phase simplex algorithm or with solutions containing artificial variables to trace the infeasibility cause.

9.4.2 Row Summing

Model solutions also may be analyzed by examining model constraint activity. In the budgeting example problem, one could have examined the reasons for the sale of 3.6 million pounds of pork. This can be done through a procedure we call row summing. Row summing is done by GAMSCHK under the POSTOPT procedure when equations are selected.

9.4.2.1 Theory Behind the Row Summing Techniques

In any linear or nonlinear programming problem, the constraints generally play a large role in determining the values of solution variables. The constraints may do many things, for example, a) providing upper limits on resource usage by the variables; b) lower limits on the variables; or c) balances between variables (when positive coefficients times a subset of variables are related to negative coefficients times another subset of variables). In all of these cases, when one finds an unrealistic solution with for example variables with values much larger than expected, one can investigate those variables by looking at constraint activity.

Basically the theory of row summing can be developed as follows. In a linear programming problem, (this presentation is done in a linear programming context but also applies to the nonlinear programming case) the model is subject to constraints of the form $AX + S=b$ where AX is the constraint matrix times the decision variables X, S is a slack variable and b is the right-hand side. The parameters in A can either be positive or negative. In a row sum we examine the components of AX. Thus, we list all variables with nonzero coefficients in the equations, the optimum X values, and their product then we sum those, subtract off the right hand side and compute the slack. The layout is as follows:

Variable Names	Coefficients	Variable Levels	Product

X_1	a_{i1}	X_1^*	$a_{i1}X_1^*$
X_2	a_{i2}	X_2^*	$a_{i2}X_2^*$
⋮	⋮	⋮	⋮
X_n	a_n	X_n^*	a_{in}Xn*
Sum	--	–	$\Sigma_j a_{ij}X_j^*$
RHS	--	–	b
Slack	--	–	b-Σj$a_{ij}X_j^*$

This shows how the activity in the row balances out and in particular how usage by certain variables times their coefficients is either complementary or competitive with other variables. Often this shows that the large usage of resources by one variable is complemented by a large supply of resources by another. It may also show that the resource usage by certain variables is so large there are no resources left for other variables or that the resource usage coefficients of particular variable may be larger than desired. Regardless, by considering the activity in each of the equations one can often find model problems.

9.4.2.2 Example

Again, this is best illustrated through example. Table 9.10 shows a slightly different, but related, example. That model is structurally the same as that in Table 9.7, but the pork production coefficient has been altered to -150, while the corn yield per unit has been changed to an incorrect value of 1200 -- the new error. We have also introduced a RHS of 20 on the corn balance equation. The resultant solution is shown in Table 9.11. The optimal value of the objective function is $1,860,055. Here 5.4 million pounds of pork are sold which would probably be judged to be unrealistically high. Further, there are 36,001 hogs on the farm.

The use of row summing will be illustrated through the GAMSCHK POSTOPT procedure beginning with a row sum examination of the pork sales constraint to see if 5.4 million lbs. of sales is reasonable (Table 9.12, Panel A).

The pork constraint balances the variables sell pork and hog production. The sell pork variable uses one pound of pork per unit, while the hog production variable yields 150 pounds of pork per unit. The second column of Table 9.12 contains the optimal variable values (X*). In the third column, the product of the variable value (X*) and its a_{ij} appears. The products are summed to give total endogenous use which in this case equals zero. We then enter the right-hand side and subtract it to determine the value of the slack variable. Now suppose we examine the resultant table. Given institutional knowledge, one would conclude the error has not yet been found as a yield of150 lbs. of pork per hog is reasonable, and all pork produced is sold. However, one would wonder if a production level of 36,001 hogs is reasonable. The next step is to examine the resources used by hog production. For illustrative purposes, we begin with the miscellaneous input supply-demand balance. The row sum for this constraint is shown in Panel B.

There are four entries in the constraint. The row sum does not reveal anything terribly unrealistic except the large amount of activity from the hog production variable causes the model to match this with a high the level of miscellaneous input but per unit rates appear appropriate. The basic question is yet to be resolved.

We next investigate the corn supply-demand balance. The row sum computations for this constraint are shown in Panel C. In this case, the constraint has a non-zero right-hand side; thus, the endogenous sum is 20 and the computed the slack variable is zero. We find the 36,001 hogs require 720,020 bushels of corn, and the reason they are able to obtain all this corn is because of the inaccurate yield on the corn production variable. The modeler would then correct the yield on the corn production variable.

9.4.2.3 Row Summing Summary

The above example illustrates the principles behind using row summing to find flaws. In particular row summing allows one to examine how the resources model and constraints are allocated in the optimum solution. One identifies an item with an unrealistically high solution value, and then row sums the constraints in which that item appears to discover the problem. Row summing can be used to discover incorrect coefficient values or coefficient placement errors. For example, suppose that the corn yield was inadvertently entered in the soybean row; then one might have discovered a solution in which soybeans are sold but no soybeans are produced. A row sum would quickly determine the source of the soybeans and indicate the error. Row summing can also be applied to discover the causes of large values for slack or surplus variables.

9.6 GAMSCHK, Post Optimality Calculations and NLPs

As covered in chapters 8 and 9 errors special considerations are involved when doing GAMSCHK supported post optimality on nonlinear programming problems. In particular two items are worthy of mention. First, because of the local Taylor's series expansion nature of the nonlinear coefficient terms as reported by GAMS to GAMSCHK, the a_{ij} and c_j information may not be strictly accurate in the optimum solution hand GAMSCHK may not properly balance out the reduced cost and equation activity. An accurate accounting of this information can only be obtained when one has resolved the model around the current solution. This cause GAMS to report out the current Taylor's series expansion. Second, one must remember that the coefficients in the nonlinear terms are local values of the nonlinear terms, not global values. Thus one must be careful in interpreting the GAMSCHK output. As the reminder GAMSCHK marks these terms with asterisks. For example consider the small model

```
1  variables z;
2  positive variables x1,x2;
```

```
3  equations obj,r1;
4  obj.. z=e= -3*(x1-2)*(x1-2) -2*(x2-4)*(x2-4);
5  r1..     x1+x2=l=2;
6  model nlpe /all/
7  option nlp=gamschk;
8  solve nlpe maximizing z using nlp
9  solve nlpe maximizing z using nlp
```

The usage of POSTOPT in GAMSCHK during the first solve statement in line 8 yields the following output:

```
##  X1
EQN                                Aij       Ui        Aij*Ui
OBJ                         *** -12.000    1.0000     -12.000
R1                               1.0000    9.6000      9.6000
REDUCED COST EXCLUDING BOUNDS                          -2.4000
Accounting Error -MIP or NLP?                           2.4000
TRUE REDUCED COST                                       0.0000
```

Therein the user receives an indication that the objective function term for X1 is nonlinear. Also an accounting error arises because the program cannot compute the level of reduced costs that optimizer output says should be attainable. This occurs because the nonlinear terms have not been expanded about the current solution. One the other hand, the POSTOPT output from the solve in line generates the output as follows:

```
##  X1
EQN                                 Aij          Ui         Aij*Ui
OBJ                          *** -9.6000      1.0000     -9.6000
R1                                1.0000      9.6000      9.6000
TRUE REDUCED COST                                         0.00000
```

Where in the output air has laminated because now the parameter for acts one in the objective function has been expanded grams true value seats the four cities in reduced cost can be matched exactly. The same sort of phenomenon happens terms of the row sum information. In particular, the row sum information may in solve some terms from the Taylor's series expansion that are

added owners constants to the right hand side thus the row sum information from the solve the law and 9 appears as follows:

```
##  OBJ

  VAR                      Aij         Xj       Aij*Xj
Z                        1.0000     -19.200     -19.200
X1                  *** -9.6000     0.40000     -3.8400
X2                  *** -9.6000      1.6000     -15.360
Other Nonlinear                                 -38.400
   =E=                                            =E=
RHS COEFF                                      0.00000E+00
```

The term in the above output labeled “other nonlinear” is a calculation of what the constant terms in the Taylor’s series expansion has to be in order that this equation is satisfied and consistent with the answer given back by the solver. This term will not disappear even when the Taylor’s series expansion is updated because that expansion will in general generate constants. However , the nonlinear terms will be more accurate when the Taylor’s series expansion has been constructed around the most recent solution point. In the first solve in line 8 the value of this other nonlinear constant term was 49.6.

9.7 GAMSCHK, Post Optimality Calculations and MIPs

Another concern in the use of GAMSCHK involves post optimality analysis of mixed integer programming problems. While the row summing information is as fully reliable Cassidy is in the linear programming case, the POSTOPT budgeting results of require some caution in their use and some clarification as to the basic nature. T and he real problem with the budgeting information lies in the less well-defined nature of the duality information in a mixed integer context. In particular, the shadow prices and marginals that are yielded during a mixed integer programming solution may be partially defined by constraints which do not appear in the original

model, but which were temporarily added during the branch and bound solution process. In particular the branch and bound solution process may yield reduced cost resolve reduced costs and were have results which are not consistent with the traditional complementary slackness principles that linear programming solution exhibit. Consider the following example

```
 1  variables z;
 2  integer variables x1,x2;
 3  equations r1,r2,r3,r4;
 4  r1.. z=e=3*x1+12*x2;
 5  r2..  -x1+2*x2=l=20;
 6  r3..  x1=g=4.1;
 7  r4..  x1+x2=l=25.1;
 8  model mip /all/
 9  option mip=gamschk;
10  solve mip maximizing z using mip;
```

When this problem is solved budget for X1 appears is appears as follows

```
##  X1
  SOLUTION VALUE                                10.0000
  UPPER BOUND                                   100.000

  EQN                          Aij         Ui       Aij*Ui
R1                          -3.0000      1.0000     -3.0000
R2                          -1.0000     0.00000E+00 0.00000E+00
R3                           1.0000     0.00000E+00 0.00000E+00
R4                           1.0000     0.00000E+00 0.00000E+00
TRUE REDUCED COST                                   -3.0000
```

This solution nominally exhibits a violation of complementary slackness conditions. In particular, the solution value for X1 is 10 which is between the lower and upper bounds of 0 and 100 so X1 is nonzero but between its bounds. Classical linear programming complementary slackness theory would say that the reduced costs for this variable should be zero. However this variable as a nonzero reduced cost of -3. This occurs because during the branch and bound algorithm apparently a bound is imposed on X1 to bring it to the integer value of ten. Furthermore, the reduced costs that are reported to the list file reflects the presence of that bound even know it is

not present in the original problem. Thus, when one solves mixed integer programming problems, one has to be aware of the characteristics the solution algorithm as they will impact the character of the budgeting that is generated when GAMSCHK runs.

9.8 Post Optimality Computations Without GAMSCHK

One may hand generate versions of the above tests without use of GAMSCHK by using GAMS calculations. For example, in a resource allocation model the lines below could be used to generate a budget.

```
 PARAMETER BUDGET(J,*,*)  BUDGET OF COLUMN J;
 BUDGET(J,I,"AIJ")=A(I,J);
 BUDGET(J,I,"SHADOWPRIC")=CONSTRAINT.M(I);
 BUDGET(J,I,"PRODUCT")=A(I,J)*CONSTRAINT.M(I);
BUDGET(J,"SUMINDIRCT","PRODUCT")=
             SUM(I,BUDGET(J,I,"PRODUCT"));
 BUDGET(J,"OBJECTIVE","PRODUCT")= C(J);
 BUDGET(J,"REDUCECOST,"PRODUCT")=
       BUDGET(J,"SUMINDIRCT","PRODUCT")-
 BUDGET(J,"OBJECTIVE","PRODUCT");
```

However, such coding is specific to a model structure and must be changed to accommodate each model to be analyzed. We feel use of GAMSCHK is easier.

Table 9.1. Infeasible Model Example

Panel A: GAMS Setup of Infeasible Example

```
    1  variable           objmax
    2                     objmin;
    3  positive variables x1
    4                     x2
    5                     A
    6  equations          obj
    7                     obj2
    8                     r1
    9                     r2
   10                     r3;
   11  obj.. objmax =e= 50*x1 +50*x2  -1000000000 * A;
   12  obj2.. objmin =e=  A;
   13  r1..                x1 +   x2                    =L=  50;
   14  r2..             50*x1 +   x2                    =L=  65;
   15  r3..                x1         +             A   =G=  20;
   16  model infe /obj,r1,r2,r3/
   17  model infe2 /obj2,r1,r2,r3/
   18  solve infe using lp maximizing objmax;
   19  solve infe2 using lp minimizing objmin;
```

Panel B: Big M solution (Model INFE)

```
                         LOWER     LEVEL     UPPER    MARGINAL
---- EQU OBJ               .         .         .        1.000
---- EQU R1              -INF      1.300    50.000       .
---- EQU R2              -INF     65.000    65.000 2.0000E+7
---- EQU R3             20.000    20.000     +INF  -1.000E+9

                         LOWER     LEVEL     UPPER    MARGINAL
---- VAR OBJMAX          -INF  -1.87E+10     +INF       .
---- VAR X1                .       1.300     +INF       .
---- VAR X2                .         .       +INF  -2.000E+7
---- VAR A                 .      18.700     +INF       .
```

Panel C: Phase I Solution (Model INFE2)

```
                         LOWER     LEVEL     UPPER    MARGINAL
---- EQU OBJ2              .         .         .        1.000
---- EQU R1              -INF      1.300    50.000       .
---- EQU R2              -INF     65.000    65.000    -0.020
---- EQU R3             20.000    20.000     +INF      1.000

                         LOWER     LEVEL     UPPER    MARGINAL
---- VAR OBJMIN          -INF     18.700     +INF       .
---- VAR X1                .       1.300     +INF       .
---- VAR X2                .         .       +INF      0.020
---- VAR A                 .      18.700     +INF       .
```

Table 9.2 List of All Possible Infeasible or Unbounded Conditions from GAMSCHK Advisory Procedure.

```
----#### Executing ADVISORY

 ----### THESE VARIABLES ARE POTENTIALLY UNBOUNDED

     TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)
     TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FUNCT,PLANT2,PLANT1)
     TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)
     TRNSPORT(CHAIRS,FANCY,PLANT2,PLANT1)
     SELL(PLANT1,FUNCT)
     SELL(PLANT1,FANCY)
     SELL(PLANT2,FUNCT)
     SELL(PLANT2,FANCY)

 ----### THESE EQUATIONS ARE POTENTIALLY INFEASIBLE

     RESOUREQ(PLANT1,SMLLATHE)
     RESOUREQ(PLANT1,LRGLATHE)
     RESOUREQ(PLANT1,CARVER)
     RESOUREQ(PLANT1,LABOR)
     RESOUREQ(PLANT1,TOP)
     RESOUREQ(PLANT2,SMLLATHE)
     RESOUREQ(PLANT2,LRGLATHE)
     RESOUREQ(PLANT2,CARVER)
     RESOUREQ(PLANT2,LABOR)
```

Table 9.3 Output from Infeasible Model with Artificials

Panel A GAMS Solution Output

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE   -9480000000000.0210

---- EQU RESOUREQ    RESOURCES AVAILABLE
                   LOWER     LEVEL     UPPER    MARGINAL
PLANT1.SMLLATHE     -INF  -1100.000 -1100.000 1.0000E+9
PLANT1.LRGLATHE     -INF   -880.000  -880.000 1.0000E+9
PLANT1.CARVER       -INF   -500.000  -500.000 1.0000E+9
PLANT1.LABOR        -INF  -1750.000 -1750.000 1.0000E+9
PLANT1.TOP          -INF   -500.000  -500.000 1.0000E+9
PLANT2.SMLLATHE     -INF  -1400.000 -1400.000 1.0000E+9
PLANT2.LRGLATHE     -INF   -900.000  -900.000 1.0000E+9
PLANT2.CARVER       -INF  -1200.000 -1200.000 1.0000E+9
PLANT2.LABOR        -INF  -1250.000 -1250.000 1.0000E+9
PLANT2.TOP          -INF         .         .         .

---- EQU PLANTPROD   PRODUCT BALANCE FOR A PLANT
                      LOWER     LEVEL     UPPER    MARGINAL
PLANT1.TABLES.FUNCT     -INF       .         .         .
PLANT1.TABLES.FANCY     -INF       .         .         .
PLANT1.CHAIRS.FUNCT     -INF       .         .      100.000
PLANT1.CHAIRS.FANCY     -INF       .         .      133.667
PLANT2.TABLES.FUNCT     -INF       .         .       14.000
PLANT2.TABLES.FANCY     -INF       .         .       18.000
PLANT2.CHAIRS.FUNCT     -INF       .         .      102.750
PLANT2.CHAIRS.FANCY     -INF       .         .      138.667

---- VAR MAKE        NUMBER OF ITEMS MADE
                               LOWER     LEVEL     UPPER    MARGINAL
PLANT1.TABLES.NORMAL.FUNCT       .         .        +INF  -4.000E+9
PLANT1.TABLES.NORMAL.FANCY       .         .        +INF  -6.000E+9
PLANT1.CHAIRS.NORMAL.FUNCT       .         .        +INF  -2.70E+10
PLANT1.CHAIRS.NORMAL.FANCY       .         .        +INF  -3.70E+10
PLANT1.CHAIRS.MAXSML.FUNCT       .         .        +INF  -3.00E+10
PLANT1.CHAIRS.MAXSML.FANCY       .         .        +INF  -3.80E+10
PLANT1.CHAIRS.MAXLRG.FUNCT       .         .        +INF  -3.00E+10
PLANT1.CHAIRS.MAXLRG.FANCY       .         .        +INF  -3.80E+10
PLANT2.CHAIRS.NORMAL.FUNCT       .         .        +INF  -2.70E+10
PLANT2.CHAIRS.NORMAL.FANCY       .         .        +INF  -3.70E+10
PLANT2.CHAIRS.MAXSML.FUNCT       .         .        +INF  -3.00E+10
PLANT2.CHAIRS.MAXSML.FANCY       .         .        +INF  -3.80E+10
PLANT2.CHAIRS.MAXLRG.FUNCT       .         .        +INF  -3.00E+10
PLANT2.CHAIRS.MAXLRG.FANCY       .         .        +INF  -3.80E+10

---- VAR TRNSPORT    NUMBER OF ITEMS TRANSPORTED
                               LOWER     LEVEL     UPPER    MARGINAL
TABLES.FUNCT.PLANT1.PLANT2       .         .        +INF       .
TABLES.FANCY.PLANT1.PLANT2       .         .        +INF       .
CHAIRS.FUNCT.PLANT1.PLANT2       .         .        +INF     -2.250
CHAIRS.FUNCT.PLANT2.PLANT1       .         .        +INF     -7.750
CHAIRS.FANCY.PLANT1.PLANT2       .         .        +INF       .
CHAIRS.FANCY.PLANT2.PLANT1       .         .        +INF    -10.000

---- VAR SELL        NUMBER OF SETS SOLD
                  LOWER     LEVEL     UPPER    MARGINAL
PLANT1.FUNCT        .         .        +INF       .
PLANT1.FANCY        .         .        +INF     -2.000
PLANT2.FUNCT        .         .        +INF       .
PLANT2.FANCY        .         .        +INF       .

---- VAR ARTRESOUR   ARTIFICIAL FOR RESOUREQ ROW
                   LOWER     LEVEL     UPPER    MARGINAL
```

Table 9.3 (continued)

```
PLANT1.SMLLATHE       .     1100.000     +INF       .
PLANT1.LRGLATHE       .      880.000     +INF       .
PLANT1.CARVER         .      500.000     +INF       .
```

```
PLANT1.LABOR          .     1750.000     +INF       .
PLANT1.TOP            .      500.000     +INF       .
PLANT2.SMLLATHE       .     1400.000     +INF       .
PLANT2.LRGLATHE       .      900.000     +INF       .
PLANT2.CARVER         .     1200.000     +INF       .
PLANT2.LABOR          .     1250.000     +INF       .
PLANT2.TOP            .         .        +INF  -1.000E+9

                        LOWER     LEVEL     UPPER    MARGINAL
---- VAR NETINCOME       -INF  -9.48E+12     +INF       .
```

Panel B NONOPT Output

```
 ----### THESE VARIABLE BOUNDS MAY CAUSE INFEASIBLE
        Since their marginals are so large

   MAKE(PLANT1,TABLES,NORMAL,FUNCT)                        marg   -0.4000000E+10
   MAKE(PLANT1,TABLES,NORMAL,FANCY)                        marg   -0.6000000E+10
   MAKE(PLANT1,CHAIRS,NORMAL,FUNCT)                        marg   -0.2700000E+11
   MAKE(PLANT1,CHAIRS,NORMAL,FANCY)                        marg   -0.3700000E+11
   MAKE(PLANT1,CHAIRS,MAXSML,FUNCT)                        marg   -0.3000000E+11
   MAKE(PLANT1,CHAIRS,MAXSML,FANCY)                        marg   -0.3800000E+11
   MAKE(PLANT1,CHAIRS,MAXLRG,FUNCT)                        marg   -0.3000000E+11
   MAKE(PLANT1,CHAIRS,MAXLRG,FANCY)                        marg   -0.3800000E+11
   MAKE(PLANT2,CHAIRS,NORMAL,FUNCT)                        marg   -0.2700000E+11
   MAKE(PLANT2,CHAIRS,NORMAL,FANCY)                        marg   -0.3700000E+11
   MAKE(PLANT2,CHAIRS,MAXSML,FUNCT)                        marg   -0.3000000E+11
   MAKE(PLANT2,CHAIRS,MAXSML,FANCY)                        marg   -0.3800000E+11
   MAKE(PLANT2,CHAIRS,MAXLRG,FUNCT)                        marg   -0.3000000E+11
   MAKE(PLANT2,CHAIRS,MAXLRG,FANCY)                        marg   -0.3800000E+11
   ARTRESOUR(PLANT2,TOP)                                   marg   -0.1000000E+10

 ----### THESE EQUATIONS MAY CAUSE INFEASIBLE
        Since their marginals are so large

   RESOUREQ(PLANT1,SMLLATHE)                               marg    0.1000000E+10
   RESOUREQ(PLANT1,LRGLATHE)                               marg    0.1000000E+10
   RESOUREQ(PLANT1,CARVER)                                 marg    0.1000000E+10
   RESOUREQ(PLANT1,LABOR)                                  marg    0.1000000E+10
   RESOUREQ(PLANT1,TOP)                                    marg    0.1000000E+10
   RESOUREQ(PLANT2,SMLLATHE)                               marg    0.1000000E+10
   RESOUREQ(PLANT2,LRGLATHE)                               marg    0.1000000E+10
   RESOUREQ(PLANT2,CARVER)                                 marg    0.1000000E+10
   RESOUREQ(PLANT2,LABOR)                                  marg    0.1000000E+10
```

Table 9.4 Output on Small Unbounded Example to Original Model

Panel A Unbounded Solution

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      3 UNBOUNDED

                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU OBJ             .         .         .        1.000
---- EQU R1            -INF        .         .        3.000
---- EQU R2            -INF     20.000    20.000      2.000

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR OBJMAX        -INF     40.000     +INF        .
---- VAR X1              .         .       +INF        .
---- VAR X2              .         .       +INF      2.000 UNBND
---- VAR X3              .      20.000     +INF        .
```

Panel B NONOPT Output

```
 ----###  LISTING NONOPTIMAL VARIABLES

     X2
       Level     0.00000000E+00 Marginal  2.0000000
       Low Bound 0.00000000E+00 Up Bound 0.30000000E+31
```

Panel C GAMS Solution Output after Bounds Applied

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE      20000000040.0001

                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU OBJ             .         .         .        1.000
---- EQU R1            -INF        .         .        1.000
---- EQU R2            -INF     20.000    20.000      2.000

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR OBJMAX        -INF  2.000E+10     +INF        .
---- VAR X1              .    1.000E+10 1.000E+10     2.000
---- VAR X2              .    1.000E+10     +INF        .
---- VAR X3              .       20.000 1.000E+10      .
```

Table 9.5 Solution for Large Unbounded Example

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE    1193000234500.0020

                       LOWER     LEVEL     UPPER    MARGINAL
---- EQU OBJT            .         .         .        1.000

---- EQU RESOUREQ    RESOURCES AVAILABLE
                     LOWER     LEVEL     UPPER    MARGINAL
PLANT1.SMLLATHE     -INF       .     1100.000      .
PLANT1.LRGLATHE     -INF       .      880.000      .
PLANT1.CARVER       -INF       .      500.000      .
PLANT1.LABOR        -INF   1750.000  1750.000   134.000
PLANT1.TOP          -INF    350.000   500.000      .
PLANT2.SMLLATHE     -INF       .     1400.000      .
PLANT2.LRGLATHE     -INF       .      900.000      .
PLANT2.CARVER       -INF       .     1200.000      .
PLANT2.LABOR        -INF       .     1250.000      .

---- EQU PLANTPROD   PRODUCT BALANCE FOR A PLANT
                       LOWER     LEVEL     UPPER    MARGINAL
PLANT1.TABLES.FUNCT     -INF       .         .      482.000
PLANT1.TABLES.FANCY     -INF       .         .      770.000
PLANT1.CHAIRS.FUNCT     -INF       .         .        5.000
PLANT1.CHAIRS.FANCY     -INF       .         .        5.000
PLANT2.TABLES.FUNCT     -INF       .         .       14.000
PLANT2.TABLES.FANCY     -INF       .         .       18.000
PLANT2.CHAIRS.FUNCT     -INF       .         .        5.000
PLANT2.CHAIRS.FANCY     -INF       .         .        5.000

---- VAR MAKE        NUMBER OF ITEMS MADE
                              LOWER     LEVEL     UPPER    MARGINAL
PLANT1.TABLES.NORMAL.FUNCT      .         .        +INF       .
PLANT1.TABLES.NORMAL.FANCY      .      350.000     +INF       .
PLANT1.CHAIRS.NORMAL.FUNCT      .         .        +INF  -1350.000
PLANT1.CHAIRS.NORMAL.FANCY      .         .        +INF  -1092.000
PLANT1.CHAIRS.MAXSML.FUNCT      .         .        +INF  -1485.000
PLANT1.CHAIRS.MAXSML.FANCY      .         .        +INF  -1093.000
PLANT1.CHAIRS.MAXLRG.FUNCT      .         .        +INF  -1486.000
PLANT1.CHAIRS.MAXLRG.FANCY      .         .        +INF  -1094.000
PLANT2.CHAIRS.NORMAL.FUNCT      .         .        +INF    -10.000
PLANT2.CHAIRS.NORMAL.FANCY      .         .        +INF    -20.000
PLANT2.CHAIRS.MAXSML.FUNCT      .         .        +INF    -11.000
PLANT2.CHAIRS.MAXSML.FANCY      .         .        +INF    -21.000
PLANT2.CHAIRS.MAXLRG.FUNCT      .         .        +INF    -12.000
PLANT2.CHAIRS.MAXLRG.FANCY      .         .        +INF    -22.000

---- VAR TRNSPORT    NUMBER OF ITEMS TRANSPORTED
                              LOWER     LEVEL     UPPER    MARGINAL
TABLES.FUNCT.PLANT1.PLANT2      .    1.0000E+9     +INF       .
TABLES.FANCY.PLANT1.PLANT2      .    1.0000E+9     +INF       .
CHAIRS.FUNCT.PLANT1.PLANT2      .    4.0000E+9     +INF       .
CHAIRS.FUNCT.PLANT2.PLANT1      .         .        +INF       .
CHAIRS.FANCY.PLANT1.PLANT2      .    6.0000E+9     +INF       .
CHAIRS.FANCY.PLANT2.PLANT1      .     2100.000     +INF       .

---- VAR SELL        NUMBER OF SETS SOLD
                 LOWER     LEVEL     UPPER    MARGINAL
PLANT1.FUNCT       .         .    1.0000E+9  -102.000
PLANT1.FANCY       .      350.000 1.0000E+9      .
PLANT2.FUNCT       .    1.0000E+9 1.0000E+9   391.000
PLANT2.FANCY       .    1.0000E+9 1.0000E+9   802.000

                       LOWER     LEVEL     UPPER    MARGINAL
---- VAR NETINCOME      -INF  1.193E+12     +INF       .
```

Table 9.6 NONOPT Output for Unbounded Model after Large Bounds Applied

```
 ----#### Executing NONOPT

 ----### THESE VARIABLES ARE POTENTIALLY UNBOUNDED
         Since their levels are so large
```

```
TRNSPORT(TABLES,FUNCT,PLANT1,PLANT2)                level   0.1000000E+10
TRNSPORT(TABLES,FANCY,PLANT1,PLANT2)                level   0.1000000E+10
TRNSPORT(CHAIRS,FUNCT,PLANT1,PLANT2)                level   0.4000000E+10
TRNSPORT(CHAIRS,FANCY,PLANT1,PLANT2)                level   0.6000000E+10
SELL(PLANT2,FUNCT)                                  level   0.1000000E+10
SELL(PLANT2,FANCY)                                  level   0.1000000E+10
```

Table 9.7 Tableau of Budgeting Example

Row	Buy Misc.	Sell Corn	Sell Soyb.	Sell Pork	Prod Corn	Prod. Soyb.	Prod Hogs	RHS
Objective Function	-1	2.5	6	0.5	-75	-50		MAX
Land Available					1	1		≤ 600
Pork Balance				1			-1000	≤ 0
Soybean Bal			1			-50		≤ 0
Corn Balance		1			-120		20	≤ 0
Misc. Inp. Bal.	-1				50	20		≤ 0

Table 9.8 GAMS Solution for Budget Example Model

```
                            LOWER     LEVEL     UPPER    MARGINAL
---- EQU OBJ                  .         .         .        1.000
---- EQU LANDAVAIL          -INF     600.000   600.000  2680.000
---- EQU PORKBALAN          -INF        .         .        0.500
---- EQU SOYBEANBAL         -INF        .         .       55.600
---- EQU CORNBAL            -INF        .         .       24.000
---- EQU MISCINPBAL         -INF        .         .        1.000

                           LOWER     LEVEL     UPPER    MARGINAL
---- VAR OBJMAX             -INF  1.6080E+6     +INF       .
---- VAR BUYMISC             .    1.4700E+5     +INF       .
---- VAR SELLCORN            .         .        +INF    -21.500
---- VAR SELLSOYB            .         .        +INF    -49.600
---- VAR SELLPORK            .    3.6000E+6     +INF       .
---- VAR PRODCORN            .      600.000     +INF       .
---- VAR PRODSOYB            .         .        +INF       .
---- VAR PRODHOG             .     3600.000     +INF       .
```

Table 9.9 Parts of POSTOPT Output for Budget Example Model

Panel A: Sell SOYB Variable

SOLUTION VALUE	0.000000E+00		
EQN	Aij	Ui	Aij*Ui
OBJ	-6.0000	1.0000	-6.0000
SOYBEANBAL	1.0000	55.600	55.600
TRUE REDUCED COST			49.600

Panel B: PRODCORN Variable

SOLUTION VALUE	600.000		
EQN	Aij	Ui	Aij*Ui
OBJ	75.000	1.0000	75.000
LANDAVAIL	1.0000	2680.0	2680.0
CORNBAL	-120.00	24.000	-2880.0
MISCINPBAL	125.00	1.0000	125.00
TRUE REDUCED COST			0.00000E+00

Panel C: PRODSOYB Variable

SOLUTION VALUE	0.000000E+00		
EQN	Aij	Ui	Aij*Ui
OBJ	50.000	1.0000	50.000
LANDAVAIL	1.0000	2680.0	2680.0
SOYBEANBAL	-50.000	55.600	-2780.0
MISCINPBAL	50.000	1.0000	50.000
TRUE REDUCED COST			0.00000E+00

Panel D: PRODHOG Variable

SOLUTION VALUE	3600.00		
EQN	Aij	Ui	Aij*Ui
PORKBALAN	-1000.0	0.50000	-500.00
CORNBAL	20.000	24.000	480.00
MISCINPBAL	20.000	1.0000	20.000
TRUE REDUCED COST			0.00000E+00

Table 9.10 Row Summing Example

Row	Buy Misc.	Sell Corn	Sell Soyb.	Sell Pork	Prod Corn	Prod Soyb.	Prod Hogs	RHS
Objective Func	-1	2.5	6	0.5	-75	-50	MAX	
Land Available					1	1	≤	600
Pork Balance				1			-150	≤ 0
Soybean Bal			1			-50	≤	0
Corn Balance		1			-1200	20	≤	20
Misc. Inp. Bal.	-1				125	50	20	≤ 0

Table 9.11 GAMS Solution for Row Summing Example Model

```
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE          1860055.0000

                          LOWER     LEVEL     UPPER    MARGINAL
---- EQU OBJ                .         .         .        1.000
---- EQU LANDAVAIL        -INF     600.000   600.000   3100.000
---- EQU PORKBALAN        -INF       .         .         0.500
---- EQU SOYBEANBAL       -INF       .         .        64.000
---- EQU CORNBAL          -INF      20.000    20.000     2.750
---- EQU MISCINPBAL       -INF       .         .         1.000

                          LOWER     LEVEL     UPPER    MARGINAL
---- VAR OBJMAX           -INF   1.8601E+6     +INF       .
---- VAR BUYMISC            .    7.9502E+5     +INF       .
---- VAR SELLCORN           .         .        +INF     -0.250
---- VAR SELLSOYB           .         .        +INF    -58.000
---- VAR SELLPORK           .    5.4002E+6     +INF       .
---- VAR PRODCORN           .      600.000     +INF       .
---- VAR PRODSOYB           .         .        +INF       .
---- VAR PRODHOG            .    36001.000     +INF       .
```

Table 9.12 POSTOPT Output for Row Summing Example Model

Panel A: PORKBALAN Equation

VAR	Aij	Xj	Aij*Xj
SELLPORK	1.0000	0.54002E+07	0.54002E+07
PRODHOG	-150.00	36001.	-0.54002E+07
=L=			=L=
RHS COEFF			0.00000E+00
SLACK EQUALS			0.00000E+00
SHADOW PRICE			0.50000

Panel B: MISCINPBAL Equation

VAR	Aij	Xj	Aij*Xj
BUYMISC	-1.0000	0.79502E+06	-0.79502E+06
PRODCORN	125.00	600.00	75000.
PRODSOYB	50.000	0.00000E+00	0.00000E+00
PRODHOG	20.000	36000.	0.72002E+06
=L=			=L=
RHS COEFF			0.00000E+00
SLACK EQUALS			0.00000E+00
SHADOW PRICE			1.0000

Panel C: CORNBAL Equation

VAR	Aij	Xj	Aij*Xj
SELLCORN	1.0000	0.00000E+00	0.00000E+00
PRODCORN	-1200.0	600.00	-0.72000E+06
PRODHOG	20.000	36000.	0.72002E+06
=L=			=L=
RHS COEFF			20.00
SLACK EQUALS			0.00000E+00
SHADOW PRICE			2.7500

Chapter 10 Dealing with Models Which Abnormally Terminate

When running GAMS programs execution can abnormally terminate. Such terminations (hereafter called failures) occur because, among other reasons: a) GAMS limits are reached; b) solver limits are reached; c) computer memory is exhausted; d) the solver uses considerable time, but does not reach an optimum; or e) the solver halts citing inability to make significant progress. There are techniques available to correct such problems. Some require problem reformulation. Others require GAMS code modifications. Yet others require the discovery of problems within the GAMS code. The sections below cover ways to repair GAMS programs so as to remove the causes of failures.

The first thing one should do when faced with a failure is examine the LST file. Special attention should be paid to all places where four asterisks (****) appear. Such places may show how to repair the problem encountered, and usually will report the solver termination status. One should also examine the end of the listing. The LST file often contains messages such as

"resources interrupt,"

"maximum executable code space exceeded," or

"iteration limit exceeded"

along with a suggested remedy. Unfortunately, such simple remedies are not always appropriate. Namely, when the LST file contains a dump or when the solver stops for time limit, lack of progress, or memory capacity, then more advanced remedies need be pursued. We will discuss ways of resolving such difficulties under the topics of correcting excessive memory use; scaling; degeneracy resolution; and problem reformulation.

10.1 Expanding GAMS and Solver Limits

There are a number of limits within GAMS and the associated solvers. Often the need to

expand these limits is indicated by messages on the screen or in the LST file indicating “resource interrupt”; “maximum executable code space exceeded”; or “iteration limit exceeded.” Here we list procedures to expand such limits.

10.1.1 Expanding Iteration, Resources and Work space

Users may discover messages indicating that a) the "iteration limit" has been reached; b) "Work space" has been exceeded; c) or the program ran out of "resources.” These are correctable using GAMS statements. Expanding the iteration limit is done using

```
OPTION        ITERLIM = 100000;
```

where the 100000 may be replaced with any number. Similarly when a resources interrupt is encountered, this can be fixed through the command

```
OPTION        RESLIM = 2000000;
```

where the 2000000 can be replaced by any other number. The Work space limit can be expanded by putting in the command

```
ModelName.Workspace = 30;
```

where the number (30) gives the Work space limit in megabytes. The number entered should not exceed the megabytes on the computer and may need to be smaller than that estimated by the solver so the problem fits on the computer.

10.1.2 Allowing More Executable Code Space

One can reach an error involving executable code space particularly when using large numbers of executable statements in LOOPS. In that case an error message appears on the screen about limits in PRESCAN and in the LST file the message appears.

"Maximum Executable Code Size Exceeded."

In this case one would add the CODEX parameter to a GAMS submission statement, i.e.,

GAMS MYMODEL CODEX = 1

or if that is insufficient use CODEX = 2 or more. Note this should only be employed when the error is encountered as this option causes GAMS to use more memory. If this does not fix the problem, then the amount of code within loops in the GAMS program must be reduced. This would be done by cutting down on the number of statements in include files within LOOPS or by not including basis files etc. Note, including a large GAMSBAS basis file in LOOPS can be the cause of this error.

10.1.3 Expanding Solver Specific Limits

A number of the GAMS based solvers contain internal limits. For example, MINOS5 has a major iteration limit and OSL has a Work space limit. In theses cases one needs to use either the option file or the GAMS WORK SPACE command as discussed in 10.1.1.

Solver limits typically are expanded through the use of the options file. The options file is invoked by using the command

Modelname.OPTFILE=1;

In turn, one creates a file named solvername.opt for example MINOS5.OPT or OSL.OPT in which solver allowed options are specified. Thus, if one finds a message about the major iteration limit in the LST file when using MINOS5, one would turn on the option file then specify the MINOS5.OPT file containing the line

Major iterations 1000

or some other appropriate value. Note the default is 50.

A review of the solver manuals distributed with GAMS reveals the types of options that can be specified for each solver. The LST file will often reveal what limits need to be relaxed to expand solver limits.

One final note on option files is in order. GAMS permits the user to have multiple option files. These are chosen by specifying the command turning on the option file with numbers other than one. For example,

modelname.OPTFILE=4;

In such a case the option file name becomes

solvername.OP4 or MINOS5.OP4 in the case of MINOS5 usage.

Values up to 99 can be used. For values between two and nine the option file name extension is OP followed by the number (i.e. OP5). For values between ten and ninety-nine the option file name extension is O followed by the number (i.e.O78).

10.2 Finding Excessive Memory Use

GAMS models may fail because of excessive memory use. This is generally accompanied by rather mystifying error messages such as “out of dynamic memory”, “segmentation fault” or some obscure message accompanying a core dump. Such an error may occur at one of four execution phases. GAMS can run out of memory when compiling, carrying out calculations, generating a model or solving. Compilation stage memory errors virtually always mean the problem is too big or the computer too small. Failures during the calculation phase involve initial data computation, assignment of upper and lower bounds to variables, assignment of scaling factors to variables/equations, or report writing. Model generation failures occur when GAMS is setting up the numerical version of the model for passage to the solver. Failures during solver execution occur because the optimizer needs additional memory to adequately store and solve the problem. Failure during report writing involve the use of excessive memory space during report writing. Different strategies are appropriate for resolving errors at each of these stages.

10.2.1 Memory Use Problems -- Root Causes

Memory induced failures occur when the computer is too small or more commonly when the implementation inappropriately renders the problem too big. The later problem usually occurs when the GAMS code has been set up with multidimensional entities which have excessive and irrelevant elements.

Consider the following example. Suppose one defines a parameter with respect to seven sets, each of which have ten elements. Suppose the sets are called I, J, K, L, M, N, and O. Now suppose we define a variable X, a parameter Y and an equation Z each with all these sets indicated as follows:

```
1       SET                     I       /1*10/
2                               J       /1*10/
3                               K       /1*10/
4                               L       /1*10/
5                               M       /1*10/
6                               N       /1*10/
7                               O       /1*10/;
8       PARAMETER               Y(I,J,K,L,M,N,O);
9       VARIABLE                X(I,J,K,L,M,N,O)
10                              OBJ;
11      EQUATION                Z(I,J,K,L,M,N,O)
12                              OB;
13      Y(I,J,K,L,M,N,O)                =10;
14      X.UP(I,J,K,L,M,N,O)             =10;
15      X.SCALE(I,J,K,L,M,N,O) =1000;
16      OB..
17              OBJ                     =E=     SUM((I,J,K,L,M,N,O),X(I,J,K,L,M,N,O));
18      Z(I,J,K,L,M,N,O)..
19              X(I,J,K,L,M,N,O)        =E=     8;
```

Execution of any of the statements 13-15 runs a 64-megabyte workstation out of memory during model calculations as the number of joint cases of all seven parameters means that we need to store 10 million numbers. Most computers would need over 100 megs of memory just to store these commands by themselves. Lines 17 or 19 also run the computer out of memory during model generation as 10 million variables are defined by 17 while 19 defines 10 million variables and equations. A huge memory requirement would be unavoidable if this number of elements is truly needed and a bigger computer needed. More commonly however the 10 million elements are not all relevant and one needs to restrict the cases treated to particular interactions of the seven subscripts. This is covered below and in Chapter 12.

10.2.2 GAMS Tools for Examining Memory Use

GAMS provides two ways that one can discover something about memory use. The easy techniques involve use of the symbolic dump and the profile option. Let us address each of these tools in sequence.

10.2.2.1 The Symbolic Dump

During any stage of GAMS program execution (not during compilation, model generation or solution) it is possible to dump out memory use by each GAMS symbol. This dump is requested by using the command:

OPTION DMPSYM;

In turn a dump of the memory use by all the symbols will be produced. This dump indicates memory use at the point in program execution where the OPTION command occurs (memory use up until that point will be reflected but memory use by subsequent statements will not). Suppose we illustrate such a dump using the example model in Table 10.1, with line 33 activated. The resultant output is in Table 10.2. Notice that in the dump, the lines numbered 1-42 are not relevant as they simply report the space used by internal GAMS functions. However, starting in line 43, all the sets, parameters, variables and equations used in the GAMS model are listed along with a reporting of the amount of space they use. For example, in line 43, the dump indicates that the set I is one dimensional, of length three elements (the other information on such lines will not be discussed as it is of little value to the user). Similarly, notice that in line 50 that Y is shown to be a seven dimensional parameter which has a length or memory usage of 2,187 elements while line 51 shows Q has 27 elements. The element counts displayed are the number of entries for each item and in this case are products of the maximum dimensions of the set elements (i.e., since Q is defined over three sets each with 3 elements then $3^3 = 27$ elements are defined). Thus, by

looking at the length parameter in the symbol dump, one can figure out the relative amount of memory used for each data item at the place the option command is invoked. By examining items with large lengths one can find out where the bulk of the memory is being utilized. (Not all element counts use equal storage. For example, set storage uses a smaller word size and less memory than most of the other data items). This dump can be requested anywhere within the program as many times as desired.

10.2.2.2 The Profile Information

The symbolic memory use dump indicates how much space is used for each array but does not necessarily tell where the arrays are becoming large. The GAMS profile option generates information on where large data sets are being built. The profile option is invoked by one of two means. One can alter the initial GAMS call to include commands like the following:

```
GAMS MYMODEL   PROFILE = 1        on DOS machines or
GAMS MYMODEL   -PROFILE   2       on UNIX machines.
```

or one can insert an option statement into the program as follows:

```
OPTION PROFILE=3;
```

The profile option causes GAMS to report the execution characteristics of individual statements. The number after the profile request tells GAMS how deeply within LOOPS and IF statements to report execution characteristics. When PROFILE = 1 is used, GAMS does not give information within any IF statements or LOOPS, just information on the overall IF or LOOP. When PROFILE = 2 is used, execution characteristics are given on statements within the first level of LOOP and IF statements. PROFILE = 3 will go within the second level of IF and LOOP statements, etc. Large profile values are needed to investigate execution characteristics within deeply nested LOOP and IF statements but they can generate a lot of output if the loops are executed repeatedly.

Let us look at a profile report (Table 10.3) generated by using OPTION PROFILE = 1 on the model in Table 10.1. The profile command causes GAMS to give information on: a) the GAMS statement number of the instruction being profiled; b) the symbol name being executed or being worked on; c) the execution time of each statement; d) cumulative program execution time; e) cumulative memory use; and f) the number of cases for which the statement is executed (if the cases exceed one). In the example, the entry labeled starting with a 21 reports that executing the 21st line of the program (Table 10.1) which is an assignment of values into Y takes 0.08 of a second execution time, contributing to 0.09 of a second of cumulative execution time while executing the statement for 2,187 cases. Through this information, the profile indicates where large numbers of cases and/or large execution times are encountered (i.e., statements 21, 22, 23, 28 and 26). In turn, one can examine those statements to see if they are generating large memory requirements.

One notable thing about the profile output regards the way the profile output works with respect to a solve statement. Notice that in the section involving lines 28-37, 28, 30, 26, and 37 in the profile output we have the output from the solver and notice that in generating the objective function our memory use jumps from .3 to .8 megabytes. Then one generating the follow b it generates from .8 up to .9. Thus one can monitor weather any of the equations are causing memory problems during their generation, subject to the frailties of the output which will be discussed below.

The profile option can generate a tremendous amount of output much of which is not informative. Our example shows several statements are reported for which there is not meaningful execution time. One can suppress this information by using a tolerance on the minimum amount of execution time that a statement must use to be reported. In the case of the

example this is a very small amount of time and we could use

OPTION PROFILETOL = .01;

In bigger models one could for example, allow reporting of statements that took 1, 2, 10 or more seconds of execution time. Notice in the example model the statements with the largest execution times are the ones that go through thousands of cases or have terms summing over thousands of set index possibilities.

.

10.2.3 Finding Memory Use Problems

Unfortunately while the above tools for memory use tracking are present in GAMS, their use rarely allows easy discovery of the causes of memory use induced crash because of the way computers and output buffers work. Namely, when a memory interruption is encountered the computer halts leaving a partial LST file. The LST file is partial because it is periodically written from internal output buffers. The information contained in the output buffer at the time of termination is usually lost, thus the end of the LST file is lost. This means that, for example under the profile option the statements executed just prior to encountering the problem will not be reported in the LST file. In fact the last statements in the LST file may be ones executed a considerable number of statements before the problem was encountered. This sets the stage for the use of search strategies in finding memory utilization problems. However, before we reveal those strategies we need to return to the small to large argument. When a large model is running out of memory, one should consider using a small version of the model with the tools outlined above to find large memory uses and if possible fix the problem, then come back to the big model and see if that has corrected the problem. On the other hand, if the problem only exists in the big model then one should use the approaches below.

10.2.3.1 Finding Excessive Compilation Memory Use

GAMS can run out of memory during compilation. This is an uncommon occurrence. Nevertheless, when it does happen it means that the problem is too big or the computer too small. One then must: a) reduce the problem size; b) or use a larger computer and/or c) use a larger GAMS version (other than the demo). Little else can be said. However, if the user feels the problem is not that big, then the small to large strategy of Chapter 5 should be employed. If that does not reveal a problem, then use the search techniques discussed in the next section to find the problem and repair the problem.

10.2.3.2 Finding Excessive Calculation Memory Use

When GAMS is doing pre or post solution calculations, it can run out of memory. Commonly this is caused by runaway dimensions in parameter definitions where excessive elements are defined. Statements involving assignments of parameters, variable bounds and scaling factors are usually the cause. Unfortunately, when one runs out of memory during GAMS calculations one does not often know where the error has arisen. The output buffer problem discussed above means that the output in the LOG and LST files may not point to the place at which the program terminated.

Thus, users often must search for the last properly executing statement. The basic way of conducting such a search involves examining the end of the LST, LOG file or screen listing to form a guess at the last line where execution was successfully accomplished. Notice only the screen output will always indicate this (providing one is fast and attentive enough to read it) because of the lost final output buffer. Regardless, once a successfully executing statement is identified then one of the subsequent statements must be the source of the problem. Users then can utilize the GAMS $ON/OFFTEXT syntax to search out the offending statement. This is done

by commenting out subsequent statements and seeing if the program runs. If the program runs then one knows that the excessive memory use is encountered somewhere within the $ON/OFFTEXT. Other uses move the $ONTEXT to allow more of the program to execute. Continue until a statement is found that if authorized causes job failure. Suppose we found out we were running out of memory in the example somewhere after statement 20. A search would begin by putting an $ONTEXT between lines 20 and 21 and then an $OFFTEXT at the bottom. If the reduced program ran, then we could move the $ONTEXT from between lines 20 and 21 down 8 statements to line 28 (allowing one half the remaining statements to execute). Then we would submit the GAMS job again and probably find out that the program didn't run. If so we could move the $ONTEXT up to line 24 halving the number of potentially improper statements. We would continue restricting the number of statements until we identified that for example statement 21 was the cause of the difficulty. Having found the error we would likely then use the $ conditions discussed in Chapter 12 to eliminate irrelevant cases correcting the excessive memory use.

This is a time consuming procedure particularly in big models make it. Efficiency can be increased by using GAMS save and restart files to save the part of the program that performs satisfactorily (i.e., up to line 20) and then only execute the small part of the program in which the problem is occurring (see the chapter on comparative model runs for an example).

10.2.3.3 Finding Excessive Model Generation Memory Use

The third place where GAMS may run out of memory is during model generation. This is again a difficult stage at which to discover exactly where the problem is located, largely because of the print buffer problem. One should, if possible, watch the screen and take note of the last statement number which is executed i.e., if it says executing line 26 one would know the problem

is in line 26 or later. If this is not possible one should save the log file (using the LO2 option on the GAMS call) along with the use of the memory status dump OPTION SYS7 = 1 and go through these outputs to find the hypothesized last "good" statement. Also one should look at the last memory status dump and make sure that an excessive amount of memory is not used up to that point.

The search for the bad statement is then on. This search is conducted by forming a restricted model. The restricted model may be formed by

1) commenting out (using asterisks or $ON/OFFTEXT) certain equations from the equation listing and the equation definitions (.. Entries)
2) creating a model statement which omits a number of the equations, or by
3) following a small to large approach by restricting the sets over which equations are generated and terms within the equations are summed. For example, if one had a model in which there were constraints generated for every month, for a number of years involving sums over every crop, the constraint and terms could be redefined over subsets containing only January and February, the first and second years and two selected crops.

In turn, the restricted model can be run to see if it works and to examine if the memory use.

The basic methodology for discovering the location and cause of excessive memory use is:

1) identify the last equation successfully executed;
2) identify the equations that have not been generated as of the last known point of successful execution. Note, GAMS equations are generated in the order in which the names appear in the EQUATION declaration section not in the order in which the equation definitions appear (the ..syntax). (That is why statement 28 appears

before equation 26 in the SYS7 dump on Table 10.5). Thus, if equation B is declared right before A and equation B is the last successfully generated, then one can conclude that equation B along with all equations above it are satisfactory. So then one would investigate the subsequent equations from equation A on down.

3) eliminate or simplify all subsequent equations using one of the procedures discussed above.

4) resubmit the GAMS job.

5) if the model runs and more than one equation was eliminated or simplified restore some of the earlier of these to full executing status. Thus reinclude at least the first eliminated equation and if say four equations were eliminated include the first two in some form of a binary search. Go to Step 4 if eliminating more equations, if not go to Step 6.

6) now an equation causing the problem has been identified. One should then visually investigate whether the problem is with the dimension of the equation or the variables and coefficients generated by the terms within the equation. This would be done by examining the equation to see if it contains a large number of different equations because of its set dependence of whether the variables within it have a large number of elements due to their set dependence. In the Table 10.1 example we have 3^7 cases of the variable created when executing line 26 while line 28 involves that many variables and constraints. If a problem is found then fix it and restore all equations to full status and regenerate the model.

7) If visual inspection is not conclusive then do numerical inspections. This is done by reducing the number or size of the sets used in defining the equation and the

included terms or by commenting out selected terms until successful execution is realized. One can support this examination with the symbol table, memory status dump, profile, and SYS7=1 output as well as the GAMSCHK tools to gain more information. Keep going until the problem is found or conclude a larger computer is needed.

10.2.3.4 Finding Excessive Solver Memory Use

One of the nastier memory use faults involves the conclusion by a solver that it has run out of memory and cannot proceed. In this case there are five possibilities:

a) The solver may overestimate needed Work space and quit when it can't get that much;

b) The solver may underestimate needed Work space and quit when it runs out;

c) The solver may be inefficient in terms of Work space and won't fit, but an alternative, more space efficient, solver may fit;

d) The problem may be too big for the computer;

e) The problem may be improperly too big;

The first two cases can be dealt with rather simply. All of the solvers are guessing when they specify their "memory requirement". The size of the basis inverse and the branch and bound tree can vary widely depending on problem characteristics. Solvers take a conservative guess which is often bigger than required. One can fix this by manually specifying a Work space limit consistent with the computer memory as follows

```
OPTION WORK = 20; or
MODELNAME.WORKSPACE= 20;
```

where 20 give the space requested in megabytes. One can use a number smaller or larger than the

solver estimate and have the model work. However, it is possible that the Work space cannot be made big enough.

Unfortunately, fixing the Work space does not always work. The third through fifth cases may arise. Consider for now the third, it is possible the problem can be made to fit by switching solvers. A recent example revealed that for the same problem OSL estimated its requirement at 69 megs, MINOS required 26 megs and CPLEX even less. Thus when one runs out of memory one could switch to one of the more memory efficient codes. However, one should investigate whether the problem is inappropriately too big.

The fourth and fifth cases involve models that are too big for the computer. Models may be too big for the computer because of model specification errors or because of reality. There is little that can be done about models that are really too big other than simplifying the model or using a bigger computer. More commonly however the fifth case arises where models are artificially large because of specification errors which cause GAMS to generate numerous inappropriate equations and or variables.

When a solver fails due to memory limits one should investigate to make sure the model is structurally correct. The primary tools for this are to run a solver PRESOLVE to see if a lot of the model can be eliminated or use the GAMSCHK structural investigation tools to see if the model is well formed. The GAMSCHK tools involve those discussed in Chapter 8 and the GAMSCHK writeup such as BLOCKLIST or BLOCKPIC to see if the number of variables and coefficients by block are reasonable. If there are a lot of irrelevant items then those cases should be eliminated using the $ controls discussed in chapter 12. For example, one might only generate transport variables for routes with viable transportation costs.

One can also have GAMS suppress generation of irrelevant variables by zeroing out their

upper bounds and then using the GAMS option command

modelname.HOLDFIX = 1

However, this option should be used sparingly as computer time and memory will be reduced more if the GAMS instructions can be altered to eliminate consideration of those variables.

10.3 Scaling

Solution of mathematical programming problems usually requires manipulation of large matrices. The heart of most solvers includes a sparse matrix inverter. Numerical problems often arise within such procedures. Many numerical analysis studies have found that such algorithms work better when the matrix to be inverted is well scaled. A poorly scaled model can cause excessive time to be taken in solution or can cause the solver to fail.

There are several ways to approach the scaling issue within GAMS. First, one can ignore scaling and let the solvers scale as they might. Second, one can use the GAMS solver option file to enhance or, if needed, invoke scaling. Scaling is usually done by default, but in some solvers like MINOS certain forms of scaling are only done on user request (for example nonlinear variable scaling must be requested in MINOS). Lastly, one can prescale the model manually where all coefficients associated with individual variables and/or constraints are divided (or multiplied) through by scaling factors. This can be done using the GAMS scaling commands or by modifying the model data. Each of these options will be discussed below. First, however we discuss the basic concept and theory of scaling.

10.3.1 The Goal of Scaling

Scaling efforts should try to reduce the disparity between the coefficient magnitudes and nonlinear variable gradients. The disparity between the coefficients should be reduced so that the absolute value of their magnitude is within 100 to 1000. We try to scale models so that the

equation coefficients fall in absolute value between 0.1 and 10 with possibly a few smaller and larger numbers. This need not include right hand side and/or objective function coefficients. There again we try to keep the numbers within a couple of orders of magnitude and often find it desirable to uniformly divide the right hand sides and bounds through by a right hand side scalar. However, GAMS does not contain a RHS/bound automatic scaling feature. Thus, such scaling would have to be implemented manually.

10.3.2 The Effect of Scaling

Let us consider scaling theoretically. Given the LP problem

$$\begin{aligned} \text{Max} \quad & c_1 X_1 + c_2 X_2 \\ \text{s.t.} \quad & a_{11} X_1 + a_{12} X_2 \le b_1 \\ & a_{21} X_1 + a_{22} X_2 \le b_2 \\ & X_1, \quad X_2 \ge 0 \end{aligned}$$

Suppose we wish to change the units of the coefficients for a variable (for example, multiplying by one thousand). This requires substituting in a new variable with different units (for example in thousands of acres rather than acres). Model consistency requires all coefficients associated with a variable have a common denominator unit, this implies every coefficient in the column associated with the new variable needs to be multiplied by the unit transformation factor. Thus, we substitute in a new variable $X_j^{'}$ which equals the old variable divided by a scalar (X_j/SC_j). We also need to adjust the a_{ij}'s and C_j's such that

$$a_{ij}' = SC_j * a_{ij}$$

where a_{ij}' is the new coefficient, a_{ij} the old coefficient and SC_j the unit transformation factor. Thus, when we X_1 we do the following. First, divide and multiply each term by the scaling factor. Second, substitute in $c_j^{'}$, $a_{ij}^{'}$, and $X_j^{'}$ yielding

$$
\begin{aligned}
\text{Max} \quad & SC_1\ c_1\ X_1 \ /\ SC_1 \quad + c_2\ X_2 \\
\text{s.t.} \quad & SC_1\ a_{11}\ X_1 \ /\ SC_1 \quad + a_{12}\ X_2 \ \le\ b_1 \\
& SC_1\ a_{21}\ X_1 \ /\ SC_1 \quad + a_{22}\ X_2 \ \le\ b_2 \\
& \qquad X_1 \ /\ SC_1, \qquad\qquad X_2 \ \ge\ 0.
\end{aligned}
$$

$$
\begin{aligned}
\text{Max} \quad & c_1'\ X_1' \quad + c_2\ X_2 \\
\text{s.t.} \quad & a_{11}'\ X_1' \quad + a_{12}\ X_2 \ \le\ b_1 \\
& a_{21}'\ X_1' \quad + a_{22}\ X_2 \ \le\ b_2 \\
& \qquad X_1', \qquad\qquad X_2 \ \ge\ 0.
\end{aligned}
$$

Where the c', X' and a' are c, X and a items as the transformed after scaling.

Scaling is also done on the equations. When scaling equations; e.g., dividing through by one thousand, every equation coefficient is divided by the scaling factor (SR) as follows:

$$
\begin{aligned}
\text{Max} \quad & c_1\ X_1 \quad + \quad c_2\ X_2 \\
\text{s.t.} \quad & a_{11} \ /\ SR\ X_1 \quad + a_{12} \ /\ SR\ X_2 \ \le\ b_1 \ /\ SR \\
& a_{21}\ X_1 \quad + \quad a_{22}\ X_2 \ \le\ b_2 \\
& \quad X_1, \qquad\qquad X_2 \ \ge\ 0
\end{aligned}
$$

where SR is a positive equation scaling factor.

Two other types of scaling are also relevant. Suppose that the right-hand sides are scaled, i.e., from single units of resources to thousands of units of resources. Then one would modify the model as follows:

$$
\begin{aligned}
\text{Max} \quad & c_1\ X_1 \quad + \quad c_2\ X_2 \\
\text{s.t.} \quad & a_{11}\ X_1 \quad + \quad a_{12}\ X_2 \ \le\ b_1 \ /\ SH \\
& a_{21}\ X_1 \quad + \quad a_{22}\ X_2 \ \le\ b_2 \ /\ SH \\
& \quad X_1, \qquad\qquad X_2 \ \ge\ 0
\end{aligned}
$$

One may also scale the objective function coefficients by dividing every objective function coefficient through by a uniform constant (SO).

Scaling of the form above alters the solution. McCarl and Spreen derive a set of equations relating the solutions before and after scaling. A summary of the effects of scaling is now in order. Consider the LP problem

$$\begin{aligned} \text{Max} \quad & CX \\ \text{s.t.} \quad & AX \leq b \\ & X \geq 0 \end{aligned}$$

Now let us apply a set of positive scaling factors. The scaling factors are: a) $COLSCAL_j$ for the j^{th} variable - a factor multiplying every coefficient under that variable; b) $ROWSCAL_i$ for the i^{th} constraint - a factor dividing every coefficient in that constraint; c) OBJSCAL for the objective function - a factor dividing every coefficient in the objective row; and d) RHSSCAL for the right hand side - a factor dividing every right hand side value. The parameters of the model after scaling are:

$$c_j' = c_j * \frac{COLSCAL_j}{OBJSCAL}$$

$$a_{ij}' = a_{ij} * \frac{COLSCAL_j}{ROWSCAL_i}$$

$$b_i' = b_i * \frac{1}{ROWSCAL_i * RHSSCAL}$$

In turn after the scaling is completed the effect on the solution is presented in Table 10.6. This shows for example that if we were to just scale the right hand sides and leave all other factors

alone we would reduce the objective function and all solution variable values in effect dividing them by the right hand side scaling factor.

10.3.3 An Example of Scaling

Suppose we adopt a relatively simple example to illustrate scaling. In particular consider the example below where the numbers ranger from a maximum absolute value of 50,000 to a minimum of one. We will try to reduce this disparity. However we should note that this problem is already adequately scaled for any GAMS based solver, so this is just a numerical illustration of how scaling proceeds, not a necessary exercise to make the solvers work. But if a much larger problem had such numerical disparities problems could arise.

$$
\begin{array}{llrcrcrcrcr}
\text{Max} & 10X_1 & - & 5000X_2 & - & 4000X_3 & - & 50000X_4 & & \\
\text{s.t.} & X_1 & - & 10000X_2 & - & 8000X_3 & & & \leq & 0 \\
 & & & 5X_2 & + & 4X_3 & - & 50X_4 & \leq & 0 \\
 & & & 1500X_2 & + & 2000X_3 & & & \leq & 6000 \\
 & & & 50X_2 & + & 45X_3 & & & \leq & 300 \\
 & X_1, & & X_2, & & X_3, & & X_4 & \geq & 0
\end{array}
$$

Suppose we begin by dividing all coefficients in the first constraint equation by 10000 and multiplying all coefficients under X_1 by 10000 while also dividing all coefficients in the third constraint equation by 1000 and in the fourth by 50. The resultant model is

$$
\begin{array}{llrcrcrcrcr}
\text{Max} & 100000X_1 & - & 5000X_2 & - & 4000X_3 & - & 50000X_4 & & \\
\text{s.t.} & X_1 & - & 1X_2 & - & 0.8X_3 & & & \leq & 0 \\
 & & & 5X_2 & + & 4X_3 & - & 50X_4 & \leq & 0 \\
 & & & 1.5X_2 & + & 2X_3 & & & \leq & 6 \\
 & & & X_2 & + & 0.9X_3 & & & \leq & 6 \\
 & X_1, & & X_2, & & X_3, & & X_4 & \geq & 0
\end{array}
$$

Now suppose we finish by dividing all coefficients in the X_4 column by 50 and divide all coefficients in the objective function by 10000. The final scaled problem then becomes

$$
\begin{array}{lrcrcrcrcl}
\text{Max} & 10X_1 & - & 0.5X_2 & - & 0.4X_3 & - & 0.1X_4 & & \\
\text{s.t.} & X_1 & - & X_2 & - & 0.8X_3 & & & \leq & 0 \\
 & & & 5X_2 & + & 4X_3 & - & X_4 & \leq & 0 \\
 & & & 1.5X_2 & + & 2X_3 & & & \leq & 6 \\
 & & & X_2 & + & 0.9X_3 & & & \leq & 6 \\
 & X_1, & & X_2, & & X_3, & & X_4 & \geq & 0
\end{array}
$$

The disparity in numbers is now much less. Note the only way that one does gain in reducing the disparity in the numbers is by simultaneously scaling both variables and equations. Simply dividing the rows through by say the largest coefficient without altering the variables will generally not be effective.

10.3.4 Implementing Scaling in GAMS

GAMS users have two choices, let the solver scale or scale the GAMS model before it is passed to the solver. When scaling the GAMS model one can use the built-in scaling features in GAMS or can manually scale and descale the data. We recommend that solver scaling always be done. In addition we feel GAMS supported, user defined model scaling should be used on numerically difficult problems.

10.3.4.1 Using Solver Scaling

Most of the GAMS solvers scale automatically. The notable exceptions are BDMLP and CONOPT. Solver scaling is transparent to the user. (Users employing GAMSKEEP which retains the scratch directories can find output about solver scaling in the GAMSSOLU.SCR file, but otherwise the solver scales the data before solving and returns a descaled solution). Solver

scaling can usually be suppressed through the solver options file. However, we do not recommend that this be done. Some solvers do contain additional scaling procedures that are activated through the options file. Important optional scaling features are resident in MINOS5. More will be said about this in the nonlinear scaling section below.

10.3.4.2 User Defined Model Scaling

User defined model scaling can be employed to improve over solver based scaling. Solver based scaling is implemented using rules where for example, an equation is divided through by the absolute value of the largest coefficient in that equation. Similarly, variables are divided through by numbers derived from the absolute values of the coefficients in those columns. Such mechanical scaling is done iteratively for a number of passes alternating between variable and equation scaling. Scaling can usually be done in a better fashion by the user. The reason user define scaling is often better is that it reflects knowledge of model structure. For example, in our empirical example above the variable X_1 is a sale variable which disposes of the products produced by X_2 and X_3 as added up in the first equation. Thus to preserve integrity of units, we scale X_1 and the first equation by the same factor. This greatly narrows the disparity of the coefficients in the first equation. Understanding of model structure often leads to simultaneous scaling of multiple rows and columns which can greatly narrow coefficient disparity.

Also, remember within most of the GAMS solvers the mechanically based scaling procedures then will then also be used by the solver so one gets the benefit of double scaling. However, in most cases the user can scale better because of the understanding of the model structure. Also, remember in doing this scaling that it is usually to reduce the absolute value of the coefficients to the neighborhood of unity where we recommend 0.1-100. However, this is not to say that models without more disparate coefficients will not work.

10.3.4.2.1 GAMS User Defined Algebraic Scaling

GAMS provides features which allow one to specify scaling factors which are used internally to scale the matrix and to automatically descale the solution. To invoke scaling one must insert the command

```
modelname.scaleopt=1;
```

where the modelname is that identified in the model and solve statements. Then for the variables and equations to be scaled one must insert statements of the form

```
varname.scale(setelements)=k;
eqnname.scale(setelements)=k;
```

where varname is the name of one of the problem variables; eqnname is the name of one of the problem equations; setelements is the associated set elements; and k is any number or GAMS data item name.

(This scaling results in all the coefficients associated with a variable being multiplied by the scaling factor or equation being divided by the scaling factor.) GAMS automatically descales all solution information so the scaling
does not affect the solution output. However the LIMROW/LIMCOL output from GAMS does display the elements after scaling and an option in GAMSCHK controls whether the data displayed are before or after scaling.

Table 10.7 illustrates the scaling procedure in the context of the example we have been using earlier in the chapter. The active statements used for the scaling are in lines 23 - 30 and are reproduced below.

```
scalemod.scaleopt=1;
obj.scale=10000;
z.scale=obj.scale;
avail.scale("r1")=10000;
```

```
x.scale("x1")= avail.scale("r1");
avail.scale("r3")=1000;
avail.scale("r4")=50;
x.scale("x4")=1/50;
```

The only additional comment about scaling that is in order involves the objective function. In particular, note we scale the objective function row and variable in a consistent fashion.

More generally in a model with variables named PRODUCTION, and SALES with equations named RESOURCES and PRODBAL one could introduce the statements

```
SCALFACTOR(ITEMS)=50;
SCALFACTOR("CARS")=100;
PRODUCTION.SCALE (ITEMS)    =       1000;
SALES.SCALE (PRODUCTS       =       2000;
RESOURCES.SCALE (TYPES,OTHERSET)      =       SCALFACTOR(ITEMS);
PRODBAL.SCALE ("CARS")      =        50;
```

much as in the manner of defining upper or lower bounds.

10.3.4.2.2 Manual Scaling in a Model

One does not always need to rely on GAMS to do the scaling, although we recommend it. Table 10.8 presents an example where scaling is done manually by introducing parameters giving the scaling factors for the variables and equations on a term by term basis into the GAMS algebraic statements for model variables and equations. Note, however, when employing manual scaling that the solution is distorted and reverse scaling must be done to correct it. Thus, we do not recommend manual scaling but rather use GAMS based scaling. That only leaves us without a mechanism to do RHS scaling (dividing all equation constants by a factor) which is rarely needed. In addition if RHS scaling is employed one must either remember to descale all solution variable levels including the objective function value by multiplying them by the right hand side scaling factor or must remember to interpret the solution levels in a manner where they are all in say 1000's of units (where the RHS scale was 1000).

10.3.5 How are Scaling Factors Determined

The essential question when one considers scaling is which variables and equations should be scaled and how much. This is obviously a numerical, empirical problem specific issue depending on the model at hand and the exact coefficients. The basic methodology for determining how much to scale is as follows:

Step 1 - Examine the empirical model first on a block by block basis then later on a variable by variable and equation by equation basis and discover poorly scaled model elements. Namely, identify those variables and/or constraints which have the largest and smallest coefficients in absolute value. If these far in excess of one and far less than one then scaling is in order. Ways of gathering this information with GAMS and GAMSCHK is discussed below.

Step 2 - Develop a set of scaling factors. For example, when all coefficients under a variable or in an equation are all in excess of an absolute value of one, then set up a scaling factor to divide all coefficients under the variable or in the equation.

Step 3 Implement the scaling factors in GAMS then go back to step 1; but scale the equations next and iteratively work over the variables and rows until the coefficients converge in value.

Note, that in using scaling factors one will often be dividing through by numbers like 10,000 and also could be dividing through like numbers like 0.0001 to raise the absolute value of the set of coefficients.

An essential step in the above procedure is numerically investigating the model to find the appropriate magnitude of scaling factors. This numerical investigation means one needs to look at the coefficients in the specific equations. This may be done either using the GAMS LIMROW/LIMCOL options or the GAMSCHK MATCHIT, BLOCKPIC, BLOCKLIST, PICTURE, or DISPLAYCR procedures. All of these procedures display the model after the actions of scale parameters defined in GAMS have been applied (provided the DESCALE option has not been implemented in GAMSCHK). MATCHIT, PICTURE and DISPLAYCR work on individual variables and equations while BLOCKPIC and BLOCKLIST work on blocks. An example, all of these applied to the unscaled example used in the last chapter appears in Table 10.9. There the LIMROW and LIMCOL listing in Panel A; the MATCHIT listing in Panel B which shows the maximum/minimum absolute values for each equation, the DISPLAYCR in Panel C and then the BLOCKLIST and BLOCKPIC outputs in Panel D and E. The recommended procedure for using GAMSCHK is:

Step 1 Fix Block level scaling. Use BLOCKLIST and BLOCKPIC to find the maximum or minimum absolute values of the coefficients in the blocks. If any variable or equation blocks exhibit all coefficients significantly above or below one. If so, scale the whole block to change the absolute maximum or minimum to something close to one. Do the same to equations. Also consider whether simultaneous variable and equation block scaling can reduce coefficient magnitude disparity. Continue iteratively until all the block level phenomena have been worked out.

Step 2 Fix individual item scaling. Use MATCHIT, DISPLAYCR or PICTURE and look at the size of the coefficients in variable and equations. In using GAMSCHK make sure that DESCALE has not been turned on in the

options file. Note that in LIMROW, LIMCOL displays will contain the coefficients after scaling not the coefficient before. Thus as scaling proceeds one can identify the cumulative effect of all scaling.

By following the steps above one can reduce coefficient disparity so that the problem is numerically easier. Remember that the solver will also try to improve the scaling so what one should try to accomplish when doing user defined scaling is one to exert knowledge of the structure of the model to improve scaling. For example, take an equation which is denominated in the units of a commodity and scale both the sales and purchase activities for that commodities in the same way that the units in the equation are scaled. In our example above the scale factors for OBJ and Z as well as X_1 and R_1 were the same.

10.3.6 Scaling of Nonlinear Terms

Nonlinear terms are meritorious of special scaling efforts. Drud in his CONOPT implementation pays particular attention to scaling diagnostics. His recommendation is that the gradients of all nonlinear terms be scaled so that their values are one at optimality. This implies several things.

a) Scaling should be done so that the absolute value of the nonlinear terms as reported back by GAMS and GAMSCHK are close to one, much as in the case with linear terms. However attention also needs to be paid to the gradients.

b) Since the numerical value of the nonlinear terms reported by GAMS and GAMSCHK are Taylor series expansions of the nonlinear terms around the starting point then one needs to use a starting point as close to the optimal solution point as possible.

c) Scaling in the nonlinear solvers involves special considerations. MINOS5 has

special options controlling whether nonlinear scaling is done. CONOPT does not scale internally frequently rejects problems because of poor scaling characteristics. Further user defined scaling takes on a bigger role as these authors have worked with cases where for example MINOS5 would not solve the model unless user defined scaling was done. It's own internal scaling only resulted in a model which would not solve in a reasonable time period.

10.4 A Priori Degeneracy Resolution

Solution processes can exhibit a lack of progress due to degenerate cycling. Solvers like MINOS5 on occasion give messages like "terminating since no progress made in last 1000 iterations" or "Sorry fellows we seem to be stuck.". Such cycling can often be avoided through three types of fixes. First, one should insure that the scaling and, in nonlinear cases, starting point characteristics of the problem are good. Second, some solvers like OSL perturb the problem during the solution process to automatically avoid cycling, and one should make sure this is permitted (it is the default case). Third, one can priori perturb the problem. Here we treat the third case.

Degenerate cycling is caused when many pivots are done in the solution process where the entering variable is essentially zero and where solvers continually add and remove the same sequence of variables. Over the years we have found that such situations may be partially avoided by insuring that equation blocks with right-hand-sides that are ordinarily zero actually have small nonzero right hand sides (i.e., .001). However having them all have the same value can also be problematic so we add numbers randomly centered around .001 to the right-hand-sides in order to aid avoiding degenerate pivots (see McCarl,1977 for discussion). For example given the equation

Balance(commodity)..-supply(commodity) + Demand(commodity)=L=O;

we would change the problem to

Balance(commodity)..-Supply(commodity) + DEMAND(commodity) =L=0.0001;

The magnitude of the small numbers should be specified so that they are not the same for all rows and so that they do not materially affect the solution. Thus, they might be random or systematically chosen numbers of the order 10^{-3} or 10^{-4} (although they can be larger or smaller depending on the scaling and purpose of the constraints as in McCarl, 1977). We have always observed reduced solution times with this modification. On occasion we have also found it necessary to vary the value of nonzero right hand sides in an analogous fashion. In particular if a model has a lot of constraints that are all less than or equal to a number like ten then we have been known to add small random constants across the equations.

10.5 Reformulating a Model

Models can fail to solve running out of time, iterations, or making insufficient progress because the problem is too difficult for the solver. One possible repair in such a case is simplification of the optimization model formulation. This can be done by deleting constraints, converting constraints to bounds, linearizing difficult nonlinear programs, adding constraints to eliminate integer solutions, tightening numerical values in constraints relating integer and continuous variables and fixing the values of integer variables, among many other things. Such structural modifications are beyond the scope of these notes. Suffice it to say Luenberger; Williams; McCarl and Spreen; and many others address problem reformulation. Also note the presolves in OSL and CPLEX automatically do some of these functions.

10.6 Using Solver Options

One other way of speeding up solver performance particularly in the case of MIP and NLP problems involves the use of solver options. The adoption of STRATEGY 48 and BBPREPROC in OSL often is useful in dealing with MIPs while choice of the interior point method use may help with large LPs. The use of the interior point and branching strategies such as varsel 3 or heuristic 1 can be useful in CPLEX as can setting heurfreq to a fairly small number about 10 and nodesel 2 or another value. Generally it is best to run a MIP solver with the default options and a small node limit to see if the MIP seems to be solving and then if not try the alternative options. If the MIP finds a solution quickly then one should try to prune the tree as fast as possible. If not The scaling procedures can improve performance in MINOS.

Table 10.1 Example Model for Memory Discussion

```
 1  option profile=1;
 2  option profiletol=0.01
 4  option limrow=0;
 5  option limcol=0;
 6  set I /1*3 /
 7  set j /1*3 /
 8  set k /1*3 /
 9  set l /1*3 /
10  set m /1*3 /
11  set n /1*3 /
12  set o /1*3 /
13  parameter y(i,j,k,l,m,n,o);
14  parameter q(i,j,k);
15  variable  x(i,j,k,l,m,n,o)
16            f(i,j,k)
17            obj;
18  equation  z(i,j,k,l,m,n,o)
19            res(i,j,k)
20            ob;
21   y(i,j,k,l,m,n,o)=10;
22   x.up(i,j,k,l,m,n,o)=10;
23   x.scale(i,j,k,l,m,n,o)=1000;
24   q(i,j,k)=10;
25
26  ob..
27        obj  =e= suM((i,j,k,l,m,n,o),x(i,j,k,l,m,n,o));
28  z(i,j,k,l,m,n,o)..
29        x(i,j,k,l,m,n,o) =l= 8;
30  res(i,j,k)..
31            f(i,j,k)=l=7;
32  model memory /all/
33  *option dmpsym;
34  *option memorystat;
35  *option sys7=2;
36  option solprint=off;
37  solve memory maximizing obj using lp;
```

Table 10.2 Example of Symbol Table

ENTRY	ID	TYPE	DIM	LENGTH	DEFINED	ASSIGNED	DATAKNOWN
1	MAPVAL	FUNCT	0	0	False	False	False
2	CEIL	FUNCT	0	0	False	False	False
3	FLOOR	FUNCT	0	0	False	False	False
4	ROUND	FUNCT	0	0	False	False	False
5	MOD	FUNCT	0	0	False	False	False
6	TRUNC	FUNCT	0	0	False	False	False
7	SIGN	FUNCT	0	0	False	False	False
8	MIN	FUNCT	0	0	False	False	False
9	MAX	FUNCT	0	0	False	False	False
10	SQR	FUNCT	0	0	False	False	False
11	EXP	FUNCT	0	0	False	False	False
12	LOG	FUNCT	0	0	False	False	False
13	LOG10	FUNCT	0	0	False	False	False
14	SQRT	FUNCT	0	0	False	False	False
15	ABS	FUNCT	0	0	False	False	False
16	COS	FUNCT	0	0	False	False	False
17	SIN	FUNCT	0	0	False	False	False
18	ARCTAN	FUNCT	0	0	False	False	False
19	ERRORF	FUNCT	0	0	False	False	False
20	UNIFORM	FUNCT	0	0	False	False	False
21	NORMAL	FUNCT	0	0	False	False	False
22	POWER	FUNCT	0	0	False	False	False
23	JDATE	FUNCT	0	0	False	False	False
24	JTIME	FUNCT	0	0	False	False	False
25	JSTART	FUNCT	0	0	False	False	False
26	JNOW	FUNCT	0	0	False	False	False
27	EXECERROR	FUNCT	0	0	False	False	False
28	GYEAR	FUNCT	0	0	False	False	False
29	GMONTH	FUNCT	0	0	False	False	False
30	GDAY	FUNCT	0	0	False	False	False
31	GDOW	FUNCT	0	0	False	False	False
32	GLEAP	FUNCT	0	0	False	False	False
33	GHOUR	FUNCT	0	0	False	False	False
34	GMINUTE	FUNCT	0	0	False	False	False
35	GSECOND	FUNCT	0	0	False	False	False
36	EXECSEED	FUNCT	0	0	False	False	False
37	TIMESTART	FUNCT	0	0	False	False	False
38	TIMECOMP	FUNCT	0	0	False	False	False
39	TIMEEXEC	FUNCT	0	0	False	False	False
40	TIMECLOSE	FUNCT	0	0	False	False	False
41	**********	FUNCT	0	0	False	False	False
42	FILE	FILE	0	0	False	False	False
43	I	SET	1	3	True	False	True
44	J	SET	1	3	True	False	True
45	K	SET	1	3	True	False	True
46	L	SET	1	3	True	False	True
47	M	SET	1	3	True	False	True
48	N	SET	1	3	True	False	True
49	O	SET	1	3	True	False	True
50	Y	PARAM	7	2187	False	True	False
51	Q	PARAM	3	27	False	True	False
52	X	VAR	7	2187	False	True	False
53	F	VAR	3	0	False	True	False
54	OBJ	VAR	0	0	False	True	False
55	Z	EQU	7	0	False	True	False
56	RES	EQU	3	0	False	True	False
57	OB	EQU	0	0	False	True	False
58	MEMORY	MODEL	0	3	True	True	True

Table 10.3 Example of Profile Output

```
----       3 OTHER                   0.000     0.010 SECONDS
----       4 OTHER                   0.000     0.010 SECONDS
----      21 ASSIGNMENT Y            0.080     0.090 SECONDS     2187
----      22 ASSIGNMENT X            0.070     0.160 SECONDS     2187
----      23 ASSIGNMENT X            0.050     0.210 SECONDS     2187
----      24 ASSIGNMENT Q            0.000     0.210 SECONDS       27
----      36 OTHER                   0.000     0.210 SECONDS
----      37 SOLVE INIT MEMORY       0.000     0.220 SECONDS
----      28 EQUATION   Z            0.490     0.710 SECONDS     2187
----      30 EQUATION   RES          0.000     0.710 SECONDS       27
----      26 EQUATION   OB           0.370     1.080 SECONDS        1
----      37 SOLVE FINI MEMORY       0.310     1.390 SECONDS
```

Table 10.5 Output Under Sys7=1

```
----     28 MEMORY MANAGEMENT STATISTICS

REC  POOL  SIZE  NAME          ACTIVE     MAXACT       NEWS      DISPS
  1     1     8  SYMBHSH           58         58         58          0
  2     1     8  UELHSH             3          3          3          0
  3     1     8  JACREF             0          0          0          0
  4     1     8  PARREC          2214       2214       2214          0
  5     2    12  SYMBREF            0          0          0          0
  6     2    12  UELREF             0          0          0          0
  7     4    20  HDCREC          2187       2187       2187          0
  8     2    12  PD                 0       2187       2187       2187
  9     3    16  CEL            13179      25216      28488      15309
 10     3    16  HDRREC          2187       2187       2187          0
 11     4    24  JACREC          2187       2187       2187          0
 12     4    24  COFREC             0          0          0          0
 13     5    32  RHS                0       2187       2187       2187
 14     4    24  UEL                3          3          3          0
 15     6    40  VARREC          2197       2197       2197          0
 TOTAL                          24215                 43898      19683
BLOCK SIZE = 4092   BLOCKS ALLOCATED = 177   TOTAL POOL ALLOC = 708K

----     30 MEMORY MANAGEMENT STATISTICS

REC  POOL  SIZE  NAME          ACTIVE     MAXACT       NEWS      DISPS
  1     1     8  SYMBHSH           58         58         58          0
  2     1     8  UELHSH             3          3          3          0
  3     1     8  JACREF             0          0          0          0
  4     1     8  PARREC          2214       2214       2214          0
  5     2    12  SYMBREF            0          0          0          0
  6     2    12  UELREF             0          0          0          0
  7     4    20  HDCREC          2214       2214       2214          0
  8     2    12  PD                 0       2187       2214       2214
  9     3    16  CEL            13257      25216      28647      15390
 10     3    16  HDRREC          2214       2214       2214          0
 11     4    24  JACREC          2214       2214       2214          0
 12     4    24  COFREC             0          0          0          0
 13     5    32  RHS                0       2187       2214       2214
 14     4    24  UEL                3          3          3          0
 15     6    40  VARREC          2197       2197       2197          0
 TOTAL                          24374                 44192      19818

BLOCK SIZE = 4092   BLOCKS ALLOCATED = 178   TOTAL POOL ALLOC = 712K

----     26 MEMORY MANAGEMENT STATISTICS

REC  POOL  SIZE  NAME          ACTIVE     MAXACT       NEWS      DISPS
  1     1     8  SYMBHSH           58         58         58          0
  2     1     8  UELHSH             3          3          3          0
  3     1     8  JACREF             0          0          0          0
  4     1     8  PARREC          2214       2214       2214          0
  5     2    12  SYMBREF            0          0          0          0
  6     2    12  UELREF             0          0          0          0
  7     4    20  HDCREC          2215       2215       2215          0
  8     2    12  PD                 0       2187       4402       4402
  9     3    16  CEL            13259      25216      47231      33972
 10     3    16  HDRREC          2215       2215       2215          0
 11     4    24  JACREC          4402       4402       4402          0
 12     4    24  COFREC             0          0          0          0
 13     5    32  RHS                0       2187       2215       2215
 14     4    24  UEL                3          3          3          0
 15     6    40  VARREC          2197       2197       2197          0
 TOTAL                          26566                 67155      40589
BLOCK SIZE = 4092   BLOCKS ALLOCATED = 190   TOTAL POOL ALLOC = 760K
```

Table 10.6 Relationships Between Items Before and After Scaling

Item	Symbol Before Scaling	Symbol After Scaling	Unscaled Value in Terms of Scaled Value	Scaled Value in Terms of Unscaled Value
Variables	X_j	X_j'	$X_j = X_j' * (COLSCAL_j * RHSSCAL)$	$X_j' = X_j / (COLSCAL_j * RHSSCAL)$
Slacks	S_i	S_i'	$S_i = S_i' * (ROWSCAL_i * RHSSCAL)$	$S_i' = S_i / (ROWSCAL_i * RHSSCAL)$
Reduced Cost	$z_j - c_j$	$z_j' - c_j'$	$z_j - c_j = (z_j' - c_j') * (OBJSCAL/COLSCAL_j)$	$z_j' - c_j' = (z_j - c_j) / (OBJSCAL/COLSCAL_j)$
Shadow Price	U_i	U_i'	$U_i = U_i' * (OBJSCAL/ROWSCAL_i))$	$U_i' = U_i / (OBJSCAL/ROWSCAL_i))$
Obj. Func. Value	Z	Z '	Z = Z' * OBJSCAL * RHSSCAL	Z '= Z / (OBJSCAL * RHSSCAL)

Table 10.7 Example of GAMS Automatic Scaling

```
panel a Model including scaling factors

 1  sets  items        names of variables   /x1*x4/
 2        resources    names of constraints /r1*r4/
 3  parameter
 4   objc(items)    obj coefs /x1 10,x2 -5000,x3 -4000,x4 -50000/
 5   rhs(resources)resource availabilities /r3 6000,r4 300/;
 6  Table amatrix(resources,items) aij matrix
 7      x1      x2     x3     x4
 8    r1    1   -10000  -8000
 9    r2         5     4     -50
10    r3         1500   2000
11    r4          50     45          ;
12  variables          z objective      function;
13  positive variables x(items)      variables;
14  equations          obj           objective function
15                     avail(resources) resource limits;
16  obj.. z=e=sum(items,objc(items)*x(items));
17  avail(resources).. sum(items,amatrix(resources,items) *x(items))
18                       =l= rhs(resources);
19  option limrow=4; option limcol=0;
20  model scalemod /all/;
21  option solslack=1;
22  solve scalemod using lp maximizing z;
23  scalemod.scaleopt=1;
24  obj.scale=10000;
25  z.scale=obj.scale;
26  avail.scale("r1")=10000;
27  x.scale("x1")= avail.scale("r1");
28  avail.scale("r3")=1000;
29  avail.scale("r4")=50;
30  x.scale("x4")=1/50;
31  solve scalemod using lp maximizing z;
```

Panel b formuation before scaling from solve at line 22

```
OBJ..  Z - 10*X(X1) + 5000*X(X2) + 4000*X(X3) + 50000*X(X4) =E= 0 ;
AVAIL(R1)..  X(X1) - 10000*X(X2) - 8000*X(X3) =L= 0 ; (LHS = 0)
AVAIL(R2)..  5*X(X2) + 4*X(X3) - 50*X(X4) =L= 0 ; (LHS = 0)
AVAIL(R3)..  1500*X(X2) + 2000*X(X3) =L= 6000 ; (LHS = 0)
AVAIL(R4)..  50*X(X2) + 45*X(X3) =L= 300 ; (LHS = 0)
```

Panel c formulation after scaling from solve at line 31

```
OBJ..  Z - 10*X(X1) + 0.5*X(X2) + 0.4*X(X3) + 0.1*X(X4) =E= 0 ;
AVAIL(R1)..  X(X1) - X(X2) - 0.8*X(X3) =L= 0 ; (LHS = 0)
AVAIL(R2)..  5*X(X2) + 4*X(X3) - X(X4) =L= 0 ; (LHS = 0)
AVAIL(R3)..  1.5*X(X2) + 2*X(X3) =L= 6 ; (LHS = 0)
AVAIL(R4)..  X(X2) + 0.9*X(X3) =L= 6 ; (LHS = 0)
```

Panel d Solution under either solve

```
        LOWER     SLACK     UPPER    MARGINAL
R1       -INF       .         .       10.000
R2       -INF       .     1000.000
R3       -INF       .     6000.000    60.000
R4       -INF    100.000   300.000       .

        LOWER     LEVEL     UPPER    MARGINAL
Z        -INF  3.6000E+5     +INF       .
X1         .    40000.000     +INF       .
X2         .        4.000     +INF       .
X3         .         .        +INF  -4.800E+4
X4         .        0.400     +INF       .
```

Table 10.8 Example of Manual Scaling in GAMS

Panel a GAMS Setup

```
    1    sets      items           names of variables   /x1*x4/
    2              resources       names of constraints /r1*r4/
    4    parameter objcoef(items)   objective fun   /x1 1, x2 -500, x3 -400, x4 -5000/
    6              rhs(resources)   resource availabilities   /r3 6000,r4 300/;
    9    Table amatrix(resources,items) aij matrix
   10           x1      x2     x3     x4
   11      r1    1    -1000  -8000
   12      r2           5     4     -50
   13      r3          1500   2000
   14      r4           50     45         ;
   16   Parameter varscale(items) variable scaling factors / x1 1, x2 .001 , x3 .001 , x4 .0002 /
   18   Parameter eqnscale(resources) equation scaling factors /r1 1, r2 .001 , r3 1 , r4 .01 /
   20   Scalar zscale /1/
   21    variables          z objective      function;
   22    positive variables xvar(items)      variables;
   23    equations          objfun           objective function
   24                       avail(resources) resource limits;
   26    objfun..   z=e=sum(items,objcoef(items)*varscale(items)/zscale*xvar(items));
   28    avail(resources).. sum(items,amatrix(resources,items)
   29                        *varscale(items)/eqnscale(resources)*xvar(items))
   30                             =l=rhs(resources)/eqnscale(resources);
   33    model scalemod /all/;
   34    solve scalemod using lp maximizing z;
   35    parameter unscalex(items,*), unscaleqn(resources,*;
   36    unscalex(items,"level")=xvar.l(items)*varscale(items);
   37    unscalex(items,"marg") =xvar.m(items)/varscale(items);
   40    unscaleqn(resources,"level")=avail.l(resources)*eqnscale(resources);
   41    unscaleqn(resources,"marg")=avail.m(resources)/eqnscale(resources);
   42    display unscalex,unscaleqn;
```

Panel B Row listing under manual scaling

```
---- AVAIL       =L=  resource limits
AVAIL(R1)..  XVAR(X1) - XVAR(X2) - 8*XVAR(X3) =L= 0 ; (LHS = 0)
AVAIL(R2)..  5*XVAR(X2) + 4*XVAR(X3) - 10*XVAR(X4) =L= 0 ; (LHS = 0)
AVAIL(R3)..  1.5*XVAR(X2) + 2*XVAR(X3) =L= 6000 ; (LHS = 0)
AVAIL(R4)..  5*XVAR(X2) + 4.5*XVAR(X3) =L= 30000 ; (LHS = 0)
```

Panel C Solution under Manual Scaling

```
---- EQU OBJFUN          .         .         .        1.000
---- EQU AVAIL       resource limits
      LOWER     LEVEL     UPPER    MARGINAL
R1     -INF       .         .        1.000
R2     -INF       .         .        0.100
R3     -INF   6000.000  6000.000     3.600
R4     -INF  13500.000 30000.000      .

                      LOWER     LEVEL     UPPER    MARGINAL
---- VAR Z              -INF  21600.000     +INF       .
---- VAR XVAR        variables
      LOWER     LEVEL     UPPER    MARGINAL
X1      .    24000.000     +INF       .
X2      .         .        +INF     -5.400
X3      .     3000.000     +INF       .
X4      .     1200.000     +INF       .
```

Panel D Transformed Scaled Solution

```
----     42 PARAMETER UNSCALEX
         LEVEL        MARG
X1   24000.000
X2               -5400.000
X3       3.000
X4       0.240

----     42 PARAMETER UNSCALEQN
         LEVEL        MARG
R1                   1.000
R2                 100.000
R3    6000.000       3.600
R4     135.000
```

Chapter 11 Working with Advanced Bases

The importance of an advanced basis has long been recognized in mathematical programming. Models which take more than 24 hours to solve from scratch can be solved in less than an hour from a good basis. GAMS can accept an advanced basis under certain conditions. Generally, three possibilities are available with respect to bases. One can solve:

1) without a basis;

2) from a saved basis file; and

3) from a user suggested basis.

In small problems one should always solve without a basis. Before discussing the latter two alternatives, let us review how GAMS forms a basis.

11.1 How Does GAMS Form a Basis?

GAMS forms a basis from stored marginal and level values for the model variables and equations. This information comes from a previous solution or a user suggested starting point. The information used in basis formation are:

a) Non zero levels for variables in the basis or super basis (for nonlinear problems). These levels are generally not equal to the upper or lower bounds for the variable;

b) Non zero values equal to the upper or lower bounds for variables which are held 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.

In this information degeneracies and alternative optimals are indicated by some of the marginals for

nonbasic variables and/or constraints equaling EPS.

GAMS uses these data to suggest a basis. The exact form of the basis depends upon the solver. For example, super basics are only suggested to nonlinear solvers. Similarly, bases suggested to LP solvers consist of a number of basic elements no greater than the number of equations.

11.2 Using an Advanced Basis

So now how does one cause an advanced basis to be used? The answer is that one must provide the solution information as listed above. There are several ways of doing this. The easiest case involves models solved in a series. In any procedure involving a sequence of solves, GAMS automatically uses the solution from the most recent solve to suggest a basis for subsequent solves. Obviously, this does not apply to the first solve in the GAMS program.

Particularly for first solves, but also in some other cases, one can suggest an advanced basis. This is done by setting level and marginal values for the variables and marginal values for the equations. This can be done by either using a stored solution from elsewhere or creating one. Each of these possibilities and the repeated solve option will be discussed below.

11.2.1 Forming a Basis from Repeated Solves

One way that assures an advanced basis is used involves repeated solves of a model. The fundamental method is to take a model like that in Chapter 8, then solve it for the base case, revise some data and solve it again. Suppose we take the model from Table 8.1 and wish to solve it twice, once for base conditions and once again doubling the transportation costs. We introduce two statements at the bottom of the model where the first statement doubles the transportation cost and the second causes another solve (see the file TWOTRAN). In this particular case, using

OSL one obtains the second solution in 1 iteration as opposed to the original cold start solve which took 19 iterations.

Yet another way of retaining the basis involves a stored "base case" model. Frequently modelers set up a "base case" model and then do alternative runs to see how the solution changes. In this case, one can gain efficiencies by using the advanced basis from the base case model. To do this we use the GAMS save and restart file commands. First, the base model is solved and its solution saved. In turn, when GAMS restarts and solves, it uses the base solution to form the next basis. Suppose we two files

FRSTPART which sets up a model and performs the first solve; and

NEXTPART which manipulates data and performs a second solve.

In such a case we would use a PC job control string like the following:

```
GAMS FRSTPART  S=F1
GAMS NEXTPART R=F1
```

This would result in the FRSTPART file being executed and then execution restarted as if NEXTPART was grafted on to the bottom of FRSTPART using all information available at the end of its execution including the basis. Through this mechanism an advanced basis can be saved.

One could also use the commands:

```
GAMS FRSTPART   S=Savefilename
GAMS MODIFY1    R=Savefilename
GAMS MODIFY2    R=Savefilename
```

which would result in the two modification files each being started utilizing the base model basis.

11.2.2 Providing a Basis to a Independent Model

Unfortunately, the above strategy can not be used when one needs to solve a problem from scratch. However, one can suggest a basis for a new model if one is careful. Let us use an example with two constraints and no upper bounds to illustrate this

$$
\begin{array}{lrcrcrcl}
\text{Maximize} & 30X_1 & + & 20X_2 & + & 10X_3 & & \\
& X_1 & + & X_2 & + & X_3 & \leq & 10 \\
& X_1 & - & X_2 & - & 1.5X_3 & \leq & 2 \\
& & & X_2 & + & X_3 & \leq & 8 \\
& X_1, & & X_2, & & X_3 & \geq & 0
\end{array}
$$

A GAMS implementation appears in Table 11.1, Panel A. Two alternative basis structures appear in Panels B and C either of which would compose the file simple.bas and would be included in line 39. In Panel B we define a version of the file simple.bas which contains exact numerical values for the variables and marginals. The binding constraints are those with non-zero marginals (i.e. the objective function, resource 1 and resource 2). The nonbinding constraints have zero marginals (Resource 3). The basic variables have nonzero levels while the nonbasic variable (X3) has a nonzero marginal. This input causes GAMS to suggest a basis with basic variables X1, X2 and the slack for the third resource constraint. Solving the model with this basis results in an immediate solution without any iterations.

An equivalent way of entering this basis through the simple.bas file is given in Panel C where the nonzero data entries are just ones (one must insure the ones fall between the lower and upper bounds). Both of these basis suggestion approaches result in solutions which take zero iterations. However, the second approach needs to be used with more care. In particular, the second approach cannot be easily used when dealing with NLP's as super basics need to take on

precise values. The above approach can be used to either totally or partially specify a basis, a full basis with all the proper nonzeros does not need to be specified. When specifying a full basis degeneracy can cause problems as one must have basic elements for each row and must be careful to have EPS specified for degenerate basic variables, non basic alternative optimals and for the marginals of constraints with degenerate slacks.

11.2.3 Through External Files from Related Models

One can obtain a basis by using one from a related model. These may be saved either through GAMSBAS or by using some mixture of PUT files and or displays in conjunction with text editing.

GAMSBAS saves the level and marginal values after a solution. An example of its use is provided in Table 11.2 (see the title MAKEBAS). The file invokes GAMSBAS in lines 38 and 41. This constructs the saved basis file MAKEBAS.BAS which is presented in Table 11.3. That file contains the solution information in the format of GAMS replacement statements. In turn, we include that file before subsequent solves. This is done by inserting an include statement as shown in line 35 of Table 11.4 or file USEBAS example.

The consequences of these actions are that the restarted problem solves in zero iterations. Inclusion of such a basis has proven to be useful when we modify the structure or data of a model, but feel that the basis from an earlier solved model would be a good starting point.

One may also retain a basis using other means. For example, before developing GAMSBAS we at times displayed solution information and then reincluded it using a text editor and replacement statements. We also used GAMS PUT file mechanisms to save information in the appropriate format. We now recommend using GAMSBAS, however because the other forms

require model specific code and give difficulties when degeneracies and alternative optimals are present.

11.2.4 By Guessing at a Starting Point

One can provide partial information to a heretofore unsolved model by guessing at which variables will be in the basis and constraints will be binding. In turn, one can suggest a basis by specifying appropriate values for the variable marginals and levels and the equation marginals. There are strategies for constructing such guesses (for example, see Dillon).

11.3 Dealing with Problematic Bases

Unfortunately use of a basis can be a doubled edged sword. GAMS or a GAMS user can suggest a basis that is unworkable in the solver. Solvers such as MINOS can reject a basis indicating that after several factorizations (attempts at forming a basis) a nonsingular basis cannot be formed. Similarly when using OSL, one may find that the algorithm terminates, reporting that it can't make acceptable progress. There are several things one can do when such a problem arises.

In our experience the difficulty usually arises when one uses saved basis from a model which is significantly different in structure from the model to be solved. This occurs when variables and/or equations in the model have been eliminated. Consequently, a number of equations and variables integral to the basis are gone. What happens in such a case is there are excess or insufficient basic variables specified for the number of equations and GAMS cannot suggest a nonsingular basis. This can happen when variables that were basic in the eliminated constraints are still present with non zero levels. In such a case GAMS draws out basic variables in the order they appear until the number of basic variables suggested equals to the number of equations arbitrarily dropping some possibly necessary variables. In such a case one could: a) use

the BRATIO option to cause GAMS to drop the basis; b) structure the model differently; c) reset the basis; or d) try to do basis maintenance. Each of these options will be discussed below. Note these options should not be used unless one is having problems. Usually GAMS can form an adequate basis regardless of the model modifications.

11.3.1 Using BRATIO to suppress a basis

One can cause GAMS to ignore the advanced basis. GAMS includes an option called BRATIO which, "causes the basis to be rejected if the number of basic variables is smaller than BRATIO times the size of the basis." If one is having difficulty with a basis, one can force it to be discarded by setting the BRATIO value to one. The format of this command is:

```
OPTION BRATIO = 1;
```

This causes the advanced basis to be rejected and the solver to start from a cold start. Repeated cold starts can be quite time consuming, therefore we recommend one try ideas in the subsequent sections first.

11.3.2 Resetting the Basis

When one radically alters the model between solves and has a basis induced failure, one could, under an assumption that the base model provides a good basis, revert to the "base model" basis every time. Consider the following example. Suppose we wish to solve a model with and without certain constraints on outgoing transportation. Suppose the base model does not have extra transportation configuration constraints, but that each alternative model does. In such cases one could run into problems with the basis. Namely, suppose one solves a model with one set of configuration constraints, then drops those constraints and adds another very different constraint set. Subsequently, the basis may not be adequate and one could try to go back to the base

unconstrained model. In order to do this the base case solution levels and marginals would need to be retained in temporary data storage then reloaded.

Suppose we illustrate this point in the context of our Chapter 8 model. Let us introduce two new constraints in a modified model (Table 11.5 or the file RESET). These are identified in rows 88 and 89 and specified in lines 115-118. Three models are now defined with the model in line 119 being the base model, the model in line 120 including the first optional constraint, and the one in 120 including the second constraint. In turn, in line 126 we solve the base model then we save the solution information retaining the variable levels and marginals and the equation marginals. This is done using the parameters defined in lines 128-137 and the saves in lines 140-156. In turn we solve the first alternative problem in line 158. Then we reestablish the base model solution in lines 163-180 and solve the second model. Note this procedure is not necessary as the model solutions do not fail when the basis is not manipulated as we can solve the models directly. Nevertheless, using the basis resetting causes solution in 20 as opposed to 26 iterations for the last solve. In more general models, this could lead to time savings and possibly the difference between solution process success and solver failure.

Also note that one could achieve the basis reset using GAMSBAS by forming a basis file and reincluding it. However if the solves and the GAMSBAS file include statement are in a LOOP, one can easily run out of executable code space. This occurs since the replacement statements for GAMSBAS can be numerous and one may have to use larger values of CODEX (see the Chapter 10 discussion of this option). In fact, in some cases one cannot make CODEX large enough to include all the statements. However, the GAMSBAS alternative may be superior since the alternative is to implement storage and replacement statements for every variable and

equation.

11.3.3 Structuring a Formulation to Avoid Basis Problems

Solution of sequences of models with variables and equations being eliminated, restored, or added can place the applicability of the basis in jeopardy. Sometimes such an exercise will cause solver failure. Modelers may wish to consider an alternative management strategy which alleviates these problems.

Rather than adding and removing constraints and variables with $ conditions, one can keep those constraints and variables active in the model throughout the set of comparative runs. but make them unattractive or nonbinding. Namely, if a set of variables is to be considered part, but not all of time, one could (in a maximization context) conditionally subtract a very large number from the objective function. With such an addition the variable takes on the characteristics of an artificial variable. Namely, when it is supposed to be inactive, it is quickly driven from the basis. For example, consider the two alternatives:

```
Objf..   obj = SUM(X, VAR(X)) + Y$ISYHERE;

Objf...  obj = SUM(X, VAR(X)) + (1-1000000 $(ISYHERE  EQ  0))*Y;
```

In the first, Y is eliminated if ISYHERE equals zero, while in the second it takes on an very large negative value if ISYHERE equals zero. In the first case when Y was previously basic and we eliminate it everywhere we now would have an incomplete basis with a variable missing. In the second case if we just have the $ control in the objective function then the Y variable would be present and could be in the initial basis but the optimization process would soon drive it out.

Similarly, if there are constraints that are active under some cases, one could enter a mechanism to relax those constraints. For example, with a less than or equal to constraint one

could add a large number to the right hand side when the constraint is to be inactive. Thus, one could write a constraint as follows.

```
Const$ISCON..  SUM(X,AIJ(X)*VAR(X)) =L= 1;

Const.. SUM(X, AIJ(X)*VAR(X))=L=1 + 1000000$(ISCON  EQ  O);
```

Under these circumstances the constraint would be eliminated in the first case if ISCON equals zero. However, in the second case the constraint is always present but has a very large RHS rendering it ineffective. These strategies maintain the same variables and equations for all bases and avoid singularity problems.

11.3.4 Updating the Basis

One may also attempt to reflect model changes in the basis by manipulating stored levels and marginals. For example, all the levels for a variable could be set to zero and the marginals be made positive if all the constraints in which that variable appeared are eliminated. However, such an exercise can be difficult. Dillon's procedures may give guidance to those undertaking such an effort.

Table 11.1 Simple Model Basis Example

Panel A Model

```
    2  sets              var         /x1*x3/
    3                    constraint  /r1*r3/;
    5  variables         objfun;
    7  positive variables x(var);
    9  equations         objective
   10                    resource(constraint);
   12  parameter         objcoef(var) objective function coeffients
   13                    /x1 30, x2 20, x3 10 /;
   17  parameter         rhs(constraint)    constraint rhs's
   18                    /r1 10, r2  2, r3  8/;
   22  table   amatrix(constraint,var) aij matrix
   23                   x1      x2       x3
   24          r1        1       1        1
   25          r2        1      -1     -1.5
   26          r3                1        1;
   28  objective..
   29       objfun =e= sum(var, objcoef(var) * x(var));
   31  resource(constraint)..
   32         sum(var, amatrix(constraint,var)* x(var)) =l= rhs(constraint) ;
   34  model mymodel /all/
   38  *option lp=gamsbas;
   39  *$include "simple.bas";
   41  solve mymodel using lp maximizing objfun;
```

Panel B Exact Basis

```
 OBJECTIVE.M        =     1;
 RESOURCE.m ("R1")  =    25;
 RESOURCE.m ("R2")  =     5;
 RESOURCE.m ("R3")  =     0;
 OBJFUN.l           =   260;
 X.l ("X1")         =     6;
 X.l ("X2")         =     4;
 X.m ("X3")         =  -7.5;
```

Panel C Simplified Basis

```
 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 ;
```

Table 11.2 Simple Example Generating a Basis Using GAMSBAS

```
2  sets              var         /x1*x3/
3                    constraint  /r1*r3/;
5  variables         objfun;
7  positive variables x(var);
9  equations         objective
10                    resource(constraint);
12 parameter          objcoef(var) objective function coeffients
13                    /x1 30, x2 20, x3 10 /;
17 parameter          rhs(constraint)    constraint rhs's
18                    /r1 10, r2  2, r3  8/;
22 table   amatrix(constraint,var) aij matrix
23                  x1      x2       x3
24         r1        1       1        1
25         r2        1      -1     -1.5
26         r3                1        1;
28 objective..
29      objfun =e= sum(var, objcoef(var) * x(var));
31 resource(constraint)..
32        sum(var, amatrix(constraint,var)* x(var)) =l= rhs(constraint) ;
34 model mymodel /all/
38 option lp=gamsbas;
41 solve mymodel using lp maximizing objfun;
```

Table 11.3 GAMSBAS Basis File

```
 OBJECTIVE.m  =    1.00000000000    ;
 RESOURCE.m ("R1")  =    25.0000000000    ;
 RESOURCE.m ("R2")  =    5.00000000000    ;
 OBJFUN.l  =    260.000000000    ;
 X.l ("X1")  =    6.00000000000    ;
 X.l ("X2")  =    4.00000000000    ;
$offlisting
 X.m ("X3")  =   -7.50000000000    ;
```

Table 11.4 Files Including a Basis

```
 2  sets               var         /x1*x3/
 3                     constraint  /r1*r3/;
 5  variables          objfun;
 7  positive variables x(var);
 9  equations          objective
10                     resource(constraint);
12  parameter          objcoef(var) objective function coeffients
13                     /x1 30, x2 20, x3 10 /;
17  parameter          rhs(constraint)    constraint rhs's
18                     /r1 10, r2  2, r3  8/;
22  table   amatrix(constraint,var) aij matrix
23                    x1      x2       x3
24          r1         1       1        1
25          r2         1      -1     -1.5
26          r3                 1        1;
28  objective..
29       objfun =e= sum(var, objcoef(var) * x(var));
31  resource(constraint)..
32         sum(var, amatrix(constraint,var)* x(var)) =l= rhs(constraint) ;
34  model mymodel /all/
35  $include MAKEBAS.BAS
36  solve mymodel using LP maximizing objfun;
```

Table 11.5 Example with Saved and Reset Basis

```
 9  *     SECTION A       SET DEFINITON
11   SET  PRODUCT       PRODUCTS             /TABLES, CHAIRS/
12        TYPE          TYPES OF PRODUCT     /FUNCT ,FANCY/
13        RESOURCE      TYPES OF RESOURCES   /SMLLATHE,LRGLATHE,CARVER,
14                                                        LABOR,TOP/
15        METHOD        PRODUCTION METHODS   /NORMAL,MAXSML,MAXLRG/
16        PLANT         DIFFERENT PLANTS     /PLANT1, PLANT2/;
17     ALIAS(PLANT,PLANTS);
18  *     SECTION B       DATA DEFINITION
19   TABLE  PRODCOST(PRODUCT,METHOD,TYPE)  PRODUCTION COST
21                                    FUNCT         FANCY
22           CHAIRS.NORMAL             15             25
23           CHAIRS.MAXSML             16             26
24           CHAIRS.MAXLRG             17             27
25           TABLES.NORMAL             80            100;
27  TABLE RES(RESOURCE,PRODUCT,TYPE,METHOD) USE OF RESOURCES IN PRODUCTION
29             CHAIRS.FUNCT.NORMAL CHAIRS.FUNCT.MAXSML CHAIRS.FUNCT.MAXLRG
30  SMLLATHE            8                 13                    2
31  LRGLATHE            5                  2                   13
32  CARVER              4                  4                    4
33  LABOR              10                 11                   11
34   +          CHAIRS.FANCY.NORMAL CHAIRS.FANCY.MAXSML CHAIRS.FANCY.MAXLRG
35  SMLLATHE           12                 17                    5
36  LRGLATHE            7                  3                   15
37  CARVER             10                 10                   10
38  LABOR               8                  8                    8
39  +          TABLES.FUNCT.NORMAL TABLES.FANCY.NORMAL
40     LABOR            3                  5
41     TOP              1                  1   ;
43   TABLE TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS) TRANSPORT COST TO PLANTS
45                   PLANT1.PLANT2   PLANT2.PLANT1
46      CHAIRS.FUNCT       5               5
47      TABLES.FUNCT      14
48      CHAIRS.FANCY       5               5
49      TABLES.FANCY      18                       ;
52   TABLE PRICE(PLANT,TYPE) PRICE OF SETS
53                            FUNCT          FANCY
54          PLANT1             400            800
55          PLANT2             425            850
57   TABLE RESORAVAIL(RESOURCE,PLANT) RESOURCES AVAILABLE
58                   PLANT1      PLANT2
59       TOP           500
60       SMLLATHE     1100         1400
61       LRGLATHE      880          900
62       CARVER        500         1200
63       LABOR        1750         1250      ;
65   TABLE PRODPERSET(PRODUCT,TYPE)  PRODUCTS PER SET
67            FANCY    FUNCT
68    CHAIRS    6        4
69    TABLES    1        1
71   TABLE ACTIVITY(PLANT,PRODUCT,METHOD) TELLS IF A PLANT MAKES A PRODUCT
72             TABLES.NORMAL  CHAIRS.(NORMAL,MAXSML,MAXLRG)
73     PLANT1     1                    1
74     PLANT2                          1
76  *   SECTION     C      MODEL  DEFINITION
78   POSITIVE VARIABLES
79        MAKE(PLANT,PRODUCT,METHOD,TYPE)     NUMBER OF ITEMS MADE
80        TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS) NUMBER OF ITEMS TRANSPORTED
81        SELL(PLANT,TYPE)                    NUMBER OF SETS SOLD;
82           VARIABLES

83        NETINCOME                           PROFIT;
```

```
 84        EQUATIONS
 85        OBJT                                  OBJECTIVE FUNCTION ( PROFIT )
 86        RESOUREQ(PLANT,RESOURCE)              RESOURCES AVAILABLE
 87        PLANTPROD(PLANT,PRODUCT,TYPE)         PRODUCT BALANCE FOR A PLANT
 88        trans1(PRODUCT,TYPE,PLANTS)           transport configuration 1
 89        trans2(PRODUCT,TYPE,PLANTS)           transport configuration 2;
 91        OBJT..      NETINCOME =E=
 92        SUM((PLANT,TYPE),
 93           PRICE(PLANT,TYPE) * SELL(PLANT,TYPE))
 94   -SUM((PLANT,PRODUCT,METHOD,TYPE)$ACTIVITY(PLANT,PRODUCT,METHOD),
 95          MAKE(PLANT,PRODUCT,METHOD,TYPE)*PRODCOST(PRODUCT,METHOD,TYPE))
 96   -SUM((PRODUCT,TYPE,PLANT,PLANTS)$TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS),
 97          TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
 98         *TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS));
100    RESOUREQ(PLANT,RESOURCE)..
101           SUM((PRODUCT,TYPE,METHOD)$ACTIVITY(PLANT,PRODUCT,METHOD),
102                RES(RESOURCE,PRODUCT,TYPE,METHOD)
103               *MAKE(PLANT,PRODUCT,METHOD,TYPE) )
104               =l= RESORAVAIL(RESOURCE,PLANT) ;
106    PLANTPROD(PLANT,PRODUCT,TYPE)..
107               SUM(PLANTS$TRANSCOST(PRODUCT,TYPE,PLANT,PLANTS)
108                          ,TRNSPORT(PRODUCT,TYPE,PLANT,PLANTS))
109             - SUM(PLANTS$TRANSCOST(PRODUCT,TYPE,PLANTS,PLANT)
110                  ,TRNSPORT(PRODUCT,TYPE,PLANTS,PLANT))
111             + SELL(PLANT,TYPE)*PRODPERSET(PRODUCT,TYPE)
112             =L= SUM(METHOD$ACTIVITY(PLANT,PRODUCT,METHOD),
113                          MAKE(PLANT,PRODUCT,METHOD,TYPE));
115  trans1(PRODUCT,TYPE,PLANTS)..
116      TRNSPORT(PRODUCT,TYPE,"PLANT1",PLANTS) =g= 2;
117  trans2(PRODUCT,TYPE,PLANTS)..
118      TRNSPORT(PRODUCT,TYPE,"PLANT2",PLANTS) =g= 2;
119   MODEL FIRMb    Base Model           /OBJT,RESOUREQ,PLANTPROD/
120   MODEL FIRM1    First Configuration  /OBJT,RESOUREQ,PLANTPROD,trans1/
121   MODEL FIRM2    Second Configuration /OBJT,RESOUREQ,PLANTPROD,trans2/
125  * SECTION D    SOLVE THE PROBLEM
126   SOLVE FIRMb USING LP MAXIMIZING NETINCOME;
127
128  *Save the base model Basis
129  set solinfo /level,marginal/
130  parameter
131  savMAKE(PLANT,PRODUCT,METHOD,TYPE,solinfo)    basis info on make var
132  savTRNSPO(PRODUCT,TYPE,PLANT,PLANTS,solinfo)  basis info on trnsport vr
133  savSELL(PLANT,TYPE,solinfo)                   basis info on sell var
134  savnetinc(solinfo)                            basis info on netincome
135  savOBJT                                       basis marg for objt eq
136  savRES(PLANT,RESOURCE)                        basis marg resoureq eq
137  savPLANTPR(PLANT,PRODUCT,TYPE)                basis marg plantprod eq;
138
139  *save variable levels
140   savMAKE(PLANT,PRODUCT,METHOD,TYPE,"level")
141                =MAKE.L(PLANT,PRODUCT,METHOD,TYPE);
142   savTRNSPO(PRODUCT,TYPE,PLANT,PLANTS,"level") =
143                                TRNSPORT.L(PRODUCT,TYPE,PLANT,PLANTS);
144   savSELL(PLANT,TYPE,"level")= SELL.L(PLANT,TYPE);
145   savnetinc("level")=NETINCOME.L;
146  *save variable marginals
147   savMAKE(PLANT,PRODUCT,METHOD,TYPE,"marginal")
148        =MAKE.M(PLANT,PRODUCT,METHOD,TYPE);
149   savTRNSPO(PRODUCT,TYPE,PLANT,PLANTS,"marginal") =
150                              TRNSPORT.M(PRODUCT,TYPE,PLANT,PLANTS);
151   savSELL(PLANT,TYPE,"marginal")= SELL.M(PLANT,TYPE);
152   savnetinc("marginal")=NETINCOME.M;
153
154   savOBJT=OBJT.M;
```

```
155   savRES(PLANT,RESOURCE)=RESOUREQ.M(PLANT,RESOURCE);
156   savPLANTPR(PLANT,PRODUCT,TYPE)=PLANTPROD.M(PLANT,PRODUCT,TYPE);
157
158   SOLVE FIRM1 USING LP MAXIMIZING NETINCOME;
159
160  *reestablish basis to that stored in base model
161
162  *reestablish variable levels
163   MAKE.L(PLANT,PRODUCT,METHOD,TYPE)
164             =savMAKE(PLANT,PRODUCT,METHOD,TYPE,"level");
165   TRNSPORT.L(PRODUCT,TYPE,PLANT,PLANTS)=
166                      savTRNSPO(PRODUCT,TYPE,PLANT,PLANTS,"level");
166   SELL.L(PLANT,TYPE)= savSELL(PLANT,TYPE,"level");
167   NETINCOME.L= savnetinc("level");
168
169  *reestablish variable marginals
170   MAKE.M(PLANT,PRODUCT,METHOD,TYPE)
171              =savMAKE(PLANT,PRODUCT,METHOD,TYPE,"marginal");
172   TRNSPORT.M(PRODUCT,TYPE,PLANT,PLANTS)=
173                       savTRNSPO(PRODUCT,TYPE,PLANT,PLANTS,"marginal");
174   savSELL(PLANT,TYPE,"marginal")= SELL.M(PLANT,TYPE);
175   NETINCOME.M= savnetinc("marginal");
176
177  *reestablish equation marginals
178   OBJT.M=savOBJT;
179   RESOUREQ.M(PLANT,RESOURCE)=savRES(PLANT,RESOURCE);
180   PLANTPROD.M(PLANT,PRODUCT,TYPE)=savPLANTPR(PLANT,PRODUCT,TYPE);
181
182   SOLVE FIRM2 USING LP MAXIMIZING NETINCOME;
```

Chapter 12 Increasing GAMS Program Execution Efficiency

The execution time and/or memory requirements for a GAMS program is a function of the GAMS implementation and the users skill in exploiting the ways that GAMS executes code. Huge efficiency gains can be achieved. In one case we were able to reduce the execution time of a segment of GAMS code from 30 minutes to 15 seconds simply by rewriting a small amount of GAMS code without changing the types of results being generated by the procedure at all.

In this chapter we cover a number of aspects of GAMS usage that can be used to improve efficiency. The coverage involves both sides of the issue: diagnosis of whether and where there is a problem and manipulation of the GAMS code to repair the problem. This chapter will most directly address the execution time issue. Section 10.2 discusses memory issues in a more detailed and also complementary fashion.

There is one important limitation to this chapter that the reviewer should note. The coverage is aimed toward the reduction of the time to execute a problem within GAMS not within the GAMS solvers i.e. the execution time within CPLEX or ZOOM or OSL or some other code. Only brief mention of this latter issue will be made in section 12.4.

Readers should also note that the material in this chapter is not only useful for the repair of specific models. Rather by studying and employing the efficiency enhancing techniques we display herein, one can improve the effectiveness of future GAMS programming. For example, the section on set addressing will reveal conventions that should be used in all GAMS programming yielding efficiency gains without the need for model efficiency investigations

12.1 Is Efficiency a Concern

GAMS can take a lot of time or use a lot of space in computations and model setup.

Ultimately the judge of whether the time use is excessive is the user. Questions such as

Does the program take more time than you feel it should?

During execution does the screen show execution of one line number for a long time?

Is the procedure used often enough that efficiency is a concern?

need to be addressed. If the answer to any of these questions is a yes then further investigation is in order to see whether there are poorly executing portions of the program.

12.1.1 Watching the Screen to Find Speed Problems

One way a modeler can get an indication of where timing problems might lie is to watch the screen during execution. GAMS reports the line numbers which it is executing to the screen. Thus if the program pauses on a line number for a moderately long time, then one would look at that line as a cause of slow execution. For example in the model in Table 12.1 panel a during execution the procedure pauses noticeably on the execution of lines 14,16,23,24,25 and 31, which as we show in the next section are the longer executing statements in the program.

The approach of watching the screen is not always a very good way to proceed for two reasons. First, staring at the screen for long time periods may not be effective and one may miss certain statements, may identify statements improperly or get distracted and have to redo the approach repeatedly. Second, GAMS line reporting is misleading when loops and if statements are being executed. For example in the model in Table 12.2 GAMS pauses longest on statements 28,20,21,22. Line 28 is the loop statement itself while 20-22 are the LP model equations. The individual calculations are not reported to the screen i.e. those on lines 29-31. Thus one would not get any indication other that the loop is taking a lot of time.

12.1.2 PROFILE Use to Find Speed and Memory Problems

GAMS provides an alternative to attentive watching of the screen during execution. Namely, one can obtain information about the execution time characteristics of a program by using the PROFILE option. When a program is run with the PROFILE option active then GAMS generates output in the LST file on the time it takes to execute each statement. PROFILE related portions of the LST file associated with the model in Table 12.1 appear in Table 12.3.

The profile command causes GAMS to give information on: a) the GAMS statement number of the instruction being profiled; b) the symbol name being executed or being worked on; c) the execution time of each statement; d) cumulative program execution time; e) the number of cases for which the statement is executed (if the cases exceed one); and f) cumulative memory use. In the example, the entry labeled starting with a 14 reports that executing the 14th line of the program (Table 12.1) which is an assignment of values into Z takes 4.07 seconds of execution time, contributing to 5.66 seconds of cumulative execution time while executing the statement for 100,000 cases. Through this information, GAMS indicates where large numbers of cases and/or large execution times are encountered (i.e., statements 12,14,16, 23, 24,25,29 and 31). In turn, one can examine those statements to see if they can be reworked for faster execution.

The profile option is invoked by one of two means. One can alter the GAMS call to include commands like the following:

```
GAMS MYMODEL   PROFILE = 1        on DOS machines or
GAMS MYMODEL   -PROFILE   2       on UNIX machines.
```

or one can insert an option statement into the program as follows:

```
OPTION PROFILE=3;
```

The number after the profile request tells GAMS how deeply within LOOPS and IF

statements to report execution characteristics. When PROFILE = 1 is used, GAMS does not give information within any IF statements or LOOPS, just information on the overall IF or LOOP. When PROFILE = 2 is used, execution characteristics are given on statements within the first level of LOOP and IF statements. PROFILE = 3 will go within the second level of IF and LOOP statements, etc. Large profile values are needed to investigate execution characteristics within deeply nested LOOP and IF statements but they can generate a lot of output if the loops are executed repeatedly.

The profile option can generate a tremendous amount of output much of which is not informative. Our example shows several statements are reported for which there is not meaningful execution time. One can suppress this information by using a tolerance on the minimum amount of execution time that a statement must use to be reported. In the case of the example this is a very small amount of time and we could use

OPTION PROFILETOL = 1;

In bigger models one could for example, allow reporting of statements that took 2, 10 or more seconds of execution time. Notice in the example model the statements with the largest execution times are the ones that go through thousands of cases or have terms summing over thousands of set index possibilities.

Perhaps it is worthwhile to illustrate the effect of using a higher number PROFILE option. The example in Table 12.2 is a looping version of the faster executing version of the example in Table 12.1. The profile related output under the setting of PROFILE = 1 executions is given in Table 12.4. (Note we have added the lines on the loop number and solve report to show program progression). This output shows two notable things. First, all of the executing statements within

the loop are not profiled. In particular, only a composite time for the non model generation statements is reported along with the generation time for each model statement. Second, the line number chosen for reporting of the information is in this case the solve statement (line 32). However in a loop without a solve then all execution time is reported solely for the LOOP statement. For example running the model in Table 12,4 with lines 32 and 33 deactivated results in a reporting only for the loop statement (line 28) without any reporting for the lines within the loop.

Now suppose we illustrate the consequences of setting the profile number to a higher level. In particular suppose we run the model in Table 12,2 with PROFILE=2. Simultaneously we will also set PROFILETOL=1 suppressing output for statements executing in less than a second. The resultant output is in Table 12.5. There note that the output for the statements before the loop is suppressed because they are fast. We then get details on how long each of the statements within the loop take.

12.1.3 Looking Deeper into Complex Statements

Sometimes one runs profile and finds that the time problem is inside a tremendously long statement. For example, we regularly run models where the objective function and some report calculation lines are well over 100 lines long in the LST file containing 40 or so added terms. The identification of a time problem in such a long statement still leaves one with the question of where is the problem. In such cases the approach we take involves the use of on/off text and *'s to deactivate parts of the code. Namely, if a multi-term piece of code spanning from listing lines 1182 to 1294 is identified as slow we would re-execute the code but split the term so say lines

1240 to 1294 are deactivated (surrounding them with an ontext - offtext sequence. Then if the remaining lines still used the bulk of the time we would use the text features to deactivate further code until the text addition greatly reduced the time use. We would then know that the last section put into the text status contained the slow executing code. We would then proceed to search therein until problematic portion of the code is identified.

12.1.4 A time related search strategy

Sometimes one cannot let the whole program run in order to find the memory problems. In particular suppose that the program starts up and begins executing some statements in an IF or a LOOP and 30 minutes later is at the same point. In such a case a number of strategies are relevant to isolate where the problem occurs. These parallel the cases treated in section 10.2.3 on finding memory use problems where they are more extensively discussed. The strategies are listed below.

12.1.4.1 Small to large

Use the small to large strategy of Chapter 5**** and look for timing problems with PROFILE with smaller, representative data sets, and if possible fix the problem, then come back to the big model and see if that has corrected the problem. On the other hand, if the problem only exists in the big model then one should use the approaches below.

12.1.4.1 Search calculations for Time Hogs

When GAMS is doing pre or post solution calculations, it can use excessive time. Commonly this is caused by a statement that treats a runaway number of cases where excessive elements are referenced. Statements involving assignments of parameters, variable bounds and scaling factors are usually the cause. Unfortunately, when time use is really excessive one does not

always wait for the PROFILE information but rather terminates the program. Also the line number from screen watching may reference a loop or an if statement which encompasses multiple statements. Thus the particular poorly executing statement may need to be discovered through a search. Such a search involves first finding the well executing portion of the code just above the problematic area then narrowing in to find the problem by finding the first poorly executing, time hog statement. To identify a guess at the last good statement one can examine the end of the LST file, the LOG file or watch the screen listing. Then one begins a search for the subsequent slow statement. Users can employ the GAMS $ON/OFFTEXT syntax to search out the offending statement basically by removing the statements from the last good one to the end of the program from execution then advancing the ontext position until the problematic code is found(this is more extensively discussed in 10.2.3.2). This can be a time consuming procedure particularly in big models and efficiency can be increased by using GAMS save and restart files to save the part of the program that performs satisfactorily and then only execute the small part of the program in which the problem is occurring (see the chapter on comparative model runs for an example of saving and restarting).

12.1.4.2 Finding Time Hogs in Model Generation

Another place where GAMS may exhibit excessive time use is during model generation. This may be a difficult stage at which to discover exactly where the problem is located, largely because of the print buffer problem as discussed in 10.2.3.2. One should, if possible, watch the screen and take note of the last statement number which is executed i.e., if it says executing line 26 one would know the problem is in line 26 or later. If this is not possible one should save the log file (using the LO=2 option on the GAMS call) to find the hypothesized last “good” statement.

The search for the bad statement is then on. This search is conducted by forming a restricted model. The restricted model may be formed by

1) commenting out (using asterisks or $ON/OFFTEXT) certain equations from the equation listing and the equation definitions (.. Entries)
2) creating a model statement which omits a number of the equations, or by
3) following a small to large approach by restricting the sets over which equations are generated and terms within the equations are summed. For example, if one had a model in which there were constraints generated for every month, for a number of years involving sums over every crop, the constraint and terms could be redefined over subsets containing only January and February, the first and second years and two selected crops.

In turn, the restricted model can be run to see if it works and to examine time use.

The basic methodology for discovering the location and cause of excessive time use is:

1) identify the last equation successfully executed;
2) identify the equations that have not been generated as of the last known point of successful execution. Note, GAMS equations are generated in the order in which the names appear in the EQUATION declaration section not in the order in which the equation definitions appear (the ..syntax). Thus, if equation B is declared right before A and equation B is the last successfully generated, then one can conclude that equation B along with all equations above it are satisfactory. So then one would investigate the subsequent equations from equation A on down.
3) eliminate or simplify all subsequent equations using one of the procedures discussed

above.

4) resubmit the GAMS job.

5) if the model runs and more than one equation was eliminated or simplified restore some of the earlier of these to full executing status. Thus re-include at least the first eliminated equation and if say four equations were eliminated include the first two in some form of a binary search. Go to Step 4 if eliminating more equations, if not go to Step 6.

6) now an equation causing the problem has been identified.

12.1.4.3 Solution Time Hogs

Another place a lot of computer time can be used is in the solution process. Approaches for resolution of that problem is discussed in chapter 10.

12.2 Improving Efficiency

Once one has found the statement(s) that are using the bulk of the time in program execution, the question then becomes how can I speed up execution. Often a lot of the time use is unnecessary. Often one can reduce execution time by better set addressing and referencing, avoiding unnecessary cases, avoiding repeat calculations and or trading memory use for speed.

12.2.1 Set Addressing and References

One big time consideration in GAMS usage involves the way set indices are referenced with respect to a data parameter. GAMS employs a sparse matrix data storage scheme. In particular, the elements of an array X(A,B,C) are stored in an order such that say for the first A element and the first B element all data associated with each C element appears. This is than followed by all the C data for the 2nd B element, then all for the third etc. After that we then have

data in a similar fashion for the second A element then the third on through the end. GAMS retrieves these data in the fastest manned when they are referenced in the same order as the parameter is defined. Thus, for example, the statement Y(a,b,c)=X(a,b,c); executes faster than the statement Y(a,b,c)=X(b,c,a). This occurs because in the first case the referencing of Xx is always in its definition order whereas in the second case the referencing will proceed in order for Y but will be out of order for X. In general a number of rules should be followed which will increase execution speed.

1) Always try to arrange calculations, sums, equation references etc so to the extent possible the referencing of large data arrays are in an order consistent with the order of the sets in their definition.

2) Try for consistency to always define set references in the same order. For example if one has the sets State, County, Industry and Process to the extent possible when an item in the parameter, equation, variable, or any other item in the program employs more than one of these sets arrange the reference so State is always the first one mentioned, then county then Industry then Process. If one has items which do not employ all the set names still follow their relative ranking.

3) When setting up sums or products, then put the set references in the same order as defined for the data items.

An example which shows the time consequences of these rules is given in Table 12.1. There note that in Panel A we have a model which does not obey the conventions while in panel B we have a model which does. The relative execution times for a model on the same Pentium 120 laptop are as follows.

Line number	Profile time Panel A Slow Version	Profile time Panel B Faster version
2	0.00	0.00
12	1.54	1.81
14	4.06	3.24
16	3.52	1.76
27	0.00	0.00
28	0.00	0.00
29	0.00	0.00
23	8.46	5.82
24	10.33	9.01
25	9.22	6.43
29	1.10	0.82
29	1.98	2.09
29	0.32	0.27
31	4.18	2.58
Presolve Total	38.34	31.14
Solve Time	38.00	38.00
Post Solve Total	4.50	2.85
Grand Total	80.84	71.99

Note the overall time savings is about 10% of execution time. This is actually a smaller savings than we expected. Sometimes we have seen performance differences in excess of 50%. Newer versions of GAMS are improving data access time, however following the above rules in GAMS programming will improve efficiency.

12.2.2 Avoiding Unnecessary Cases by Adding Conditions

The biggest time and memory hog in most GAMS programs involves programs which do not adequately avoid treating unnecessary cases. For example one may define an array with set indices for State and County then in sums have the program when dealing with a state index counties that are in another state (i.e. Texas has 256 counties and the US over 3000 when summing for Texas one should avoid treating all the counties in the other states. In such a setting, it is desirable to define $ conditions to avoid consideration of unneeded or irrelevant terms. The general approach is to reason out a condition that specifies when an equation, variable, computed quantity or expression term is desired, then restrict the cases considered by the relevant expression(s) using that condition. Such a strategy can be used to in the specification of equations, calculations, and report writing statements. It is also desirable to introduce the most limiting conditions as early as possible in the expression so as little calculation and data lookup is done as possible before the condition is checked. Consider our state, county example. Suppose we had land area by county and land type [landarea(county, landtype)], and wished to add land area by state. Suppose we also had a two dimensional set telling which states were composed of which counties [matchup(state, county)]. Now consider 3 pieces of code:

```
totalland(state)=sum((county,landtype),landarea(county,landtype)$matchup(state,county));
totalland(state)=sum((county,landtype)$matchup(state,county),landarea(county,landtype));
totalland(state)=sum(county$matchup(state,county),Sum(landtype,landarea(county,landtype)));
totalland(state)=sum(matchup(state,county),Sum(landtype,landarea(county,landtype)));
```

The last two would be the fastest. The first would go through all cases, the second

12.2.2.1 Calculation Statement Specifications

Data defined over several sets can cause execution to be quite slow in performance. The use of $ conditions in dealing with such data can yield dramatic time and space reductions. For example, GAMS considers all cases of all subscripts when a parameter is defined. Thus the statement

X(A, B, C, D, E) = 5

will execute for all possible members of A, B, C, D, and E. If for example, each of these sets had 20 members then 3.2 million values would be assigned taking considerable time and memory. If on the other hand if only several hundred active cases were defined, then using the command

X(A, B, C, D, E) $ GOODCASE (A, B, C, D, E) = 5

where the variable GOODCASE equals one for the active cases and zero otherwise will greatly cut down on computation time. The general approach is to reason out a condition that specifies when an equation, variable, computed quantity or expression term is desired, then restrict the cases considered by the relevant expression(s) using that condition. It is desirable to introduce the most limiting conditions as early as possible in the expression so as little calculation and data lookup is done as possible before the condition is checked.

Consider a state, county example. Suppose we had land area by county and land type [landarea(county, landtype)], and wished to add land area by state. Suppose we also had a two dimensional set telling which states were composed of which counties [MATCHUP(state, county)]. Now consider 2 pieces of code:

```
totalland(state)=sum((county,landtype), landarea(county,landtype));
totalland(state)=sum((state,county,landtype)$MATCHUP(state,county),landarea(county,landtype)
);
```

The last would be the fastest. The first would go through all cases including counties which are not in a state, the second would be much faster. A small example (landexam.gms) when executed takes 3.03 seconds for the first case and 0.10 second for the latter (less than 1/30 th the time).

12.2.2.1.1 An Aside -- Placement of Conditions

One important element in the use of conditions involves their placement in the string of GAMS code. Consider two cases

```
x(i,j,k)$data(i,j,k)=1+y(i,j,k);
x(i,j,k)=1+y(i,j,k)$data(i,j,k);
```

These two expressions will have very different timing and results implications. The first will only go through cases where the data array contains nonzero entries. The second will cover all cases only adding in y when the data array have nonzero cases. In addition if the x array was previously defined then the first expression will only redefine the cases where data are nonzero leaving the other cases at their original values. The second will redefine the entire array.

This implies that when using GAMS maximum speed will be attained by placing conditions so that they have maximum effect on limiting the cases considered. Thus,

1) the conditions should be placed as close to the beginning of the statement as possible (for example on the left hand side of a calculation equation, before the .. part of the model equations or in the earliest sums in calculations).
2) the most powerful condition eliminating the most cases with the least data look up should be placed first; and
3) the modeler should pay attention to defining tight conditions when dealing with

time intensive parts of the GAMS code.

12.2.2.2 Equation Statement Specifications

When defining equations using the .. notation one can use $ conditions to condition the existence of whole equations or of terms within equations. Conditions on the existence of an equation are entered by specifying a $ in association with the .. part of the specification. This causes equations to be skipped if the condition is not true. An example in the context of the transportation problem appears in Table 12.6. Here a transportation problem with data for four destinations and three sources is specified as if it had 300 destinations and 150 sources. The equation suppression example involves a comparison of lines 36-37 and 39-40 which are reproduced here.

```
34   slDEMANDEQ(MARKET)..
35       SUM(PLANT, SHIPMENTS(PLANT, MARKET))=G=DEMAND(MARKET);

39   inDEMANDEQ(MARKET)$demand(market)..
40     SUM(PLANT,  SHIPMENTS(PLANT, MARKET)) =G= DEMAND(MARKET);
```

In line 34-35 the slDEMANDQ equation is always generated so the model will have 300 of those equations. However, in an alternative setup (inDEMANDQ line 39-40) the demand equation is only generated when the quantity demanded (DEMAND(market)) is nonzero, thus only four constraints will be present. These modifications cause the PROFILE timing report to indicate the equation took 0.27 seconds to generate as opposed to 2.31 seconds without the conditional. Note other speedups are possible in terms of the supply equation however to do it right these must be incorporated with terms that suppress equation terms for places without shipment constraints as discussed in the next section.

Conditions on the existence of terms in an equation are entered by specifying a $ in association with the algebraic part of the equation specification. This causes terms to be skipped if

the condition is not true. An example of this also appears in Table 12.6 (trnspsu.gms) . A term suppression example involves a comparison of lines 34-35 and 49-50.

```
34   slDEMANDEQ(MARKET)..
35       SUM(PLANT, SHIPMENTS(PLANT, MARKET))=G=DEMAND(MARKET);

49   fsDEMANDEQ(MARKET)$demand(market)..
50    SUM(PLANT$cost(plant,market),  SHIPMENTS(PLANT, MARKET)) =G= DEMAND(MARKET);
```

Note in line 35 all the set elements are considered in the sum involving the SHIPMENTS variable whereas in line 50 only those variables with a nonzero transport cost are considered. Thus, in generating the equations rather than defining terms for all 45000 possible SHIPMENTS variables only 12 will pass the test. Similar terms are suppressed in the supply and objective function equations.

The net effect of activating the speedup conditionals in the fast as opposed to the slow model is that PROFILE reports a total model generation time across the three equations of 0.11 as opposed to 3.19 seconds. Also the model size is reduced from 45,001variables and 451 equations to 13 variables and 8 equations. This also saves memory reducing use from 5.2 to 3.4 megabytes.

12.2.2.3 Specification of Variables in Models

The speedups in the last section involving the use of conditionals within equation terms also suppressed variables. For example the lines

```
45    TCOST =E= SUM((PLANT,MARKET)$cost(plant,market)
46                          , SHIPMENTS(PLANT,MARKET)*COST(PLANT,MARKET));
47   fsSUPPLYEQ(PLANT)$supply(plant)..
48    SUM(MARKET$cost(plant,market), SHIPMENTS(PLANT, MARKET))=L=  SUPPLY(PLANT);
49   fsDEMANDEQ(MARKET)$demand(market)..
50    SUM(PLANT$cost(plant,market),  SHIPMENTS(PLANT, MARKET)) =G= DEMAND(MARKET);
```

all suppressed the SHIPMENTS variables for which there were no transportation costs.

The treatment of variable in a model where equations and terms are being suppressed can be a tricky enterprise. For example modification of lines 48 and 50 to drop the condition on shipment cost

```
45    TCOST =E= SUM((PLANT,MARKET)$cost(plant,market)
46                        , SHIPMENTS(PLANT,MARKET)*COST(PLANT,MARKET));
47   fsSUPPLYEQ(PLANT)$supply(plant)..
48    SUM(MARKET, SHIPMENTS(PLANT, MARKET))=L=  SUPPLY(PLANT);
49   fsDEMANDEQ(MARKET)$demand(market)..
50    SUM(PLANT,  SHIPMENTS(PLANT, MARKET)) =G= DEMAND(MARKET);
```

results in a model (transbad.gms) with 8 equations and1500 variables and an objective function value of zero. This occurs because the model generates

a) all the outgoing shipments in line 48 to both real and zero cost demand points (an extra 296*4 shipments); and

b) all the incoming shipments in line 50 from the zero cost supply points (an extra 147 *3 shipments)

The latter shipments cause the zero objective function as the model contains shipment possibilities for plants 1-147 which go to each demand constraint which meet the demand (equation 50) have a zero cost in the objective function and are not limited by a supply equation (since equation 48 is suppressed when supply equals zero).

This introduces another of the Golden Rules of GAMS use. When suppressing key equations and or variables be sure to suppress the variables in a consistent fashion across all equations. Thus in our example everywhere the variable SHIPMENT occurs we condition on the cost variable. In a more extensive model we might also have to condition shipments on having nonzero supply at point of origin and nonzero demand at the destination. We would also have to be careful to have the same condition at all points where the SHIPMENT variable appears.

Another possibility is available for variable suppression. One can set irrelevant variables to zero by fixing their values using a statement like

```
SHIPMENTS.FX(PLANT,MARKET)$(COST(PLANT,MARKET) eq 0)=0;
```

However this will causing the bad solution immediately above does not speed up anything causing

all the GAMS operations to still be needed (see the model transfix.gms). One can go even further and instruct GAMS to suppress generation of any variable held fixed by .FX commands using the instruction

```
Modelname.holdfixed=1;
```

However this does not speed up the GAMS computations up through model generation although it does reduce the model passed to the solver down to the 13 variable, 8 equation version. Thus for speedups regarding variables we recommend the use of the conditions in the equations.

12.2.2.4 Post Solution Report Writing Calculations

One may also use $ controls to suppress report writing calculations. For example the equation:

Y=SUM((A,B,C,D,E,F,G), (DAT(A)+IT(B,C)+Y(D,E)+W(F,G))*X.L(A,B,C,D,E,F,G))

will perform much faster with the addition of a $ condition on the sum as follows

Y=SUM((A,B,C,D,E,F,G)$X.L(A,B,C,D,E,F,G),

(DAT(A)+IT(B,C)+Y(D,E)+W(F,G))*X.L(A,B,C,D,E,F,G));.

This will result in the calculation only being done when nonzero solution values (which are commonly sparse) are involved and will avoid excess work.[b]

12.2.3 Better Using Sets

When one is trying to gain efficiency in model calculations there are several ways that set referencing can be employed.

a) One should make sure that the ordering of sets in sums and calculations is as

[b]Such a modification reduced run time for a report writer from 2 hours to less than 10 minutes.

consistent with the ordering of the data items as possible. (Note GAMS has recently introduced internal optimization procedures which try to automatically reorder set addresses to speed up things but this is still good modeling practice.)

b) In cases where relatively complex and time consuming calculations are needed to see if an item should be calculated one can introduce a set that expresses this relationship. Consider the state county land example again. In that example(landexam.gms) we computed the set MATCHUP in lines

```
12      MATCHUP(state,county)$((ord(county) ge (ord(state)-1)*60+1)
13                         And (ord(county) le ord(state)*60+1))=yes;
```

then we used that set in our calculation

```
17  totalland(state)=sum((county,landtype)$MATCHUP(state,county),
18                         landarea(county,landtype));
```

However we could have used the MATCHUP definition in the calculation

```
17  totalland(state)=sum((county,landtype)
18                     $((ord(county) ge (ord(state)-1)*60+1)
19                            And (ord(county) le ord(state))),
20                          landarea(county,landtype));
```

Without any reference to the MATCHUP set. This latter revision takes 3.90 seconds to execute as opposed to the original 0.27. Thus if the need to match the states and counties was prevalent in the code, it would be much faster to define the set once and then repeatedly use it.

c) The introduction of sets which define relevant cases does not necessarily have to involve conditionals. We could have the same effect as the MATCHUP conditional in lines 17-18 above if we used the code

```
17  totalland(state)=sum((MATCHUP(state,county),landtype),
18                         landarea(county,landtype));
```

In such a case we are summing over all cases of MATCHUP for a particular state choice. This code will not be any faster than the proper use of conditionals but is

more compact.

d) One can place conditions on the sets in sums and should do it to eliminate as many cases as possible as early as possible as discussed in Section 12.2.2.1.

12.2.4 Trading Memory for Time

The section above on better using sets introduces a more general topic. Namely, there are a number of cases during GAMS use where one can in effect trade memory for time. In particular, one can define parameters or depicting calculation intensive items then reuse those items in the code. The cases of this worth mention involve repeat calculations and report writing ordering as well as the set example of the last section.

12.2.4.1 Avoiding Repeat Calculations

One may store computationally expensive items which are needed in several places in the execution of a GAMS code. For example in the AMS model (McCarl et al) the objective function and a number of report writing calculations include the computation of variable cost summed across about 100 inputs. Further for versions of the model the sum is the same across a tillage systems, land treatments and rotations. The basic code looks something like

```
Z=SUM(( CROP, TILLAGE, LANDTREAT, ROTATION ),
        ACREPLANT( CROP, TILLAGE, LANDTREAT, ROTATION)*
        SUM(INPUT, USAGE(INPUT,  CROP)));
```

This code was revised by defining a parameter for the input usage sum and substituting i.e.;

```
INPUTUSE( CROP) =
         SUM(INPUT, USAGE (INPUT, CROP));

Z=SUM(( CROP, TILLAGE, LANDTREAT, ROTATION ),
        ACREPLANT( CROP, TILLAGE, LANDTREAT, ROTATION)*
        INPUTUSE (CROP));
```

This avoided recalculating the sum for every TILLAGE, LANDTREAT, AND ROTATION case

in the many places the computation occurred. It saved more than 20 minutes in execution time for each model run of which we do more than a thousand a year. However on cautionary note is in order. Note that when one modifies the USAGE array during a study that when using this procedure one will only have and effect on the model arrays like INPUTUSE are recalculated.

12.2.4.2 Attaining Natural Ordering for Displays

Another case of memory substitution involves the interplay of the order of items for output and subscript ordering. Suppose for convenience one wishes a to have the array xsum(j,i) computed as

```
xsum(j,i)=sum((k,l,m,n),x.l(i,k,l,m,n,j));
```

where the j,i order for xsum is chosen to best work with the GAMS DISPLAY statement output formatting. In this case one may find it faster to compute a temporary sum

```
tempsum(i,j)=sum((k,l,m,n),x.l(i,k,l,m,n,j));
xsum(j,i)=tempsum(i,j));
```

turning the data around into the desired order after the more involved calculation rather than treating the data out of order during that calculation.

12.4 Solver Efficiency Modifications

One place a lot of time can be used in a GAMS exercise is in the solver. As said above this book does not deal extensively with solver and problem formulations issues. Nevertheless a few comments are in order on problem numeric properties, advanced basis and starting point usage, problem reformulation, solver control and solver choice.

12.4.1 Problem reformulation

When dealing with large problems solver efficiency is almost always increased by attention to improving the numeric properties of the problem. Such topics are treated in chapter 10 under the sections on scaling (10.3) and avoiding degenerate cycling (10.4)

12.4.2 Advanced Basis Usage and Starting Points

Solver efficiency is almost always enhanced by the provision of an improved starting point. This point is important in LP and sparse NLP problems for the formation of an initial basis as discussed in Chapter 11. It is also important in NLPs as the initial Jacobian and gradients are formed using the starting point. The best possible starting point should be used as close as possible to the optimal solution, providing as full a basis as possible.

12.4.3 Problem reformulation

One can also change the optimization model formulation by deleting constraints, converting constraints to bounds, linearizing difficult nonlinear terms, adding constraints to eliminate integer solutions, and fixing the values of integer variables, among many other things. Details on such structural modifications are beyond the scope of these notes. Suffice it to say Luenberger; Williams; McCarl and Spreen; and many others address problem reformulation aspects of this topic while the presolves in OSL and CPLEX automatically do some of these functions. Also the reader should note that

a) Linearization of difficult nonlinear terms can be a real boon to efficiency allowing these authors to alter the time for completion of a solve sequence from 6 days to 6 hours in one case.

b) Tightening of constraints in MIP is a very important endeavor, The addition of constraints better reflecting interlinkages both between feasible values of alternative integer variables and between integer and continuous variables has in more than one instance cut solution time for us by more than a factor of one hundred.

12.4.4 Solver Choice

One important but difficult to address topic involves solver choice. GAMS provides access to at least a dozen solver with at least half a dozen of them applicable to for example linear programs. Solver choice can greatly influence solution times. An experiment 2 years ago with solution of a sequence of large LPs took 6 days on MINOS5, 18 hours on OSL and 6 hours on CPLEX. A similar experiment on a MIP showed OSL was the best performer but other experiments have favored CPLEX. Generally all that can be said is that the different solvers do perform differently and by choice of solver, along with solver options, one can substantially alter solution time. Several factors should be taken into account when choosing a solver in situations where execution time is an important consideration:

a) Get a solver fundamentally designed to solve the type of problem you are addressing. Solvers like MINOS5 and CONOPT while they can solve LPs are designed for NLPs. Find this out by consulting the solver manuals distributed with the GAMS system.

b) Investigate how current the solver is? Some of the GAMS solvers are the subject of continuing research and improvement. Others have remained essentially static for as long as a decade.

c) Find out whether the solver is a commercial product that is maintained by a group

and sold industrially or whether it is largely academic, this may have implications for its continued success and for bug fixing.

d) Start simple first. You maybe happy with the default solver which comes with GAMS (i.e. BDMLP for LPs). However you may find a need for better performance.

e) Ask experienced users about which solver you should buy for your problem type. Try the GAMS email list for information (GAMS - L@vm.gmd.dc)

f) If still in doubt try to arrange a demonstration licence through GAMS (www.gams.com) and test alternatives for problem versions.

12.5 GAMS and Solver Options

When using the GAMS based solvers one can alter performance by manipulating solver parameters using option statements within GAMS or parameters within the solver option files. The performance alteration largely falls into general options or solver specific options. In terms of general options, one can alter OPTCR or OPTCA which are relative and absolute optimality criteria. One can also manipulate work space allowing the solver more or less memory to operate within. Also, through a .OPT file one can put in specific items that affect the relative performance of the solver. For example, in CPLEX one can affect branching direction, tolerances, number of solutions, and whether or not the pre-solve is used. In OSL one can manipulate the type of solution method to be used, ,pre-solve usage, scaling, mixed integer strategy and many other factors. For the naive user probably the most important ones are the optimality criteria tolerances. Experienced users may find that they need to set a number of mixed integer and/or nonlinear

options.

Table 12.1 Example of Speedups Through Better Set Addressing

Panel A. Program before set index reordering

```
 4  option  profile=1;
 6  Set     a /1*10/
 7          B /1*10/
 8          C /1*10/
 9          D /1*10/
10          e /1*10/
11  parameter x(e,d,c,b,a);
12                    X(e,d,c,b,a)=10;
13  parameter z(a,b,c,d,e);
14  z(a,b,c,d,e)=x(e,d,c,b,a);
15  parameter y;
16                    Y=sum ((a,b,c,d,e),z(a,b,c,d,e)*x(e,d,c,b,a));
17  variables obj
18  Positive variables var(e,b,a);
19  equations objeq
20            R( b,c,d)
21            q(a,b,c);
22
23  objeq.. obj=e=sum((a,b,c,d,e),z(a,b,c,d,e)*x(e,d,c,b,a)*var(e,b,a));
24  r(b,c,d).. sum((a,e),Var(e,b,a))=l=sum((a,e),x(e,d,c,b,a)*z(a,b,c,d,e));
25  q(a,b,c).. sum((d,e),var(e,b,a)/x(e,d,c,b,a)*z(a,b,c,d,e))=l=20;
26  model slow /all/
27  option lp=bdmlp;
28  slow.workspace=13;
29  solve slow maximizing obj using lp;
30  parameter sumofvar;
31  sumofvar=sum((a,b,c,d,e),z(a,b,c,d,e)*x(e,d,c,b,a)*var.l(e,b,a));
```

Panel B. Program after set index reordering

```
 6  Set     a /1*10/
 7          B /1*10/
 8          C /1*10/
 9          D /1*10/
10          e /1*10/
11  parameter x(a,b,c,d,e);
12                    X(a,b,c,d,e)=10;
13  parameter z(a,b,c,d,e);
14  z(a,b,c,d,e)=x(a,b,c,d,e);
15  parameter y;
16                    Y=sum ((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e));
17  variables obj
18  Positive variables var(a,b,e);
19  equations objeq
20            R( b,c,d)
21            q(a,b,c);
22
23  objeq.. obj=e=sum((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e)*var(a,b,e));
24  r(b,c,d).. sum((a,e),Var(a,b,e))=l=sum((a,e),x(a,b,c,d,e)*z(a,b,c,d,e));
25  q(a,b,c).. sum((d,e),var(a,b,e)/x(a,b,c,d,e)*z(a,b,c,d,e))=l=20;
26  model slow /all/
27  option lp=bdmlp;
28  slow.workspace=13;
29  solve slow maximizing obj using lp;
30  parameter sumofvar;
31  sumofvar=sum((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e)*var.l(a,b,e));
```

Table 12.2 A looping Example

```
   4  option  profile=1;
   6  Set     a /1*10/
   7          B /1*10/
   8          C /1*10/
   9          D /1*10/
  10          e /1*10/
  11  parameter x(a,b,c,d,e);
  12  parameter z(a,b,c,d,e);
  13  parameter y;
  14  variables obj
  15  Positive variables var(a,b,e);
  16  equations objeq
  17            R( b,c,d)
  18            q(a,b,c);
  20  objeq.. obj=e=sum((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e)*var(a,b,e));
  21  r(b,c,d).. sum((a,e),Var(a,b,e))=l=sum((a,e),x(a,b,c,d,e)*z(a,b,c,d,e));
  22  q(a,b,c).. sum((d,e),var(a,b,e)/x(a,b,c,d,e)*z(a,b,c,d,e))=l=20;
  23  model slow /all/
  24  option lp=bdmlp;
  25  slow.workspace=13;
  26  set loops /1*2/
  27  parameter sumofvar;
  28  loop(loops,
  29       X(a,b,c,d,e)=10;
  30       z(a,b,c,d,e)=x(a,b,c,d,e);
  31       Y=sum ((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e));
  32       solve slow maximizing obj using lp;
  33       sumofvar=sum((a,b,c,d,e),z(a,b,c,d,e)*x(a,b,c,d,e)*var.l(a,b,e));
  34  );
```

Table 12.3 PROFILE information for example in Table 12.1

```
----       2 OTHER                    0.000     0.060 SECONDS
----       3 OTHER                    0.000     0.060 SECONDS
----       4 OTHER                    0.000     0.060 SECONDS
----      12 ASSIGNMENT X             1.530     1.590 SECONDS   100000
----      14 ASSIGNMENT Z             4.070     5.660 SECONDS   100000
----      16 ASSIGNMENT Y             2.910     8.570 SECONDS
----      28 ASSIGNMENT SLOW          0.000     8.570 SECONDS
----      29 SOLVE INIT SLOW          0.000     8.570 SECONDS
----      23 EQUATION   OBJEQ         7.470    16.040 SECONDS        1
----      24 EQUATION   R            13.130    29.170 SECONDS     1000
----      25 EQUATION   Q             9.220    38.390 SECONDS     1000
----      29 SOLVE FINI SLOW          1.320    39.710 SECONDS
----      29 GAMS FINI                2.200    42.020 SECONDS
----       1 EXEC-INIT                0.000     0.000 SECONDS
----      29 SOLVE READ SLOW          0.380     0.380 SECONDS
----      31 ASSIGNMENT SUMOFVAR      4.120     4.500 SECONDS
----      31 GAMS FINI                0.000     4.500 SECONDS
```

Table 12.4 Profile information for looping example when profile=1

```
----       2 OTHER                    0.000     0.050 SECONDS
----      24 OTHER                    0.000     0.050 SECONDS
----      25 ASSIGNMENT SLOW          0.000     0.050 SECONDS
loop 1
----      32 SOLVE INIT SLOW          0.000     6.970 SECONDS
----      20 EQUATION   OBJEQ         6.160    13.130 SECONDS        1
----      21 EQUATION   R             9.060    22.190 SECONDS     1000
----      22 EQUATION   Q             6.480    28.670 SECONDS     1000
----      32 SOLVE FINI SLOW          0.880    29.550 SECONDS
----       1 EXEC-INIT                0.000     0.000 SECONDS
solve report
----      32 SOLVE READ SLOW          0.390     0.390 SECONDS
loop 2
----      32 SOLVE INIT SLOW          0.000     5.990 SECONDS
----      20 EQUATION   OBJEQ         7.140    13.130 SECONDS        1
----      21 EQUATION   R            12.800    25.930 SECONDS     1000
----      22 EQUATION   Q             7.800    33.730 SECONDS     1000
----      32 SOLVE FINI SLOW          1.040    34.770 SECONDS
----      32 GAMS FINI                2.410    37.240 SECONDS
solve report
----      32 SOLVE READ SLOW          0.270     0.270 SECONDS
```

Table 12.5 Profile information for looping example when profile=1 and profiletol=1

```
loop 1
----      29 ASSIGNMENT X             1.810     2.090 SECONDS   100000
----      30 ASSIGNMENT Z             4.180     6.270 SECONDS   100000
----      31 ASSIGNMENT Y             1.640     7.910 SECONDS
----      32 SOLVE INIT SLOW          0.000     7.910 SECONDS
----      20 EQUATION   OBJEQ         6.270    14.180 SECONDS         1
                                      6.210              ROWGEN
                                      0.060              COLGEN
----      21 EQUATION   R            10.540    24.720 SECONDS      1000
                                      9.550              ROWGEN
                                      0.990              COLGEN
----      22 EQUATION   Q             6.590    31.310 SECONDS      1000
                                      6.540              ROWGEN
                                      0.050              COLGEN
----      32 SOLVE FINI SLOW          0.830    32.140 SECONDS
                                      0.060              ROWS
                                      0.710              COLUMNS
                                      0.060              OTHERS
solve report
----      32 GAMS FINI                2.030    34.170 SECONDS

----       1 EXEC-INIT                0.000     0.000 SECONDS
----      32 SOLVE READ SLOW          0.330     0.330 SECONDS
----      33 ASSIGNMENT SUMOFVAR      2.140     2.470 SECONDS
loop 2
----      30 ASSIGNMENT Z             1.260     4.280 SECONDS   100000
----      31 ASSIGNMENT Y             1.650     5.930 SECONDS
----      32 SOLVE INIT SLOW          0.000     5.930 SECONDS
----      20 EQUATION   OBJEQ         7.250    13.180 SECONDS         1
                                      7.250              ROWGEN
                                      0.000              COLGEN
----      21 EQUATION   R            11.640    24.820 SECONDS      1000
                                     10.430              ROWGEN
                                      1.210              COLGEN
----      22 EQUATION   Q             8.680    33.500 SECONDS      1000
                                      8.630              ROWGEN
                                      0.050              COLGEN
----      32 SOLVE FINI SLOW          1.370    34.870 SECONDS
                                      0.000              ROWS
                                      0.770              COLUMNS
                                      0.160              OTHERS
----      32 GAMS FINI                2.480    37.350 SECONDS
----       1 EXEC-INIT                0.000     0.000 SECONDS
solve report
----      32 SOLVE READ SLOW          0.330     0.330 SECONDS
----      33 ASSIGNMENT SUMOFVAR      2.200     2.530 SECONDS
```

Chapter 13. Verifying Data

Often after analyzing a model via the methods in Chapters 8 and 9 or other means, modelers suspect that there is something wrong with the data, the calculation process or the equation specification. Modelers may also release a model to a set of nontechnical users and may wish to have automatic procedures for verifying data. This chapter deals with ways to employ GAMS in an investigation of the model data. Here we cover:

a) common causes of bad data

b) investigating for the causes of bad data in the model equations

We will discuss each of these items below:

13.1 What is Wrong with the Data -- Some things to Check

GAMS capability to easily handle algebraic symbols over a wide variety of data is a double edged sword. While one can easily manipulate large blocks of data one also loses intimate knowledge of what goes on in many sub cases. There are some common places to look for errors that we discuss below.

13.1.1 Check for Completeness and Consistency of Input Data

One of the most common statements in computing is garbage-in garbage-out. Namely, if bad data are entered then it usually causes problems in computer programs and often the data are the limiting factor. One should go back and make sure that the input data set is complete and inconsistent and that the numbers in it match up with expectations. In section 13.2 we go through some procedures for checking out data completeness and consistency.

13.1.2 Check for the non dynamic calculation

A common mistake made during the execution of the GAMS program involves dynamic

versus non-dynamic calculations. In GAMS a dynamic calculation is performed every time a model is generated whereas a non-dynamic calculation is only performed once at the point it appears in the program execution. GAMS model definition statements, i.e., that in line #43 in the example below,

```
41      cost(i,j)=10+2*distance(i,j);
42      distance(i,j)=distance(i,j)/2;
43      obj.. Costtran=e=sum((i,j),cost(i,j)*shipment(i,j));
```

are dynamic in that they're done every time a SOLVE statement is executed. On the other hand, lines 41 - 42 are non-dynamic and the calculations therein only done once at the time that the statements are executed. The value of cost used in generating the line 43 equation in any subsequent SOLVE is the value dependent on distance as it existed before line 41 not as altered in line 42. In general, there are two ways to fix this

g) The modeler needs to make sure that any calculations used in the model equations are brought up to date when their input data have been revised. In the example, we could move or repeat line 41 after line 42.

h) The modeler can rewrite the model specification equations so that the calculations are embedded. Thus, we could revise line 43 as below to incorporate the cost dependence on the distance parameter into the equation making the calculation dynamic and automatically revised whenever a solve statement is executed.

```
43      obj.. Costtran=e=sum((i,j),(10+2*distance(i,j))*shipment(i,j));
```

13.1.3 Check for Inadvertent Multiplicative Sums

One phenomena we run into more often than we would like to admit is the multiplicative sum. In particular, we find instances where sums add terms whose definition does not include the

index summed over. For example, if the set N contains ten elements and one uses an equation of the form

res(j).. sum(N, a(j)*x(j)) =e= b(j);

or

Z= sum((j,N), a(j)*x(j))

then in either case all the a(j) coefficients would be multiplied by ten because they are repeatedly added for each element in the set N, and N is not an argument of either a(j) or x(j). Thus, if numerical investigations show the coefficients are higher by some multiple make sure the set indices being summed over are all part of the definitions of the mathematical sums therein. In both cases above the N sum is probably unnecessary and a specification error.

13.1.4 Included Irrelevant Terms

Often a modelers perception of relevant items in a model and GAMS perception are at odds. GAMS automatically considers every possible dimension for every item even though the modeler may not. For example (see file tranzero.gms), suppose one is setting up a transportation problem and in the data set transport costs are omitted for routes which cannot be used (i.e. one might exclude data on a place without a harbor between two places that must involve movement by ship). Suppose the model specification equations are:

```
46   TCOSTEQ.. TCOST =E= SUM((PLANT,MARKET), SHIPMENTS(PLANT,MARKET)*
47                                            COST(PLANT,MARKET));
48   SUPPLYEQ(PLANT)..   SUM(MARKET, SHIPMENTS(PLANT, MARKET))
49                                       =L= (SUPPLY(PLANT);
50   DEMANDEQ(MARKET)..  SUM(PLANT,  SHIPMENTS(PLANT, MARKET))
51                                       =G= (DEMAND(MARKET);
```

But this formulation would define shipment possibilities from every source to every destination. This will include variables representing the omitted routes which have zero transport costs and a solution will likely involve movements across omitted routes. What one needs to do is either

a) constrain all variables which do not have transportation costs to be zero using a command like

shipments.fx(plant,market)$(cost(plant,market) eq 0)=0;

b) modify the model specification to conditionally omit the irrelevant variables using $ commands as follows.

```
52   SUPPLYEQ2(PLANT)..  SUM(MARKET$cost(plant,market)
53                           ,SHIPMENTS(PLANT, MARKET))=L= SUPPLY(PLANT);
54   DEMANDEQ2(MARKET).. SUM(PLANT$cost(plant,market),
55                            SHIPMENTS(PLANT,MARKET)) =G= DEMAND(MARKET);
```

which only allows the shipment variable into the model when cost has been defined. Note a condition is not needed in the objective function since the multiplication by a zero cost does not yield a coefficient (also GAMS will only consider nonzero entries so will skip over that case)

13.1.5 Check for Partially eliminated variables

The above case raises yet another problem which often plagues modelers. One must be careful to include the same conditionals for variable existence everywhere. Suppose we had the transport model from the last section plus one additional constraint (tranpart.gms) which omits the conditional on the shipments variable as follows:

```
41   TCOSTEQ.. TCOST =E= SUM((PLANT,MARKET), SHIPMENTS(PLANT,MARKET)*
42                                             COST(PLANT,MARKET));
43   SUPPLYEQ(PLANT)..  SUM(MARKET$cost(plant,market)
44                        ,SHIPMENTS(PLANT, MARKET))=L= SUPPLY(PLANT);
45   DEMANDEQ(MARKET).. SUM(PLANT$cost(plant,market),
46                           SHIPMENTS(PLANT,MARKET)) =G= DEMAND(MARKET);
47   minships(plant)..      sum(market,shipments(plant,market))
48                               =g= minship(plant);
```

The new constraint imposes a minimum level of outgoing shipments. Solution of the resultant model involves nonzero quantities of the shipment variable for the omitted routes since the conditional in the last constraint does not omit them. Note these variables would only exist in the

last constraint.

Modelers should insure that when a $ condition is included for a variable, that the same $ condition appears in association with all appearances of the variable. Otherwise, cases can arise where the variable appears in the model even though the user feels it should be eliminated. This can also be rectified by fixing the variable at zero as discussed in the previous section.

One other consideration involves the use of complex $ conditions. In one of the models that we work with there is a nine dimensional variable which represents harvest time for trees. Conditions on existence of this variable depend on a number of data items including nonzero wood yields, available acreage, harvest costs etc. This makes for a complex $ condition which needs to be repeatedly included. Given the possibility for error in including and revising that condition, we chose to enter a calculated set called ISEXIST with the same dimensions as the variable and define set elements only for the relevant cases. In turn the model was setup to generate the variable conditional only on that parameter. Suppose we illustrate this in the transport case (tranexst.gms) where we wish the variable to exist only for:

1) cases where there are nonzero distance when the source and destination differ
2) cases where there are zero distances where the source and destination are the same place
3) cases where the distance is less than 5000 miles.

Suppose we then compute a set (yestran) which tells when a route exists and condition our model on it.

```
31   yestran(plant,market)$((cost(plant,market) gt 0
32                                  and distance(plant,market) lt 5000)
33                                or (sameas(plant,market) ))=yes;
43   TCOSTEQ.. TCOST =E= SUM(yestran(plant,market),
44                    SHIPMENTS(PLANT,MARKET)*COST(PLANT,MARKET));
45   SUPPLYEQ(PLANT)..  SUM(yestran(plant,market)
```

```
46                            ,SHIPMENTS(PLANT, MARKET))=L= SUPPLY(PLANT);
47   DEMANDEQ(MARKET).. SUM(yestran(plant,market),
48                             SHIPMENTS(PLANT,MARKET)) =G= DEMAND(MARKET);
```

Note in this case we are not using conditionals but are rather using the set as an index to sum over. We could have expressed the TCOSTEQ sum as

SUM((plant,market)$yestran(plant,market) .....

The approaches are equivalent.

13.1.6 Calculation Specification Mistakes

Naturally the GAMS code can be incorrect coded with mathematical mistakes in laying out calculations or equations. For example calculations might have conditionals messed up, parentheses entered in the wrong places, terms exponentiated rather than multiplied, added rather than multiplied, divided rather than added, etc. The next section provides procedures for discovering problems of this nature.

13.2 Procedures to discover Problems

Unfortunately, when one has found some improper calculation results in a program numbers one cannot always easily recognize the underlying problem. In the section below we outline a number of numerical ways of finding problems.

13.2.1 Basic approach

The basic approach to finding a problem in GAMS is a mixture of isolating faulty code sections and displaying inputs and outputs. Suppose one had an equation which computed transport cost as a parameter laden function of distance. What we would do is print out the names of the sources, names of the destinations, the distances, and the cost function parameters, as well as the calculated costs. Then for cases of interest results we might

1) hand verify the calculation

2) check the input data making sure that source and destination are proper, that there is a well defined distance, and that the parameters the cost function are appropriate.

This is done using a GAMS displays for the input items along with a mixture of display statements or the procedures in chapter 8 for model equations, i.e., GAMSCHK and/or LIMROW/LIMCOL .

Our methodology for finding the cause of model data problems reverts back to the small to large arguments in chapter five.

13.2.1.1 Small to Large Strategies to Check Out Subcases

The easiest way to check data is to examine a small set as it passes through the program manipulations making sure all is proceeding according to expectations. One can do this by a) specifying a small version of the large data set as in chapter 5, b) by using subsets, or c) by temporarily zeroing out most of the data in the original array. Let us illustrate the later two cases.

The example in Table 13.1(transml.gms) illustrates this. In that transportation model we specify a 100x100 problem and compute distances using a peculiar function. Later we might find this function causes problems and set out to discover the difficulty. Initially we would define a smaller problem or as the example shows use subsets. Note in lines 6-9 we define smaller sets. Then in lines 11-13 we compute cost only for the restricted set. On the other hand it may be more convenient to temporarily reduce the amount of data being used by zeroing a lot of it out as in lines 16-18 (lines we would remove after debugging). Notice under either case the effective distance matrix is a three by three, even though it started out as a hundred by a hundred.

One also could have simply modified the calculations in the program so that rather than using the set “origin” they used the set “smallorig” and “smalldest” instead of “destinat” (transet.gms). This is often convenient as one can alter the definitions of the restricted set to the full set using commands like

smalldest(destinat)=yes;

smallorig(origin)=yes;

to restore the data set to full size after debugging.

In any case we are now ready to look for the problem.

13.2.1.2 Example of Checking Out a Calculation

Given a set of problem data, hopefully a small one and the indication of a symptom that there is something wrong with a calculation one proceeds to display the input and output. In the example, suppose we had investigated the solution and found the objective function contained negative costs of shipment. We then would investigate the calculation of cost in lines 19 and 22 where we print out the input data and output. These items are reproduced below.

```
----     19 PARAMETER DISTANCE
             d3          d44          d99
o4      230.000     2280.000     5030.000
o47    -200.000     1850.000     4600.000
o91    -640.000     1410.000     4160.000

----     22 PARAMETER COST
             d3          d44          d99
o4      463.000     4563.000    10063.000
o47    -397.000     3703.000     9203.000
o91   -1277.000     2823.000     8323.000
```

An investigation of that output would quickly reveal the negative distances as the problem and lead one to investigate the calculation thereof and find the flaws in the calculation in lines 4-5.

```
----     19 PARAMETER DISTANCE

             d3          d44          d99
o4      230.000     2280.000     5030.000
o47    -200.000     1850.000     4600.000
```

```
o91    -640.000    1410.000    4160.000


----      22 PARAMETER COST

              d3          d44          d99
o4       463.000    4563.000   10063.000
o47     -397.000    3703.000    9203.000
o91    -1277.000    2823.000    8323.000
```

13.2.2 Check out data via calculation

One need not just display tables and hunt for problems visually. GAMS can be used to do calculations which verify the accuracy of the data. For example, consider the statements in lines 23-27 of Table 13.1. Here we define an array and compute its entries so that it contains faulty distances as identified by screening criteria that fill the array with the distance number whenever that distance is negative or in excess of 4,000 miles.

One can pursue similar automated data checks to reveal bad interrelationships between data items, or missing data. For example in the above model, (the remaining examples in this section are all in the file trandem.gms) if we also had a parameter with demand by destination the statements below would tell us where we had a demand location which had no incoming transportation links (where we define a link only if there is nonzero distance).

```
29  set isolated(destinat) isolated demand locations;
30  isolated(destinat)$(demand(destinat) gt 0 and
31                  sum(origin,abs(distance(origin,destinat))) le 0)
32                 =yes;
33 display isolated;
```

We also can use calculations to look for missing or extraneous data. The statements below indicate where we have destinations we can ship to which do not exhibit nonzero demand

indicating either excess data in the transport matrix or missing demand data.

```
33  set nodemand(destinat) destinations without demand;
34  nodemand(destinat)$(demand(destinat) eq 0 and
35                  sum(origin,abs(distance(origin,destinat))) gt 0)
36                 =yes;
37 display nodemand;
```

We now arrive at the general point of this section. One can enter calculations based on a screening rule to automatically verify either large or small data sets and display the results in a diagnostic format. One can also limit the output of such checks by using code such as the following:

```
37  if (sum(nodemand(destinat),1),display nodemand);
38  if (sum(isolated(destinat),1),display isolated);
```

which will only print out the data problems when they are present.

13.2.3 Code simplification to find problems

If the results from a calculation are turning out unexpectedly, one can eliminate large parts of the equation and just examine sub components. For example if one takes a transportation model which contains conditionals to eliminate unneeded variables and removes the Cost(i,j) term and just leaves the shipment variable then one can examine which variables pass that conditional statement by looking at the LIMROW, LIMCOL GAMSCHK output on the particular equations.

Also in more complex calculations it is often desirable to eliminate terms so as to isolate the problem. Table 13.2 contains a thirty-four line calculation drawn from the ASM report writer as in chapter five. If this calculation was malfunctioning one might want to focus on particular subsets of the terms. For example the sum starting in line 2 goes on through line 33 but one might wish to only consider say the first part of it which is line 3 though 16. This could be accomplished

by doing one of several things. First, one could go in line 17, 24, 29, and 34 and multiply each of these terms by a zero which would eliminate those terms from consideration. Second, one could put an $ONTEXT before line 17 and an $OFFTEXT after line 34 (although one would have to make sure the ending parentheses in line 33 was kept active). Third, one could put asterisks in front of line 17 through 33 reducing them to comments (again having to preserve the closing parentheses). One also can use strategies investigating which terms pass the conditionals by for example surrounding lines 4-5 with a 1$(..) then multiplying line 7 and 11 by zero which would yield a sum which is the count of how many crop , tech and wtech alternatives passed the conditional and make sure that some cases passed.

Another strategy one can use when examining a complex calculation is to move some of the sums into arrays. Thus rather summing in lines 3 through 16 over crop, wtech and tech one could just define an array which had those three dimensions and add in the term in the sum then display the results. In turn this would allow one to examine each case.

In closing complex terms may be better examined by:

1) simplifying elements using text, surrounding by 1$(..) or by using zero times.
2) when conditionals are present use something that only puts a simple value like a one to see how often the conditionals are met.
3) compute intermediate terms into parameters so you can examine them to see what the problems are.

13.2.4 Focusing in on Problematic Areas

Sometimes GAMS programs get quite slow and can take a long time to run. There are several ways to focus and check on only particular code segments. In particular, let's suppose that

we have a relatively large model and we are questioning a particular calculation that occurs a substantial time after the program has begun. Two strategies are possible here,

1) One could take the GAMS model, put displays before and after the calculation and let the model reexecute and proceed, or

2) One can take the existing model up to right before the calculation (or at some later more desirable stage) and save the execution status using the GAMS save/restart features, then just execute the calculation all by itself again with displays of inputs and outputs.

The second approach is the one we commonly use although one has to be careful to make sure that the input data of the calculation have not changed between the point at which the calculation was originally made in the program and the end of the program where the reap harvest are. Thus, for example we could take and save the program that we have in Table 13.1 and then just recalculate whatever individual parameters that we wanted assuming the data hasn't changed and if we wanted to solve the model execute a solve statement. This would result for example in a perhaps a two statement file that GAMS was restarted with. The first one of which provided a revised cost calculation the second one of which was the solve statement. One can also then isolate the calculation of interest trying out different alternatives without needing to execute the whole rest of the GAMS program.

13.3 Tracing How Model Data are used -- Cross Reference Lists

When one finds bad data in a GAMS model one needs to be careful to see that all manipulations done on that data are appropriate. One may, for example, have a module of the code which one is forgotten about which did the small to large simplification and eliminated a lot

of the data and that may inadvertently be included. It is then important when one finds bad data to refer to some form of cross-reference map or use text editing to do a global search on a variable and trace through how that variable is manipulated during the course of the program. This can be difficult to do in cases because the module may include files and save/restart can make one overlook items. Two approaches are recommended for looking at this. One of which is using the cross-reference map the other of which involves using a systematic cross-reference map that was designed for multi-part program. Each is defined below.

13.3.1 Cross Reference Map

A fundamental tool in diagnosing models is the cross reference list. One can use the command $OFFSYMLIST OFFSYMXREF to suppress a cross reference list or $ONSYMLIST ONSYMXREF to turn it on. (If you do not use either of the commands, the default will suppress a cross reference list automatically.) The cross reference list, an example of which is given in GAMSMENU. This cross reference list only covers the current module and does not expand across different components when restart files are used. Nevertheless, it matches up with the list file and allows one to examine the definition and use of all GAMS model symbols in the current model. One can also use the undocumented RF option on the job control file to save a cross reference across modules in a file.

13.3.2 GAMSMAP

The cross-reference map is not very helpful when one is dealing with large multi-part models such as the ASM model as discussed in Chapter five. As a consequence we have developed a program called GAMSMAP, shows the modules where items appear. Instructions for GAMSMAP appear in Appendix III of this manual. The ASM output of GAMSMAP (Table 13-3

Panel A-D) shows each module in which data are manipulated and then shows where the data are used in terms of calculations and model generation. As a consequence when one finds bad data one can go back and trace through the modules in which that data are calculated and make sure no inadvertent things are being done in portions of the code one isn't directly looking at because of included statements.

Table 13.1 Example model for Small to Large Reduction

```
 1  set     origin          /o1*o100/
 2          destinat        /d1*d100/;
 3  parameter distance(origin,destinat);
 4          distance(origin,destinat)
 5              =120+50*ord(destinat)-10*ord(origin);
 6  set     smallorig(origin) small set of origins for testing
 7                                    /o4,o47,o91/
 8          smalldest(destinat) small set of destinations
 9                                    /d3,d44,d99/;
10  parameter cost(origin,destinat);
11      Cost(origin,destinat)
12                $(smallorig(origin) and smalldest(destinat) )
13                    =3+2*distance(origin,destinat);
14 display cost;
15 *          Temporarily remove data from distance
16          distance(origin,destinat)
17                $(   (not smallorig(origin))
18                  or (not smalldest(destinat)))=0;
19  display distance;
20            Cost(origin,destinat)$distance(origin,destinat)
21                   =3+2*distance(origin,destinat);
22  Display cost;
23  parameter baddist(origin,destinat) bad distances where below zero and above 4000;
24      Baddist(origin,destinat)$(  distance(origin,destinat) lt 0 or
25                                  distance(origin,destinat) gt 4000)
26                                  =distance(origin,destinat);
27  display baddist;
```

Table 13.2 Complex Calculation

```
 1 BALANCEP(PRIMARY,"PRODUCTION") =
 2       SUM(SUBREG,
 3              SUM((CROP,WTECH,TECH)$FARMPROD("SLIPPAGE",CROP),
 4                     CROPBUDGET.L(SUBREG,CROP,WTECH,"NONPART",TECH) *SCALPROD
 5                            *CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"NONPART",TECH)$
 6                            (CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"NONPART",TECH) GT 0)
 7                     + CROPBUDGET.L(SUBREG,CROP,WTECH,"PARTICIP",TECH)  *SCALPROD
 8                            *CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"NONPART",TECH)
 9                            *(1.0-FPPART(SUBREG,CROP))$
10                            (CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"NONPART",TECH) GT 0)
11                     + CROPBUDGET.L(SUBREG,CROP,WTECH,"PARTICIP",TECH)
12                            *SCALPROD
13                            *CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"PARTICIP",TECH)
14                            *FPPART(SUBREG,CROP)
15                            * (1-FARMPROD("PERCNTPAID",CROP))$
16                            (CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"PARTICIP",TECH) GT 0) )
17              + SUM((CROP,WTECH,TECH)$(FARMPROD("SLIPPAGE",CROP) GT 0.0 AND
18                                    FARMPROD("FPYIELD",CROP) LE 1.0),
19                     CROPBUDGET.L(SUBREG,CROP,WTECH,"PARTICIP",TECH)
20                            *SCALPROD *FPPART(SUBREG,CROP)*FARMPROD("PERCNTPAID",CROP)
21                            *CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"PARTICIP",TECH)
22                            *(1.0-FARMPROD("FPYIELD",CROP))$
23                            (CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"PARTICIP",TECH) GT 0) )
24              + SUM((CROP,WTECH,TECH)$(FARMPROD("SLIPPAGE",CROP) LE 0.),
25                     CROPBUDGET.L(SUBREG,CROP,WTECH,"BASE",TECH)
26                     *SCALPROD
27                            *CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"BASE",TECH)$
28                            (CBUDDATA(PRIMARY,SUBREG,CROP,WTECH,"BASE",TECH) GT 0)    )
29              + SUM((ANIMAL,LIVETECH),
30                     LVSTBUDGET.L(SUBREG,ANIMAL,LIVETECH)
31                            *SCALLIVE
32                            *LBUDDATA(PRIMARY,SUBREG,ANIMAL,LIVETECH)$
33                            (LBUDDATA(PRIMARY,SUBREG,ANIMAL,LIVETECH) GT 0)   )     )
34       +DEFPRODN.L(PRIMARY)$FARMPROD("SLIPPAGE",PRIMARY)*SCALE(PRIMARY);
```

Table 13.3 Sample Abstract Output from GAMSMAP

Panel A Part of File gamsout

```
----List of Parameters which are Computed
Parameter    File where values are computed
 CBUDDATA   ALLOFIT.GMS
```

```
 PDEMAND    ASMCALRN.93
 PEXPORT    ASMCALRN.93
 SDEMAND    ASMCALRN.93
 SEXPORT    ASMCALRN.93
 CCCBUDDATA ASMCALSU.93
 FARMPROD   ASMCALSU.93
 LBUDDATA   ASMCALSU.93
 LIM        ASMMODEL.GMS
 AUMSSUM    ASMREPT.GMS
 BALANCEP   ASMREPT.GMS
 BALANCES   ASMREPT.GMS
 CROPSUBREG ASMREPT.GMS
 CROPSURP   ASMREPT.GMS
 LANDSUM    ASMREPT.GMS
 LIVESUBREG ASMREPT.GMS
 NATINPUSE  ASMREPT.GMS
 PCONSUR    ASMREPT.GMS
 WELSUM     ASMREPT.GMS
 CONVERGE   ASMSOLVF.GMS
 FARMPROD   ASMSOLVF.GMS
 RESULT     ASMSOLVF.GMS
 TOL        ASMSOLVF.GMS
 TOLER      ASMSOLVF.GMS
 CRPLANDDIF CRP.SML
 LANDAVAIL  CRP.SML
 SLANDAVAIL CRP.SML
 REGMIXDATA NATMIX.SML
 SCALE      SETS.SML


Panel B Part of file gamsmap.sc0

List of items which appear ill defined

FPSCENARIO **** no use
        ref in FPDATA.SML
FPTECH     **** no use
        ref in SETS.SML
LANDTWO    **** no use
        ref in SETS.SML
NFPTECH    **** no use
        ref in SETS.SML


List of all files used in the program

   ALLOFIT.GMS
   ASMCALRN.93
   ASMCALSU.93
   ASMMODEL.GMS
   ASMREPT.GMS
   ASMSEPER
```

```
   ASMSOLVF.GMS
   CROP.SML
   CRP.SML
   DEMAND.SML
   EROSION.SML
   FPDATA.SML
   LIVE.SML
   MIX.SML
   NATMIX.SML
   PROC.SML
   REPTSETS.93
   SETS.SML
```

Panel C Part of file gamsmap.sc1

Files where actions on SETS appear

ITEM NAME	DECLARED	DEFINED	ASSIGNED	CONTROL	REF
ALLCOM	REPTSETS.93		REPTSETS.93	ASMREPT.GMS	ASMREPT.GMS
ALLI	SETS.SML	SETS.SML		SETS.SML	SETS.SML
				REPTSETS.93	REPTSETS.93
				FPDATA.SML	FPDATA.SML
				ASMCALSU.93	PROC.SML
				ALLOFIT.GMS	CROP.SML
				ASMREPT.GMS	LIVE.SML
					ASMCALSU.93
					ALLOFIT.GMS
					ASMMODEL.GMS
					ASMREPT.GMS
ALLSUBREG	SETS.SML	SETS.SML			SETS.SML
					CRP.SML

Files where actions on PARAMETERS appear

ITEM NAME	DECLARED	DEFINED	ASSIGNED	REF
AUMSSUBREG		REPTSETS.93		**** no use
AUMSSUM	REPTSETS.93		ASMREPT.GMS	ASMREPT.GMS
AUMSSUP	DEMAND.SML	DEMAND.SML		ASMCALSU.93
				ASMMODEL.GMS
				ASMCALRN.93
				ASMREPT.GMS
BALANCEP	REPTSETS.93		ASMREPT.GMS	ASMREPT.GMS
BALANCES	REPTSETS.93		ASMREPT.GMS	ASMREPT.GMS

Files where actions on EQUATIONS appear

ITEM NAME	DECLARED	DEFINED	ASSIGNED	REF
ARTIFICIAL	ASMMODEL.GMS	ASMMODEL.GMS		ASMMODEL.GMS
				ASMREPT.GMS
AUMSCONVEX	ASMMODEL.GMS	ASMMODEL.GMS		ASMMODEL.GMS
AUMSIDENT	ASMMODEL.GMS	ASMMODEL.GMS		ASMMODEL.GMS
AUMSR	ASMMODEL.GMS	ASMMODEL.GMS		ASMMODEL.GMS
				ASMREPT.GMS

```
--------------------------------------------------------------------------------
 Files where actions on VARIABLES  appear
   ITEM NAME DECLARED      DEFINED     ASSIGNED          REF
--------------------------------------------------------------------------------
    ARTIF      ASMMODEL.GMS                           ASMMODEL.GMS
                                                      ASMREPT.GMS
    ARTS       ASMMODEL.GMS                           ASMMODEL.GMS
                                                      ASMREPT.GMS
    AUMSPRIV   ASMMODEL.GMS              ASMMODEL.GMS ASMMODEL.GMS
                                         ASMCALRN.93  ASMREPT.GMS
--------
    SECTOR     ASMMODEL.GMS ASMMODEL.GMS ASMSOLVF.GMS ASMSOLVF.GMS
--------------------------------------------------------------------------------
Items worked on in file ASMCALRN.93
 SETS
 Item Name   DEFINED   DECLARED   ASSIGNED   IMPL-ASN    CONTROL        REF
  ANIMAL                                                   X            X
  CROP                                                     X            X
  CRPMIXALT                                                X            X
  CTECH                                                    X            X
  LANDTYPE                                                 X            X
  LIVETECH                                                 X            X

 PARAMETERS
 Item Name   DEFINED   DECLARED   ASSIGNED   IMPL-ASN    CONTROL        REF
  AUMSSUP                                                              X
  CBUDDATA                                                             X
  LABORSUP                                                             X
  LANDSUPPL                                                            X
  LBUDDATA                                                             X
  MIXDATA                                                              X
  NATMIXDATA                                                           X
  PDEMAND                          X                                   X
  PEXPORT                          X                                   X
  PIMPORT                                                              X

 VARIABLES
 Item Name   DEFINED   DECLARED   ASSIGNED   IMPL-ASN    CONTROL        REF
  AUMSPRIV                         X
  CROPBUDGET                       X
  DEMANDP                          X
  DEMANDS                          X
  EXPORTP                          X
  EXPORTS                          X
  HIRED                            X
  IMPORTP                          X
  IMPORTS                          X

--------------------------------------------------------------------------------



Panel D Output of file gamsmap.sc2

##----List of Parameters given input values
Parameter    File where values are entered
```

```
QINC        ASMSEPER
SEPAG       ASMSEPER
CCCBUDDATA  CROP.SML
CRPMOVE     CRP.SML
CRPSTUFF    CRP.SML
YESCONV     CRP.SML
AUMSSUP     DEMAND.SML
INPUTPRICE  DEMAND.SML
LABORSUP    DEMAND.SML
LANDAVAIL   DEMAND.SML
LANDSUPPL   DEMAND.SML
PDEMAND     DEMAND.SML
PEXPORT     DEMAND.SML
PIMPORT     DEMAND.SML
POPULATION  DEMAND.SML
SDEMAND     DEMAND.SML
SEXPORT     DEMAND.SML
SIMPORT     DEMAND.SML
WATERSUP    DEMAND.SML
EROSION     EROSION.SML
FARMPROD    FPDATA.SML
FPPART      FPDATA.SML
LBUDDATA    LIVE.SML
MIXDATA     MIX.SML
NATMIXDATA  NATMIX.SML
PROCBUD     PROC.SML
BASEOUTPUT  SETS.SML
CONVERGE    SETS.SML
SCALLIVE    SETS.SML
SCALMIX     SETS.SML
SCALOBJ     SETS.SML
SCALPROC    SETS.SML
SCALPROD    SETS.SML
TOL         SETS.SML
```

Chapter 14 Improving GAMS Output

Many users generate their GAMS output solely through use of the display statement, others use put capabilities, yet others generate output into spreadsheets, do graphing and or provide output that can be accessed through the web. In this chapter we review topics regarding output via GAMS. In a later chapter (17) we present information on interfacing GAMS with other packages. First, we discuss report writing. Then, we address how one might make the displays more effective changing their ordering and reformatting displays. We then go on and provide material on put files both as an alternative to displays and as a way of accessing information from other programs.

14.1 Report Writing

Users of GAMS may be frustrated by their perceived need to take the LST file solution report and do a number of hand calculations on it. However, this is totally unnecessary. GAMS provides procedures whereby one can use information generated in the solution part of the calculations. Consider the following lines that are from ASMREPT.GMS that is included in chapter 5.

```
1       BALANCEP(PRIMARY,"PROC-USE") =
2               +SUM(NONFEED$(PROCBUD(PRIMARY,NONFEED) GT 0),
3                     PROCESS.L(NONFEED)*PROCBUD(PRIMARY,NONFEED)) ;
4       BALANCEP(PRIMARY,"DOM-DEMAND") =   DEMANDP.L(PRIMARY) ;
5       BALANCEP(PRIMARY,"EXPORT") = EXPORTP.L(PRIMARY) ;
6       BALANCEP(PRIMARY,"PRICEX100") =  PRIMARYBAL.M(PRIMARY) * 100.;
```

In statement one, processing usage by primary commodities is added into a primary commodity supply demand balance table called balancep. This processing use involves the optimal levels of the processing variables (PROCESS.L) times data from the processing budgets (PROCBUD) as computed in line 3. We also place the optimal levels of the primary demand (DEMANDP.L) and export (EXPORTP.L) variables into our report through the code in lines 4 and 5. Line 6 adds in the shadow price from the primary commodity balance (PRIMARYBAL.M)times a hundred so we

can display item with zero decimal places and get numbers down to the penny.

In general one can freely use report writing calculations like this to summarize the results even suppressing the GAMS solution listing and only working from the summary reports.

14.2 Making Displays More Effective

GAMS displays can be enhanced in terms of form, content and ordering. GAMS often orders things in a fashion that one dislikes and also has an undesirable format in terms of decimal places, omissions of zeros, etc.

14.2.1 Display Element Ordering

Many users are frustrated with the ordering of set elements in the output. This can be changed. First, however it is worthwhile knowing the rules that the ordering follows. To understand the way GAMS orders display output, one has to understand the way that set elements are stored. GAMS uses something called the unique element list (UEL) to store set elements. This is a single list of all set elements. The elements enter that list in the order of their appearance and that is the order in which they will appear in the output. This means if we have the sets ONE and TWO with the elements below

```
set     ONE   /A,C,B,Total/
Set     TWO   /D,A,F,TOTAL/
```

and a parameter defined over the set TWO as follows,

```
Parameter       item(two) /D 1,A 3, F 5/;
item("total")=sum(two$(not sameas(two,"total")),item(two));
```

when displayed (DISPSET.GMS) will exhibit the order as follows

```
A    3.000,   Total 9.000,   D    1.000,   F    5.000
```

where the elements "A" and "Total" appear before "D" without regard to the order in which the

set TWO was specified The reason is that “A” and “Total” appeared in set ONE and their names were seen and put into the UEL before “D” and “F”.

One can try to take control of this in several ways. First, if the sets do not overlap other than say in the use of some summary items like TOTAL one can change the name of the TOTAL set element for example redefining set TWO and item as

```
Set      TWO   /D,A,F,TOTAL2/
Parameter      item(two) /D 1,A 3, F 5/;
item(“total2”)=sum(two$(not sameas(two,”total2”)),item(two));
```

Under those circumstances (DISPSET2.GMS) the display is reordered to be

```
A    3.000,   D    1.000,   F    5.000,   TOTAL2 9.000
```

with the TOTAL2 element at the end.

Second, one can introduce in effect a UEL set which contains all elements in the order desired using set definitions as follows

```
set      UELORDER  /A,B,C,D,F,Total/
set      ONE       /A,C,B,Total/
Set      TWO       /D,A,F,TOTAL/
```

In that case a display of the set ONE (DISPSET3.GMS) is reordered to be

```
A  ,  B  ,  C  ,  Total
```

and ITEM becomes

```
A    3.000,   D    1.000,   F    5.000,   Total 9.000
```

where the appearance is governed by the order of appearance in the first set seen – UELORDER.

One may find this procedure difficult as it may be difficult to always include all items. However the compiler can be enlisted as an aid by using a subsetting strategy. If the definition above is slightly modified so sets ONE and TWO are subsets of UELORDER

```
set     UELORDER          /A,B,C,D,F,Total/
set     ONE(UELORDER)     /A,C,B,Total/
Set     TWO(UELORDER)     /D,A,F,TOTAL/
```

then including an item in set TWO without including that item in UELORDER generates compiler errors.

Sometimes the desired order depends on the item being displayed and more desperate measures may be in order. In this case the authors sometimes define an ordering set and include that set in the definition of the items. Consider the more complex example (dispset4.gms) below

```
 1  Set     numberordr             /1*100/
 2  set     One                   /A,B,C,D,F,Average,ITEMS/
 3  set     order1(numberordr,One) /2.(A,D,F),1.Average,3.(B,C)/
 4  Set     order2(numberordr,One) /2.(A,D,F),99.Average,1.(C,B)/
 5  Set     PQ /Price,Quantity/
 6  Table Items(One,PQ)
 7                  Price   Quantity
 8          A       2       9000
 9          B       6       3000
10          C       2.5     4000
11          D       2.1     3000
12          F       2.4     1.90;
13  items("Average",PQ)=    SUM(One,items(one,pq))
14                          /sum(one,1$items(one,pq));
15  parameter items1(numberordr,one,pq) item ordered first way;
16          items1(numberordr,one,pq)
17                  =sum(order1(numberordr,one),items(one,pq));
18  parameter items2(numberordr,one,pq) item ordered second way;
19          items2(numberordr,one,pq)
20                  =sum(order2(numberordr,one),items(one,pq));
21  display items,items1,items2;
```

Here we have defined the set NUMBERORDR to provide the ordering help. In turn we define alternative orders with respect to the ONE set in lines 3 and 4. Then we copy the information in from the items array according to the two orders in lines 17 and 18 using parameters with an extra dimension for the ordering set. Displays of the original data mirror the order in the One set.

```
----     21 PARAMETER ITEMS
                Price    Quantity
A               2.000    9000.000
B               6.000    3000.000
C               2.500    4000.000
D               2.100    3000.000
F               2.400       1.900
Average         3.000    3800.380
```

However the displays of the three dimensional sets are ordered first according to the

NUMBERORDR set since it is the first index in the parameter definition and it appeared first in the program so it is first in the UEL list in GAMS.

```
----     21 PARAMETER ITEMS1        item ordered first way
                Price    Quantity
1.Average       3.000    3800.380
2.A             2.000    9000.000
2.D             2.100    3000.000
2.F             2.400       1.900
3.B             6.000    3000.000
3.C             2.500    4000.000

----     21 PARAMETER ITEMS2        item ordered second way
                 Price    Quantity
1 .B             6.000    3000.000
1 .C             2.500    4000.000
2 .A             2.000    9000.000
2 .D             2.100    3000.000
2 .F             2.400       1.900
99.Average       3.000    3800.380
```

Note that in these statements the order is primarily controlled by the numberordr set but when several elements have the same value for that set, then the ordering in the other sets (ONE in this case) determines how the items appear. Sometimes this ordering is a simple as providing items like TOTAL with a high number and all others with a low one.

14.2.2 Controlling the Ordering of the Parameter Indices As they Appear

An additional item that GAMS users may be frustrated with involves the ordering of the way the indexes are referenced in a display statement. Considering, the small example below(dispord.gms)

```
 1  set index1 /index11*index12/
 2  set index2 /index21*index22/
 3  set index3 /index31*index32/
 4  set index4 /index41*index42/
 5  parameter data(index1,index2,index3,index4);
 6  data(index1,index2,index3,index4)=2;
 7  display data;
 8  option data:0:1:3;display data;
 9  option data:0:3:1;display data;
10  option data:0:0:4;display data;
11  option data:0:2:2;display data;
12  parameter data2(index2,index4,index1,index3);
13  data2(index2,index4,index1,index3)
14         = data(index1,index2,index3,index4);
15  option data2:0:2:2;display data2
```

Here in this example we have the four dimensional parameter “data” defined over the sets index1 through index4. When a display occurs, what will happen regardless of the formatting in that display is that the data for index four for vary most rapidly while the data for index one will vary least rapidly. GAMS always varies the display from the right most stated element to the left most element. Also by default if a parameter is defined with respect to four or more sets it will presented with the right most set defining the columns in the display, the next two from the right put in the row, and then the fourth and higher elements from the right will be altered one at a time with a display for each item. See for example the LST file portion reproduced in table 14.1, in which the display from line 7 of the above example appears. Here note that the index4 items are in the columns and are varied the fastest, then index3 and index2 and finally index1. Also since this is a four dimensional array we have a display section for the first element of index1, then a section for the second element of index one.

Users may not find this display style consistent with what they want. The GAMS OPTION statement permits one to alter this. In particular, an option statement of the form

OPTION itemname:decimals:row indices:column indices;

can modify the display formatting. Namely, when this option is executed it causes all subsequent displays involving the named item to have the specified decimal places with the number of set items allocated to rows and columns as specified . Three alternative examples appear in table 14.1, notice in the line 8 display we have only the first index in the row definition but the second, third and fourth are used in the column definition. Again note as one looks across the column, the fourth index is the one varying the fastest than the third, and there is one row for each element of the third. The second formatting alternative is in table 14.1 as controlled by the option in line 9

where we have three indices varied in the rows and one in the columns. A third alternative is given in line 10 where data are displayed without any row and column distinctions because we allocated zero of them to the rows and all to the columns. This is a particularly convenient way to lay items out when one wants to use a text editor to clip data out and include them in subsequent GAMS runs using a parameter statement.

There are several notes on display options that merit mention. First, if one has a five-dimensional array but specifies less than five dimensions in the option command i.e. indicating two by two then the extra dimensions will be used as in the index notation in the output associated with line 7 of table 14.1. Thus one can specify less than the full dimensionality. But if one specifies more than the full dimensionality a GAMS error arises. Second, the decimal control does have nuances that will be discussed in the formatting section below.

Another feature which on occasion is frustrating involves the ordering of the way the Sets are varied. Sometimes one might wish because the first index position to be varied fastest or some other progression other that right to left. The only way to achieve this in GAMS is to create a new parameter which has a new ordering of subscripts and copy the data over. An example of this is given in lines 12-15 of the above example. The implications of this for the output order can be seen by comparing lines 11 and 15 where because the indices have been reordered they come out in a different order in the table display.

14.2.3 Reformatting the Appearance of Numbers

Yet another potential frustration with the output from GAMS displays involves numerical formatting. Consider the example (numb.gms) which appears in table 14.2. There we define the table DATA with rather disparate numbers. A resultant display of DATA yields

```
----      8 PARAMETER DATA
            index21     index22
index11 1.000000E-5 1.000000E+7
index12       3.720     200.100
```

In that display GAMS mixes together exponential and normal format by default trying to print out three decimal places. If such a display is unsatisfactory, there are several ways of altering its appearance.

1) We can alter the number of decimals using the global option statement for example reducing the default decimals in subsequent displays to one

```
option decimals=1;
```

yielding

```
----     10 PARAMETER DATA
              index21     index22
index11 1.000000E-5  10000000.0
index12         3.7       200.1
```

In turn, we still get the exponential display for the number that is of the order ten to the minus 5, but we don't get the exponential display for the large number as it fits.

2) Users may wish to suppress small numbers in the display. For example, using

```
data2(index1,index2)$(data2(index1,index2) lt 0.01)=0;
```

sets all numbers to zero which are less than .01. Note one needs to employ absolute value if negative numbers are present i.e. using a command like:

```
data2(index1,index2)$(abs(data2(index1,index2)) lt 0.01)=0;
```

yielding

```
----     14 PARAMETER DATA2
            index21     index22
index11                10000000.0
index12          3.7       200.1
```

3) Users may wish to cap the value of large numbers. We can cause the output to have the number infinity for anything greater than the number 10,000 using

```
data2(index1,index2)$(data(index1,index2) gt  10000)=inf;
```

yielding

```
----     18 PARAMETER DATA2
              index21     index22
index11 1.000000E-5        +INF
index12       3.720     200.100
```

4) Users may wish to round numbers using syntax like

```
data2(index1,index2)=round(data(index1,index2),0);
```

. 5) Users may desire a report of percentage changes. These first need to be calculated in a manner such as

```
24  data4(index1,index2)$data2(index1,index2)=
25    100*(data3(index1,index2)/data2(index1,index2)-1);
26  data4(index1,index2)$(abs(data4(index1,index2)) lt 0.1)=0;
27  data4(index1,index2)$(data2(index1,index2) eq 0)= na;
```

Here we are careful to use absolute values so negative changes are not zeroed out. We also report numbers that would report percentage changes from a base of zero with the coding “na”.

```
----     29 PARAMETER DATA4
              index21     index22
index11           NA
index12         80.6         1.5
```

6) Local reformatting can be done on the decimal places for a table using the OPTION display modifier as discussed in 14.2.2 or as implemented in line 29 where we display the item data4 with one decimal place even though the active option for decimals was set to three in line 15.

14.2.4 Reformatting Item Name Case and Appearance

Another item of concern in displays may be the appearance of the set names, parameter names etc in terms of case. Users may wish the output to be in mixed upper and lower case or all

in upper case facilitating inclusion through a word processor into a report. Version 2.50 and higher of GAMS preserve element case in the output. However, users should note that

1) The case form of an element such as a set name that is copied to the output follows the conventions used elsewhere in GAMS in terms of the unique element list as discussed in section 14.2.1. Namely, the case used is the first one encountered. Thus in our examples above using the word "total" the output had whatever case sensitive spelling of Total was first encountered in the GAMS program. Thus one must insure that the appearance wanted in reflected in the first appearance of the item in the program.

2) Internally GAMS does not distinguish with respect to case. Thus

```
item("TOTAL")
item("total")
item("TotaL")
```

would all refer to the same thing.

3) In GAMS 2.50 and greater indices can be up to 31 characters long and may enclose special character spaces, etc. What one does when one wants such an index is put it in quotes for example the following are valid set elements

"Number of items"or "Money Spent ($)".

14.2.5 Controlling Page Size and Width

Users may wish to exercise control over page width and length. The default page length is usually 60 lines which means that longer file contain a lot of GAMS headers in inconvenient spots. One can do is respecify page size in two ways.

1) Job specific page characteristics are specified on the GAMS call using the

syntax

GAMS modelname ps=n1 pw=n2

GAMS trnsport ps=9999 pw=100

where ps gives the page length and can be as small as 30 or as big as 9999 (may be bigger) and pw the page width which must be between 72 and 255.

2) The default page length and width can be altered by editing a file in their GAMS source directory. In particular on Windows 95 machines the file is called GMSPRM95.doc, or Gamsparm.doc and on NT and UNIX machines it has the same format gmsprmnt.doc and gmsprmun.doc respectively. In this file one defines the page length with a line ps followed by a space and a number

ps 1000

and page width with

pw number.

This alters the page size and page width for all future applications of GAMS using that GAMS source. There are a number of other parameters that can be altered through this mechanism as described in the file gamsparm.doc.

14.3 Controlling Output Volume

Often users of GAMS find that they get an awful lot more output than they want. Generally, in applications these authors use a suite of three commands which greatly limit the amount of output one gets from a model. These commands are as follows:

```
option limrow=0;
option limcol=0;
$offsymlist offsymxref
```

They cause GAMS to suppress the listing of the equations, variables, symbol list and symbol cross-reference map. These may be useful output in some cases but ordinarily the authors find that turning all these off is the appropriate first action especially so new users can find their output. Another command which can be used is

option solprint=off;

which suppresses the variable and equation solution output. This is ordinarily only used when calculated reports are being used which summarizes the output. One can also just get the nonzero solution for key variables by displaying variablename.L , equationname.M or variablename.M.

Another output management scheme involves the use of the command

$offlisting

All lines after this command until either the end of the file (or the end of an include file) or a $onlisting is found are not copied to the LST file. This often is used to help suppress the size of the listing, where one for example doesn't necessarily want a big long listing of a raw data file.

Another output management scheme we commonly use is as follows. Develop a code with a substantial amount of report writing calculation and possible solution output. After executing this code use the GAMS save and restart capability and save the program status. Then restart with a piece of code which displays selected items. If additional items are desired, then display them. This permits one to focus on only a few pieces of output but still have the full set available. Such a procedure is partially present in the ASM model example used in chapter 5 where ASMREPT.GMS computes a lot of reports but it does not print them all out.

One final note, when the solution is suppressed or many solves are being used in a loop and report writing done on the output, it is useful to have a table or display of the solver status to

insure that infeasible or unbounded or other types of solution irregularities did not occur. One can do this by displaying or saving the parameter modelname.modelstat i.e. for a model named agfor run in a loop

solstat(run)=agfor.modelstat;

In turn a display will show values of 1 or 2 for successful solves (the GAMS model details these results).

14.4 Including Slacks In the Output

GAMS unlike the rest of the mathematical programming world includes equation "levels" in its output not slacks. An equation level for the equation $AX \leq b$ is the term AX whereas a slack is b-AX. Users desiring slacks can get them by using the command

option solslack=1;

14.5 Moving Beyond Display to Put

A lot more control over the output can be achieved by using the GAMS put commands. However, with this control comes some cost. Usage of the put commands requires more technical programming. Those with the latest release of the GAMS users guide will find a rather long chapter on put which covers the technical language elements. Thus we will not cover those here rather we will just present examples and overview discussion of put usage.

The first example we would like to show is a combination of report writing features and the put. It appears in table 14.3 this is an excerpt from the file called invest2.gms and is an excerpt that puts a report out regarding some investments. The resultant put file that arises from(with a few blank line spaces removed) appears in table 14.4.

Let us examine the "notable assumption" section of that output. Line number 2 is

generated by lines 509 and 510 of table 14.3 where the calculated scalar “length” is put out as a 5 digit result with 2 decimal places. Similarly, line 513 and 514 put out the scalars that identify the last month, day and year that define the investment plan as input in the associated invest.gms file. Put commands can mix messages and calculated values into sentence like output.

Also note in the later part of the output on the “investment plan” that this mixes set element definition names for the security issuer and three maturity date columns with calculated parameters. Also note we can control the decimal places by item

In general the PUT command allows generation of much more highly formatted reports that can be used with decision makers or in published reports.

The put command can also be used to format data for export into other programs. Table 14.5 and 14.6 illustrate such cases. In 14.5 we have a excerpt of a larger program that we use to dump data to a statistics program. We used this in a setting where we ran hundreds of alternative runs then put out solution summary results in a fixed format that a regression code could be set up to read. Panel B gives a sample of the output that we generated. Note the resultant data set has the run name appearing first, then data in a fixed format.

Table 14.6 presents a similar case where data was dumped to a mapping program which read data in comma delimited form. In this case we were able to use the PUT output control feature for passing data to a spreadsheet. Namely, line 24 tells GAMS to use output control option 5. The resultant output is in panel B. In that file GAMS automatically surrounded all text elements with quotes and put a comma between all elements. These data are also in perfect shape to be imported by a many program. For example, if one imports this as a text file in Lotus 123 and tells LOTUS to parse on commas, then the data fits into the spreadsheet perfectly. In doing this

users do need to make sure that there are labels for each column in the spreadsheet. Thus in the put statements in lines 26 and 30, we put the word “region” to label the set names of the regions.

Many other features of put files could be discussed. We will only note a couple more aspects(Readers should study GAMS manual and the example on their web page -- under the documentation section of www.gams.com).

1) When including data related to multiple set elements in a row, one uses loop commands like in lines 26 and 28.

2) Put only skips to a new line when it finds a /. Thus multiple puts may define a single line of output. For example in line 28, we first put out the region name and then loop over a put that enters the data items. Finally we end the line with a slash (/).

3) Through these procedures GAMS allows one to put out either the set and variable names or the associated text along with those, this can be helpful in some cases as longer explanatory text can be included (using the te syntax).

The put command is the best way to insure high quality output while staying within GAMS. It is also the best general way to move data from GAMS to other.

One other comment is that if one wants to put data say in the table editor in WordPerfect, this can be achieved by moving the data through the put command to a spreadsheet and then importing the data from the spreadsheet to the wordprocessor.

14.6 Interfacing with other Programs

The use of the PUT capability as well as other GAMS capabilities allows more advanced interfaces with other programs and computers elsewhere. These topics are covered in Chapter 17.

Table 14.1 Alternative Display index options

```
----      7 PARAMETER DATA  – no option
INDEX 1 = index11
                          index41     index42
       index21.index31      2.000       2.000
       index21.index32      2.000       2.000
       index22.index31      2.000       2.000
       index22.index32      2.000       2.000
INDEX 1 = index12
                          index41     index42
       index21.index31      2.000       2.000
       index21.index32      2.000       2.000
       index22.index31      2.000       2.000
       index22.index32      2.000       2.000

----      8 PARAMETER DATA – option allows 1 row by 3 columns (:0:1:3)
            index21     index21     index21     index21     index22     index22
            index31     index31     index32     index32     index31     index31
            index41     index42     index41     index42     index41     index42
index11           2           2           2           2           2           2
index12           2           2           2           2           2           2

      +     index22     index22
            index32     index32
            index41     index42
index11           2           2
index12           2           2


----      9 PARAMETER DATA – option allows 3 row by 1 columns (:0:3:1)
                          index41     index42
index11.index21.index31           2           2
index11.index21.index32           2           2
index11.index22.index31           2           2
index11.index22.index32           2           2
index12.index21.index31           2           2
index12.index21.index32           2           2
index12.index22.index31           2           2
index12.index22.index32           2           2

----     10 PARAMETER DATA – option allows 0 row by 5 columns (:0:0:5)
index11.index21.index31.index41 2,    index11.index21.index31.index42 2
index11.index21.index32.index41 2,    index11.index21.index32.index42 2
index11.index22.index31.index41 2,    index11.index22.index31.index42 2
index11.index22.index32.index41 2,    index11.index22.index32.index42 2
index12.index21.index31.index41 2,    index12.index21.index31.index42 2
index12.index21.index32.index41 2,    index12.index21.index32.index42 2
index12.index22.index31.index41 2,    index12.index22.index31.index42 2
index12.index22.index32.index41 2,    index12.index22.index32.index42 2


----     11 PARAMETER DATA – option allows 2 row by 2 columns (:0:2:2)
                    index31     index31     index32     index32
                    index41     index42     index41     index42
index11.index21           2           2           2           2
index11.index22           2           2           2           2
index12.index21           2           2           2           2
index12.index22           2           2           2           2

----     15 PARAMETER DATA2 – option allows 2 row by 2 columns (:0:2:2)
                    index11     index11     index12     index12
                    index31     index32     index31     index32
index21.index41           2           2           2           2
index21.index42           2           2           2           2
index22.index41           2           2           2           2
index22.index42           2           2           2           2
```

Table 14.2 Display Number Formatting

```
 1  set index1 /index11*index12/
 2  set index2 /index21*index22/
 3  table data(index1,index2)
 4            index21   index22
 5  index11   0.00001    10000000
 6  index12     3.72       200.1;
 7
 8  display data;
 9  option decimals=1;
10  display data;
11  parameter data2(index1,index2);
12  data2( index1,index2)=data(index1,index2);
13  data2(index1,index2)$(data2(index1,index2) lt 0.01)=0;
14  display data2;
15  options decimals=3;
16  data2( index1,index2)=data(index1,index2);
17  data2(index1,index2)$(data(index1,index2) gt  10000)=inf;
18  display data2;
19  parameter data3(index1,index2);
20  parameter data4(index1,index2);
21  data2( index1,index2)=data(index1,index2);
22  data2(index1,index2)$(data2(index1,index2) lt 0.01)=0;
23  data3( index1,index2)=3+data2(index1,index2);
24  data4(index1,index2)$data2(index1,index2)=
25    100*(data3(index1,index2)/data2(index1,index2)-1);
26  data4(index1,index2)$(abs(data4(index1,index2)) lt 0.1)=0;
27  data4(index1,index2)$(data2(index1,index2) eq 0)= na;
28  display data3;
29  option data4:1:1:1;display data4;
```

Table 14.3 Put file Example for Improved Formatting

```
503  file investp
504  put investp
505  put 'Notable Assumptions' //;
506  scalar length;
507  length=365.25/12;
508  Put '1.  All Issues face value of $1,000 with no minimum investment '//
509  Put '2.  All maturity month limits done in days using 365.25 days per year '/
510  put '                        or  365.25/12 = ' length:5:2 ' days per month'//
511  put '3.  All security share restrictions done in terms of face value -- '/
512  put '            not purchase price' //
513  Put '4.  Portfolio value Maximized on Last day -- date' lastmonth:3:0
514            lastday:3:0 lastyear:5:0 //
515  Put '5.  Portfolio acquired starting on ' firstmonth:3:0
516            firstday:3:0 firstyear:5:0 //
517  Put '6.  All proceeds from Maturing Securities plus any initial unused cash'/
518  put '        placed in money market at interest rate of' annual:5:2//
520  Put 'Overall Results' //
521  Put '     Initial Funds                        ' Funds:10:0 //
522  Put '     Value of Ending Portfolio            ' endvalue.l:10:0//
523  parameter ratio;
524  ratio=endvalue.l/funds;
525  parameter dailyror;
526  dailyror=ratio**(1/(daylast-dayfirst))-1;
527  ratio=ratio*100;
528  parameter annualror;
529  annualror=((1+dailyror)**365.25-1)*100;
530  Put '     Percentage Increase in Portfolio Value' ratio:10:4 //
531  Put '     Rate of Return                       ' annualror:10:4  /// /
532  parameter portvalue
533            facevalue
534            share
535            mmonth    maturity month
536            cumul     cumulative share
537            portage  average maturity;
538           cumul=0;
539           portage=0;
540  portvalue=
541  sum((security,month,day,year)$invest.l(security,month,day,year),
542        invest.l(security,month,day,year)*
543        returndata(security,month,day,year,"facevalue"))/1000000;
544  Put 'Investment Plan'//
545  put '         '    @23 '           Number of          Face    Port.  Cumul' /
546   put '  Security ' @23 ' Maturity  Months to Number   Value   Share  Share' /
547   put '  Issuer'    @23 '   Date    Maturity  Bought   Mill$    (%)    (%) ' /
548   scalar maxage /0/
549          share18 /0/;
550   loop(year,
551    loop(month,
552     loop(day,
553      loop(security$invest.l(security,month,day,year),
554            facevalue=
555                   invest.l(security,month,day,year)*
556                   returndata(security,month,day,year,"facevalue")/1000000;
557            mmonth=(returndata(security,month,day,year,"maturity day")
558                    -dayfirst)/(365.25/12);
559            portage= portage+facevalue*mmonth/portvalue;
560            share=facevalue/portvalue*100;
561            maxage=max(maxage,mmonth);
562            if(mmonth gt 18,share18=share18+share);
563            cumul=cumul+share;
564            put security.tl:20 ' '
565                monthn(month):3:0 dayn(day):3:0 yearn(year):5:0 mmonth:7:2
566                invest.l(security,month,day,year):9:0
567                facevalue:9:3 share:8:2 cumul:7:2  /
568       ))));
569   put 'Total                                    ' portvalue:16:3 '  100.00' /;
570   put 'Average Maturity              ' portage:8:2 '  ' //;
```

Table 14.4 Output of Put File in Table 14.3

Notable Assumptions

1. All Issues face value of $1,000 with no minimum investment
2. All maturity month limits done in days using 365.25 days per year
   or 365.25/12 = 30.44 days per month
3. All security share restrictions done in terms of face value --
   not purchase price
4. Portfolio value Maximized on Last day -- date 6 17 2003
5. Portfolio acquired starting on 6 19 1998
6. All proceeds from Maturing Securities plus any initial unused cash
   placed in money market at interest rate of 5.25

Overall Results

Initial Funds	40000000
Value of Ending Portfolio	52390860
Percentage Increase in Portfolio Value	130.9771
Rate of Return	5.5524

Investment Plan

Security Issuer	Maturity Date	Number of Months to Maturity	Number Bought	Face Value Mill$	Port. Share (%)	Cumul Share (%)
GE COMM. PAPER	7 21 1998	1.05	4000	4.000	10.01	10.01
AMERICAN EXPRESS CP	8 21 1998	2.07	4000	4.000	10.01	20.01
MERRILL LYNCH CP	8 21 1998	2.07	4000	4.000	10.01	30.02
FORD MOTOR CREDIT CP	9 21 1998	3.09	4000	4.000	10.01	40.03
TOYOTA CP	3 21 1999	9.03	3986	3.986	9.97	50.00
INTL PAPER	7 10 2000	24.71	2251	2.251	5.63	55.63
CATERPILLAR	7 10 2001	36.70	4000	4.000	10.01	65.64
J.P. MORGAN	1 15 2002	42.91	4000	4.000	10.01	75.64
MORGAN ST. DEAN	8 1 2002	49.41	4000	4.000	10.01	85.65
SALOMON BROS	1 15 2003	54.90	4000	4.000	10.01	95.66
SEARS ROEBUCK ACC	3 20 2003	57.00	1736	1.736	4.34	100.00
Total				39.973	100.00	
Average Maturity		24.00				

Table 14.5 Put file to Dump Data to regression program

Panel A PUT Instructions

```
loop(run,
  put run.tl;
   if(redone(run) gt 0,put 'r');put @12;
loop(fpitems,put fpscen(run,fpitems):13:3);
put /;
loop(decwant,s=  fawelsum( "Agconswelf",decwant,run)/1000;put s:13:0;);
put /;
loop(decwant,s=  fawelsum( "Agprodwelf",decwant,run)/1000;put s:13:2;);
put /;
);
```

Panel B Sample of Data saved

```
FARMPR121r         2.000       33.000        0.000        0.000        0.000        1.000
         1159         1240         1335         1441
        34.82        27.79        32.43        34.28
         1297         1383         1500         1628
FARMPR122r         2.000       33.000        0.000        0.000      875.000        1.000
         1159         1241         1333         1439
        35.07        27.02        32.56        34.38
         1296         1383         1498         1626
FARMPR123r         2.000       33.000        0.000        0.000     1750.000        1.000
         1159         1241         1329         1433
        35.07        26.92        33.03        38.61
         1296         1383         1495         1624
```

Table 14.6 Put file for Export to Mapping Program

Panel A PUT Instructions

```
   1  sets meas /nitrogen,phosporous,potassium,cropland,
   2                    watererosn,winderosn,sediment,pub-water,pumpwater,
   3                    chemicalco/;
   4  sets region /EAST,STHEAST,MIDWEST,WEST,STHCENTRAL,NORTHERNPL  /
   5  table data(region,meas) data to be put
   6                  nitrogen   phosporous  potassium  chemicalco  cropland
   7  EAST               0.96      -0.17       0.52      -0.24       0.00
   8  STHEAST            0.13       0.09       0.13      -0.12       0.02
   9  MIDWEST            0.40       0.36       0.54      -0.03       0.36
  10  WEST               1.74       1.51       1.73       0.59       1.63
  11  STHCENTRAL        -0.09      -0.15       0.04       0.17       0.12
  12  NORTHERNPL         3.55       1.70       2.59       3.16       1.65
  13    +              watererosn winderosn sediment pub-water pumpwater
  14  EAST              -1.14      0.01     -1.16      0.00    -10.71
  15  STHEAST            3.13      0.67      3.64      0.34      0.00
  16  MIDWEST           -0.23      0.59     -0.23      0.00     -1.11
  17  WEST               0.57      0.02      0.74      0.01      0.26
  18  STHCENTRAL        -0.16     -1.33     -0.08      0.00     -1.55
  19  NORTHERNPL         0.92     -3.06      0.85      0.00     -0.07
  21  file mapdat2;
  22  put mapdat2;
  23  mapdat2.pw=250;
  24  mapdat2.pc=5;
  25  set s1(meas) /nitrogen,phosporous,potassium,chemicalco,cropland/
  26  put 'region'; loop(s1,put  s1.tl ); put /;
  27  loop(region,
  28     put  region.tl ; loop(s1,put  data(region,s1):10:2); put /);
  29  set s2(meas) / watererosn,winderosn,sediment,pub-water,pumpwater/
  30  put 'region'; loop(s2,put   s2.tl ); put /;
  31  loop(region,
  32     put  region.tl ; loop(s2,put  data(region,s2):10:2); put /);
```

Panel B Put Output

```
"region","nitrogen","phosporous","potassium","cropland","chemicalco"
"EAST",0.96,-0.17,0.52,0.00,-0.24
"STHEAST",0.13,0.09,0.13,0.02,-0.12
"MIDWEST",0.40,0.36,0.54,0.36,-0.03
"WEST",1.74,1.51,1.73,1.63,0.59
"STHCENTRAL",-0.09,-0.15,0.04,0.12,0.17
"NORTHERNPL",3.55,1.70,2.59,1.65,3.16
"region","watererosn","winderosn","sediment","pub-water","pumpwater"
"EAST",-1.14,0.01,-1.16,0.00,-10.71
"STHEAST",3.13,0.67,3.64,0.34,0.00
"MIDWEST",-0.23,0.59,-0.23,0.00,-1.11
"WEST",0.57,0.02,0.74,0.01,0.26
"STHCENTRAL",-0.16,-1.33,-0.08,0.00,-1.55
"NORTHERNPL",0.92,-3.06,0.85,0.00,-0.07
```

Chapter 15 Sensitivity Analysis

Occasionally users are interested in getting sensitivity analysis information from GAMS often in the form of LP ranging analysis results. Unfortunately, the base version of GAMS does not yield such information. The user wishing such information has two alternatives. First, one may cause the model to be repeatedly solved varying a parameter and examine the results. Second, one can use solver dependent features of GAMS (which currently work with OSL or CPLEX) and retrieve the ranging information. In this chapter we cover how to obtain such information.

15.1 Obtaining Ranging Analyses From the GAMS Solvers

GAMS provides a document through its web-page (www.gams.com) which gives instructions on how to get ranging analyses when using the OSL or CPLEX solvers. Use of this approach requires one to implement an options file telling the solver to generate all possible selected ranging information. There is also an option one can use which causes the ranging information to be saved in an auxiliary file importable by GAMS, subject to some potential small editing changes. An example of the usage of these features is given in Table 15.1. In particular, suppose that we use our transportation model again and in using the transportation model we invoke OSL as the solver (line 53). We also need to activate the options file (line54) and instruct GAMS to write a dictionary file of type 4(line 55). Then within an options file named OSL.opt (or CPLEX.opt if using CPLEX), we enter the commands

```
rhsrng
objrng
```

This in turn causes OSL to do the right-hand side and objective function ranging and the information in Table 15.2 are generated in the list file.

It may not only be desirable to have such information in the list file thus GAMS also provides a way of generating the sensitivity information on a subset of the model equations and variables. In particular, if rather than wanting the ranging information on all equations the information is only wanted for selected items, one uses the syntax in the options file like that below

```
rhsrng supplyeq,demandeq
objrng shipments
```

in which case only the named equations (supplyeq,demandeq) and named variables (shipments) will have ranging information generated for them.

Finally the user may desire to use ranging information in GAMS output reports or calculations. This requires several additional steps.

c) Enter a line in the option file which specifies the name of a file in which GAMS importable code containing the ranges will be written.

```
rhsrng
objrng
rngrestart trnsport.rng
```

where in this case the name is trnsport.rng. The ranging file that is written by the GAMS solver is in Table 15.3.

d) Stop execution of the original program at this point and save a GAMS restart file. This can be accomplished using a command like the following.

```
gams tranrng -s sav
```

e) Construct a continuation file starting up form the restart file using a command like

gams addrang -r sav

In that file include the range file and a definition of the set rnglim as follows

```
set rnglim /lo,up/
```

In our example this would mean we construct a file like the following

```
set rnglim /lo,up/
$include trnsrnge.rng
display demandeqr,shipmentsr;
```

where the last line display the ranges. One could also use the ranges in calculations as with any other piece of GAMS data.

Note in Table 15.3 there is one flaw in the GAMS procedure and this is that the variable and equation names are simply augmented by the three letter code RNG and due to the current ten character limitation we end up with names which are in excess of ten characters. When directly included this file caused compilation errors. However we simply reduced the names down to acceptable lengths and continued shortening the RNG just to a R.

15.2 Automatic Sensitivity Analyses Using Looping Features

Given GAMS' capability to solve related problems in loops another way one can do sensitivity analyses is by proposing alternative scenarios. Consider the risk model in table 15.4. In this model suppose we wish to see how sensitive the solution is to alternative risk aversion coefficients. We do this by defining a set of risk aversion coefficients (lines 45-47) and then defining a loop (lines 50-62) which replaces the active risk aversion coefficients then solve the model and save information for a report. The procedure for repeated solution and report writing will be more adequately discussed in chapter 16. For now let us just say that the basic approach to this form of sensitivity analyses is to set up data for scenarios, then run the alternative scenarios and report on the solutions.

Table 15.1 Listing of file employing Sensitivity Analysis

```
 1  *                          DATA DEFINITION
 2
 3  SETS  PLANT    PLANT LOCATIONS
 4              /NEWYORK  , CHICAGO , LOSANGLS/
 5        MARKET   DEMAND MARKETS
 6                 /MIAMI,   HOUSTON, MINEPLIS, PORTLAND/
 7
 8  PARAMETERS   SUPPLY(PLANT)  QUANTITY AVAILABLE AT EACH PLANT
 9                 /NEWYORK   100, CHICAGO   275, LOSANGLS   90/
10               DEMAND(MARKET)   QUANTITY REQUIRED BY DEMAND MARKET
11                  /MIAMI     100, HOUSTON    90,
12                   MINEPLIS  120, PORTLAND   90 /;
13
14   TABLE   DISTANCE(PLANT,MARKET)   DISTANCE FROM EACH PLANT TO EACH MARKET
15
16                         MIAMI   HOUSTON   MINEPLIS   PORTLAND
17           NEWYORK      1300     1800       1100       3600
18           CHICAGO      2200     1300        700       2900
19           LOSANGLS     3700     2400       2500       1100
20
21        ;
22
23  *                          DATA CALCULATION
24
25   PARAMETER COST(PLANT,MARKET)    CALCULATED COST OF MOVING GOODS;
26             COST(PLANT,MARKET) = 50 + 1 * DISTANCE(PLANT,MARKET);
27
28  *                          MODEL DEFINITION
29
30   POSITIVE VARIABLES
31           SHIPMENTS(PLANT,MARKET) AMOUNT SHIPPED OVER A TRANSPORT ROUTE;
32   VARIABLES
33           TCOST                   TOTAL COST OF SHIPPING OVER ALL ROUTES;
34   EQUATIONS
35           TCOSTEQ                 TOTAL COST ACCOUNTING EQUATION
36           SUPPLYEQ(PLANT)         LIMIT ON SUPPLY AVAILABLE AT A PLANT
37           DEMANDEQ(MARKET)        MINIMUM REQUIREMENT AT A DEMAND MARKET;
38
39   TCOSTEQ..            TCOST =E=
40                          SUM((PLANT,MARKET), SHIPMENTS(PLANT,MARKET)*
41                                              COST(PLANT,MARKET));
42
43   SUPPLYEQ(PLANT)..    SUM(MARKET, SHIPMENTS(PLANT, MARKET))
44                                          =L= SUPPLY(PLANT);
45
46   DEMANDEQ(MARKET)..   SUM(PLANT,  SHIPMENTS(PLANT, MARKET))
47                                          =G= DEMAND(MARKET);
48
49   MODEL TRANSPORT /ALL/;
50
51  *                          MODEL SOLUTION
52   display cost;
53  option lp=osl;
54  transport.optfile=1;
55  transport.dictfile=4;
56  SOLVE TRANSPORT USING LP MINIMIZING TCOST;
```

Table 15.2 GAMS Output for Sensitivity Analysis

EQUATION NAME	LOWER	CURRENT	UPPER
TCOSTEQ	-INF	0	+INF
SUPPLYEQ(NEWYORK)	100	100	+INF
SUPPLYEQ(CHICAGO)	210	275	+INF
SUPPLYEQ(LOSANGLS)	90	90	+INF
DEMANDEQ(MIAMI)	0	100	100
DEMANDEQ(HOUSTON)	0	90	155
DEMANDEQ(MINEPLIS)	0	120	185
DEMANDEQ(PORTLAND)	0	90	90

VARIABLE NAME	LOWER	CURRENT	UPPER
SHIPMENTS(NEWYORK, MIAMI)	-1350	0	900
SHIPMENTS(NEWYORK, HOUSTON)	-500	0	+INF
SHIPMENTS(NEWYORK, MINEPLIS)	-400	0	+INF
SHIPMENTS(NEWYORK, PORTLAND)	-2500	0	+INF
SHIPMENTS(CHICAGO, MIAMI)	-900	0	+INF
SHIPMENTS(CHICAGO, HOUSTON)	-1350	0	500
SHIPMENTS(CHICAGO, MINEPLIS)	-750	0	400
SHIPMENTS(CHICAGO, PORTLAND)	-1800	0	+INF
SHIPMENTS(LOSANGLS, MIAMI)	-2400	0	+INF
SHIPMENTS(LOSANGLS, HOUSTON)	-1100	0	+INF
SHIPMENTS(LOSANGLS, MINEPLIS)	-1800	0	+INF
SHIPMENTS(LOSANGLS, PORTLAND)	-1150	0	1800
TCOST	2.22e-016	1	+INF

Table 15.3 File automatically Generated by GAMS with ranging information

```
PARAMETER  TCOSTEQRNG(RNGLIM) /
LO -INF
UP +INF
/;
PARAMETER  SUPPLYEQRNG(PLANT,RNGLIM) /
NEWYORK.LO                  100
NEWYORK.UP +INF
CHICAGO.LO                  210
CHICAGO.UP +INF
LOSANGLS.LO                  90
LOSANGLS.UP +INF
/;
PARAMETER  DEMANDEQRNG(MARKET,RNGLIM) /
MIAMI.LO 0
MIAMI.UP                 100
HOUSTON.LO 0
HOUSTON.UP                  155
MINEPLIS.LO 0
MINEPLIS.UP                 185
PORTLAND.LO 0
PORTLAND.UP                  90
/;
PARAMETER  SHIPMENTSRNG(PLANT,MARKET,RNGLIM) /
NEWYORK.MIAMI.LO               -1350
NEWYORK.MIAMI.UP                900
NEWYORK.HOUSTON.LO              -500
NEWYORK.HOUSTON.UP +INF
NEWYORK.MINEPLIS.LO              -400
NEWYORK.MINEPLIS.UP +INF
NEWYORK.PORTLAND.LO             -2500
NEWYORK.PORTLAND.UP +INF
CHICAGO.MIAMI.LO               -900
CHICAGO.MIAMI.UP +INF
CHICAGO.HOUSTON.LO              -1350
CHICAGO.HOUSTON.UP                500
CHICAGO.MINEPLIS.LO              -750
CHICAGO.MINEPLIS.UP               400
CHICAGO.PORTLAND.LO             -1800
CHICAGO.PORTLAND.UP +INF
LOSANGLS.MIAMI.LO              -2400
LOSANGLS.MIAMI.UP +INF
LOSANGLS.HOUSTON.LO             -1100
LOSANGLS.HOUSTON.UP +INF
LOSANGLS.MINEPLIS.LO             -1800
LOSANGLS.MINEPLIS.UP +INF
LOSANGLS.PORTLAND.LO             -1150
LOSANGLS.PORTLAND.UP              1800
/;
PARAMETER  TCOSTRNG(RNGLIM) /
LO 0
UP +INF
/;
```

Table 15.4 Looping Sensitivity Analysis Example

```
 2   OPTION LIMCOL = 0; OPTION LIMROW = 0;
 3   SETS STOCKS POTENTIAL INVESTMENTS / BUYSTOCK1*BUYSTOCK4 /
 4        EVENTS EQUALLY LIKELY STATES OF NATURE /EVENT1*EVENT10 / ;
 5   ALIAS (STOCKS,STOCK);
 6   PARAMETERS     PRICES(STOCKS) PURCHASE PRICES OF THE STOCKS
 7                                / BUYSTOCK1   22
 8                                  BUYSTOCK2   30
 9                                  BUYSTOCK3   28
10                                  BUYSTOCK4   26 / ;
11   SCALAR      FUNDS    TOTAL INVESTABLE FUNDS / 500 / ;
12   TABLE RETURNS(EVENTS,STOCKS) RETURNS BY STATE OF NATURE EVENT
13              BUYSTOCK1   BUYSTOCK2   BUYSTOCK3   BUYSTOCK4
14     EVENT1      7           6           8           5
15     EVENT2      8           4          16           6
16     EVENT3      4           8          14           6
17     EVENT4      5           9          -2           7
18     EVENT5      6           7          13           6
19     EVENT6      3          10          11           5
20     EVENT7      2          12          -2           6
21     EVENT8      5           4          18           6
22     EVENT9      4           7          12           5
23     EVENT10     3           9          -5           6
24   PARAMETERS
25      MEAN (STOCKS)       MEAN RETURNS TO X(STOCKS)
26      COVAR(STOCK,STOCKS) VARIANCE COVARIANCE MATRIX;
27   MEAN(STOCKS) = SUM(EVENTS , RETURNS(EVENTS,STOCKS) / CARD(EVENTS) );
28   COVAR(STOCK,STOCKS)
29      = SUM (EVENTS ,(RETURNS(EVENTS,STOCKS) - MEAN(STOCKS))
30                     *(RETURNS(EVENTS,STOCK)- MEAN(STOCK)))/CARD(EVENTS);
31    DISPLAY MEAN , COVAR ;
32    SCALAR RAP   RISK AVERSION PARAMETER / 0.0 / ;
33    POSITIVE VARIABLES    INVEST(STOCKS)  MONEY INVESTED IN EACH STOCK
34    VARIABLE              OBJ             NUMBER TO BE MAXIMIZED ;
35    EQUATIONS             OBJJ            OBJECTIVE FUNCTION
36                          INVESTAV        INVESTMENT FUNDS AVAILABLE;
37   OBJJ..
38   OBJ =E=   SUM(STOCKS, MEAN(STOCKS) * INVEST(STOCKS))
39           - RAP*(SUM(STOCK, SUM(STOCKS,
40                        INVEST(STOCK)* COVAR(STOCK,STOCKS) *INVEST(STOCKS))));
41    INVESTAV..     SUM(STOCKS, PRICES(STOCKS) * INVEST(STOCKS)) =L= FUNDS ;
42    MODEL EVPORTFOL /ALL/ ;
43    SCALAR VAR  THE VARIANCE ;
44    SET RAPS   RISK AVERSION PARAMETERS /R0*R4/
45    PARAMETER RISKAVER(RAPS) RISK AVERSION COEFICIENT BY RISK AVERSION PARAMETER
46                /R0   0.00000,  R1   0.00075,
47                 R2   0.01500,  R3   0.30000,  R4   1/
48   PARAMETER OUTPUT(*,RAPS) RESULTS FROM MODEL RUNS WITH VARYING RAP
49   OPTION SOLPRINT = OFF;
50   LOOP (RAPS,RAP=RISKAVER(RAPS);
51              SOLVE EVPORTFOL USING NLP MAXIMIZING OBJ ;
52              VAR = SUM(STOCK, SUM(STOCKS,
53                    INVEST.L(STOCK)* COVAR(STOCK,STOCKS) * INVEST.L(STOCKS))) ;
54              OUTPUT("RAP",RAPS)=RAP;
55              OUTPUT(STOCKS,RAPS)=INVEST.L(STOCKS);
56              OUTPUT("OBJ",RAPS)=OBJ.L;
57              OUTPUT("MEAN",RAPS)=SUM(STOCKS, MEAN(STOCKS) *INVEST.L(STOCKS));
58              OUTPUT("VAR",RAPS) = VAR;
59              OUTPUT("STD",RAPS)=SQRT(VAR);
60              OUTPUT("SHADPRICE",RAPS)=INVESTAV.M;
61              OUTPUT("IDLE",RAPS)=FUNDS-INVESTAV.L
62                );
63   DISPLAY OUTPUT;
```

Table 15.5 Looping Sensitivity Example Output

	R0	R1	R2	R3	R4
BUYSTOCK1			1.273	3.905	3.835
BUYSTOCK2		5.324	12.420	5.354	4.958
BUYSTOCK3	17.857	12.152	3.550	1.064	0.968
BUYSTOCK4				8.602	9.223
RAP		7.500000E-4	0.015	0.300	1.000
OBJ	148.214	135.688	123.380	102.254	66.674
MEAN	148.214	141.331	129.839	117.774	117.230
VAR	19709.821	7523.441	430.560	51.734	50.556
STD	140.392	86.738	20.750	7.193	7.110
SHADPRICE	0.296	0.260	0.234	0.173	0.032

Chapter 16 Conducting a Comparative Model Analysis

Models are almost always constructed to be used in a comparative statics analysis. Such an analysis involves multiple model solutions each of which reflects alternative assumptions in terms of data, imposed constraints, fixed variables etc. (each alternative model setup will be called a scenario from here on). Such studies examine how the modeled entity reacts to the scenarios and usually involves a comparison of results across scenarios along with potential examination of results within a scenario. GAMS permits such repeated analyses. Here we cover procedures for: a) altering data; b) activating and deactivating model structural components; c) setting up comparative report writing ; and d) repeatedly solving the model.

16.1 Basic Structure of a Comparative Analysis

The basic structure of a comparative analysis is outlined in Figure 16-1. The first three boxes reflect preparatory steps which would be done in a conventional GAMS program where one sets up the initial data and model then solves. In the box labeled step 1, the comparative model analysis begins. There the scenarios are identified and the scenario data defined. After that we save the data that are to be changed during the scenario runs preserving “base” scenario values. We then enter a loop that is repeated for each scenario to be analyzed. In that loop the first task is to restore the data to its base scenario levels(step 3). This is done so that we always start from the same data. For example if we are altering prices in some scenarios and costs in others once we changed the prices they will be altered forever unless they are changed back (restored) to their original values. Then the data and model differences for that scenario are imposed in step 4 and the model solved. Step 6 involves a report on the individual scenario with items displayed as desired. Then in step 7 data for a cross scenario comparative report is saved. In step 8 we check

to see if more scenarios are to be solved and if so return to repeat steps 3-8 until all scenarios are completed. Finally, we display a comparative report which presents the information saved across scenarios.

Example of such a procedure is given in the context of the basic resource allocation problem (see Chapter 5 of McCarl and Spreen for discussion of the problem). Here we will solve a base version of the model then alter it so the labor and lathe constraints are suppressed and finally solve a version with 25% higher prices for fancy chairs.

Table 16.1 contains the GAMS file that does this. Lines 1-47 define the base model constituting the preparatory steps A-C in figure 16.1. Several model structure features are present to allow the analysis. First, in line 43, the existence of available resource constraint has a $ sign condition attached. This causes the constraints to be active only when there is non zero resource availability and provides us with the mechanism to suppress the labor and lath constraints. Also in lines 72-77 we have defined and filled arrays wherein the original resource availability and prices will be saved.

The scenario steps begin in Line 70 where we define a set which contains the scenario names. In this case we will run: a) a base case which leaves all the data alone; b) a case where the labor constraint is suppressed which is implemented by setting the labor endowment to zero; c) a case where the large lathe constraint is suppressed which is implemented by setting the large lathe endowment to zero; and d) a high price fancy chair scenario. Data further defining the scenarios are entered in lines 79-85.

Lines 87-105 execute our comparative model analysis loop. Lines 88-89 reestablish all data at their original levels. Lines 90-92 put in the data appropriate to the scenario. Line 94 then

executes a solution while lines 96-98 do individual scenario report writing and lines 101-104 store data into arrays for the comparative report. Finally the comparative report is displayed in line 108 after all the solutions are completed. The resultant output is given in Table 16.2. Note the results show varying profits, production patterns, resource usages, and resource values across the scenarios.

16.2 Revising Data

One important concern when doing comparative statics analysis over a number of scenarios involves proper management of data revisions. Modelers must be aware then when a number is changed in GAMS is changed permanently. If one has the following GAMS code

```
1       SCALAR      MYDATA              /1/
2                   MOREDATA  /2/;
3       DISPLAY MYDATA,MOREDATA;
4       MYDATA=3;
5       MOREDATA=MOREDATA*10;
6       DISPLAY MYDATA,MOREDATA;
```

then the initial value of MYDATA and MOREDATA are 1 and 2 respectively. But after lines 5 and 6 these values have changed to 3 and 20 and will remain so for the rest of the GAMS execution, the 1 and 2 are gone. Thus, if one places a new value in during an analysis for an item, then that item is permanently changed unless one resets that value. This may not be desirable in a comparative statics analysis as one may wish to fun first with more resources but original prices then with the original level of resources but with altered prices. The resources will only be reset to

their original values if the modeler explicitly enters instructions to do so.

This point is illustrated in table 16.1 example where parameters are defined into which the original data is to be saved in lines 72-73 while data are stored therein in lines 76-77 and before any scenario alterations are put in place the data are reset to their original vales in lines 88-89. If this were not done the data changes would accumulate during the scenarios.

For example the commands

```
SCALAR LAND /100/
PARAMETER SAVELAND;
SAVELAND = LAND;
SET LANDCHANGE    SCENARIOS FOR CHANGES IN LAND/R1,R2,R3/
PARAMETER VALUE(LANDCHANGE) PERCENT CHANGE IN LAND
            /R1 +10 , R2 + 20 , R3 +30/
LOOP ( LANDCHANGE,
   LAND = LAND * (1 + VALUE ( LANDCHANGE ) / 100. ) );
```

results in land equaling 110, 132 and 171.6 during the loop. However, alteration of the calculation statement so it operated from a saved parameter value

```
LAND = SAVELAND * (1 + VALUE ( LANDCHANGE ) / 100. )
```

results in values of 110 , 120, and 130.

One other important item involves computations. GAMS automatically recomputes all terms specified in the model specification equations (the.. expressions). However, the other computations encountered when setting up the model are not repeated. This leads to two cases that one need be concerned about. Suppose a model was setup as follows:

```
Price(Crop) = 2.00;
Yield(Crop) = 100;
```

```
Cost(Crop) = 50;
Revenue (Crop) = Price(Crop)*Yield(Crop)-Cost(Crop);
Equations
obj             objective function
Land            Land available;
Positive Variables      Acres(Crop)    Cropped Acres
Variables               Objf     Objective function;
        obj..           objf=E=Sum(Crop,Revenue((Crop)*Acres(Crop));
        Land..          Sum(Crop, Acres(Crop))=L=100;
Model           FARM/ALL/
SOLVE FARM USING LP Maximizing Profit;
Price (“corn”)=2.50;
Solve FARM USING LP Maximizing Profit;
Revenue (Crop)=Price (CROP)*Yield(Crop)-Cost(Crop);
Solve FARM Using LP Maximizing Profit
```

In this case the first two solves would be identical since the parameter revenue was not recomputed to reflect the change in the PRICE parameter for corn. The third solution could differ since revenue is recomputed.

One could also fix this by specifying the objective function as

```
Obj..   ObjF=E=SUM(Crop,(Price(CROP)*Yield(CROP)-Cost (CROP))Acres(CROP);
```

Thus in general in a model one either has to include all calculations directly in the model specification equations (those with the .. syntax), or one has to explicitly reissue statements needed to recompute any items affected by data alterations.

Similarly, variable upper and lower bounds as well as variable scaling factors are not automatically recomputed unless one reissues the statement defining .LO , .UP , .FX and .SCALE cases. Thus, when any of these items depend on revised data, then the bound definition calculations need to be repeated.

16.3 Changing Model Structure

Many comparative studies involve model structure modification. One of the big advantages of using a modeling system is the ability to add/delete constraints, variables, or equation terms and reanalyze the problem. This can be achieved with $ controls as done in lines 43-45 of Table 16.1. Suppose we consider an alternative example. Suppose the following lines are put in a GAMS problem:

```
SCALAR   ISITACTIVE     tells whether items are active /0/;
CONDEQ$ISITACTIVE..     sum(stuff,x(stuff)) =L= 1;
EQNOTH(index)..         sum(stuff,r(index,stuff)*x(stuff)) +
                        4*sum(stuff,Y(stuff))$ISITACTIVE =L= 50;
```

This addition would cause the CONDEQ equation and the Y term in the EQNOTH equations to only appear in the empirical model when the ISITACTIVE parameter was nonzero. Thus, the sequence

```
ISITACTIVE = 0;
SOLVE MODELNAME USING LP MAXIMIZING OBJ;
ISITACTIVE =1;
SOLVE MODELNAME USING LP MAXIMIZING OBJ;
```

would cause the model to be solved with and without the constraint and term.

16.4 Solving Repeatedly

More than one model can be solved in a run. Thus, one can stack solve statements as in the example immediately or loop over solves as in Table 16.1. In such case the optimal basis from the model solved immediately before will provide the a starting basis for the solve at hand. This means ordering of the scenarios for similarity of solutions.

16.5 Comparative Report Writing

The development of a comparative report writer is usually attractive when doing multiple runs. Report writing commands which use information from the optimal solution (M and .L

references) use information from the most recent solution so one must save the data if comparing reports are desired. A report writer which does that is illustrated in Table 16.1. In that case a parameter (COMPAR) is defined over the scenario set (RUNS)-see line 74. In turn, during loop execution the COMPAR array is saved with scenario dependent values of variables and shadow prices. Finally, when the output is displayed a comparison across scenarios appears (Table 16.2).

Table 16.1. Example of Comparative Run

```
 2   SET    PROCESS       TYPES OF PRODUCTION PROCESSES
 3                            /FUNCTNORM , FUNCTMXSML , FUNCTMXLRG
 4                            ,FANCYNORM , FANCYMXSML , FANCYMXLRG/
 5          RESOURCE      TYPES OF RESOURCES
 6                            /SMLLATHE,LRGLATHE,CARVER,LABOR/ ;
 7
 8   PARAMETER PRICE(PROCESS)     PRODUCT PRICES BY PROCESS
 9                     /FUNCTNORM  82, FUNCTMXSML  82, FUNCTMXLRG  82
10                     ,FANCYNORM 105, FANCYMXSML 105, FANCYMXLRG 105/
11              PRODCOST(PROCESS)     COST BY PROCESS
12                     /FUNCTNORM  15, FUNCTMXSML  16  , FUNCTMXLRG  15.7
13                     ,FANCYNORM  25, FANCYMXSML  26.5, FANCYMXLRG  26.6/
14              RESORAVAIL(RESOURCE) RESOURCE AVAILABLITY
15                     /SMLLATHE 140, LRGLATHE  90,
16                      CARVER   120, LABOR    125/
17
18   TABLE RESOURUSE(RESOURCE,PROCESS) RESOURCE USAGE
19
20                  FUNCTNORM   FUNCTMXSML  FUNCTMXLRG
21   SMLLATHE          0.80       1.30        0.20
22   LRGLATHE          0.50       0.20        1.30
23   CARVER            0.40       0.40        0.40
24   LABOR             1.00       1.05        1.10
25    +              FANCYNORM   FANCYMXSML  FANCYMXLRG
26   SMLLATHE          1.20       1.70        0.50
27   LRGLATHE          0.70       0.30        1.50
28   CARVER            1.00       1.00        1.00
29   LABOR             0.80       0.82        0.84;
30
31    POSITIVE VARIABLES
32              PRODUCTION(PROCESS) ITEMS PRODUCED BY PROCESS;
33    VARIABLES
34              PROFIT              TOTALPROFIT;
35    EQUATIONS
36              OBJT                  OBJECTIVE FUNCTION ( PROFIT )
37              AVAILABLE(RESOURCE)   RESOURCES AVAILABLE ;
38
39     OBJT..  PROFIT =E=
40                SUM(PROCESS,(PRICE(PROCESS)-PRODCOST(PROCESS))
41                              * PRODUCTION(PROCESS)) ;
42
43     AVAILABLE(RESOURCE)$RESORAVAIL(RESOURCE)..
44         SUM(PROCESS,RESOURUSE(RESOURCE,PROCESS)*PRODUCTION(PROCESS))
45                           =L= RESORAVAIL(RESOURCE);
46
47    MODEL RESALLOC /ALL/;
48    option solprint=off;
49    option limrow=0;
50    option limcol=0;
51
52    SOLVE RESALLOC USING LP MAXIMIZING PROFIT;
55    set type   types of chairs               /functional,fancy/
56        item   items for reports             /level,production,usage,value/
57        map(type,process)    map of chair types to processes
58                      /functional.(FUNCTNORM , FUNCTMXSML , FUNCTMXLRG)
```

```
 59                    fancy     .(FANCYNORM , FANCYMXSML , FANCYMXLRG)/;
 60
 61   parameter resourstat(resource,item) resource status
 62             chairs(type)              production of chairs;
 63
 64             resourstat(resource,"usage")=available.l(resource);
 65             resourstat(resource,"value")=available.m(resource);
 66             chairs(type)=sum(map(type,process),production.l(process));
 67
 68   display chairs,resourstat;
 69
 70   set runs  /base,nolabor, noLRGLATHE ,hifancy/
 71
 72   parameter savRESORAv(RESOURCE) saved resource availability
 73             savprice(process)    saved prices
 74             compar(item,*,runs)       comparative report;
 75
 76             savRESORAv(RESOURCE)=RESORAVAIL(RESOURCE);
 77             savprice(process)=price(process);
 78
 79   table adjust(*,runs) alternative run configuration
 80               base nolabor noLRGLATHE  hifancy
 81  smllathe     0
 82  lrglathe     0               1
 83  labor        0     1
 84  fancy        0                         0.25
 85  functional   0                             ;
 86
 87   loop(runs,
 88             RESORAVAIL(RESOURCE)=savRESORAv(RESOURCE);
 89             price(process)=savprice(process);
 90             RESORAVAIL(RESOURCE)$adjust(resource,runs)=0;
 91             price(process)$sum(map(type,process),adjust(type,runs))=
 92             price(process)*(1+sum(map(type,process),adjust(type,runs)));
 93
 94             SOLVE RESALLOC USING LP MAXIMIZING PROFIT;
 95
 96             resourstat(resource,"usage")=available.l(resource);
 97             resourstat(resource,"value")=available.m(resource);
 98             chairs(type)=sum(map(type,process),production.l(process));
100             display chairs,resourstat;
101             compar("level","profit",runs)=profit.l;
102             compar("usage",resource,runs)=resourstat(resource,"usage");
103             compar("value",resource,runs)=resourstat(resource,"value");
104             compar("production",type,runs)=chairs(type);
105        );
107             option decimals=2;
108             display compar;
```

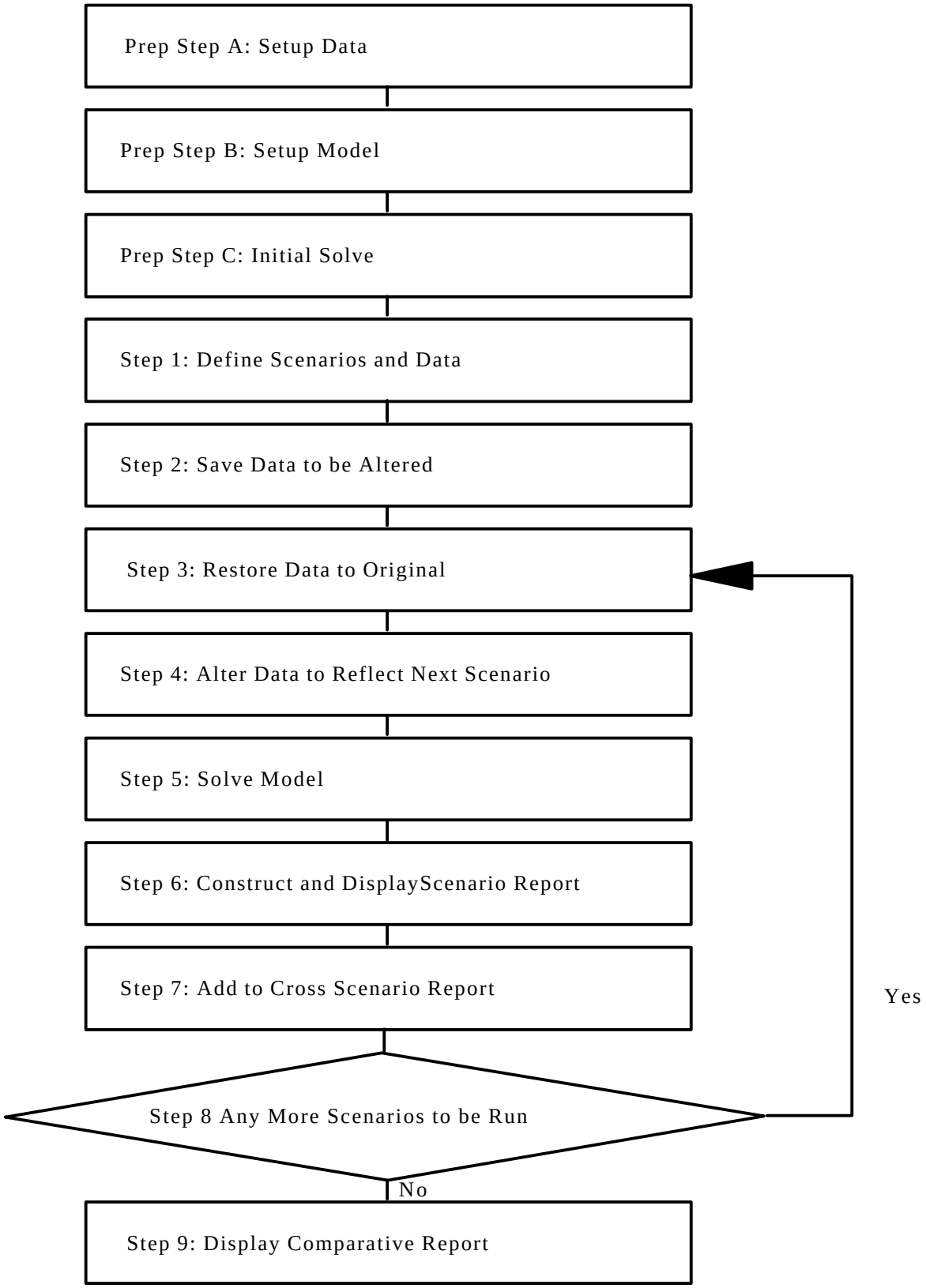
Table 16.2. Comparative Report Writing Output

```
----    108 PARAMETER COMPAR        comparative report

                                BASE      NOLABOR  NOLRGLATHE     HIFANCY

LEVEL      .PROFIT         10417.29     11830.43    11002.82    12798.83
PRODUCTION.FUNCTIONAL         62.23       176.60       41.20        2.44
PRODUCTION.FANCY              78.20                   103.52      119.02
USAGE      .SMLLATHE         140.00       140.00      140.00      140.00
USAGE      .LRGLATHE          90.00        90.00       90.00       90.00
USAGE      .CARVER           103.09        70.64      120.00      120.00
USAGE      .LABOR            125.00       125.00      125.00       97.93
VALUE      .SMLLATHE          33.33        57.39        5.09       48.66
VALUE      .LRGLATHE          25.79        42.17       42.17       40.58
VALUE      .CARVER                                     34.63       19.45
VALUE      .LABOR             27.44        27.44       49.08
```

Figure 16.1 Outline of a Comparative Model Analysis



Chapter 17 Interfacing with other Programs

Users may find a need to interface GAMS with other programs. Such interfaces may span the continuum from being interactive to a one way data pass where GAMS results are sent to a spreadsheet package for summarization and graphing. One may wish this link to either be carried out by a manual or an automated process. In this chapter, we cover a number of aspects of communicating with other programs. We should also reference readers to the continuing development efforts by Tom Rutherford and Michael Ferris who distribute interfaces with other programs and with web based processes. Rutherford's material is distributed through http://robles.colorado.edu/~tomruth , Ferris' can be inquired about through email contact to ferris@cs.wisc.edu)

The presentation below is organized into three main sections. First, we deal with input topics, then output issues followed by discussion of interactive applications.

17.1 Input

One may be interested in providing input to a GAMS program from another program. This is done by providing GAMS executable instructions from the companion program. Such instructions can be provided via three mechanisms. First, and most generally one can have the companion program write a text file of GAMS executable code then including that code. The second involves special purpose interfaces from particular types of programs such as spreadsheets. The third involves the use of Rutherford's computer generated interface. Each will be discussed below.

17.1.1 General Purpose Approaches

The general mechanism via which GAMS can use data from another program is through the use of the INCLUDE or the related BATINCLUDE , LIBINCLUDE, and SYSINCLUDE

syntaxes. The INCLUDE syntax simply includes a named file as part of the GAMS program, while the other forms include code segments with substitutable parameters. The LIBINCLUDE syntax causes a program to be included from by default the subdirectory INCLIB under the GAMS system directory or more generally whatever directory is specified by the libincdir parameter in the GAMS call or the gamsparm.txt file. The SYSINCLUDE syntax operates similarly to the LIBINCLUDE except that the files which are utilized are drawn from the GAMS system directory or wherever specified in the SYSINCDIR parameter of the GAMS start-up call or GAMSPARM.TXT file. Each of these statements is explained in the GAMS systems manual, so here we will confine our treatment to examples illustrating the general use of the procedures.

17.1.1.1 Including files – simple variants

Suppose one is running a GAMS program which creates a solution that is used in another program to revise some input data. For example suppose an agricultural study is being done which involves a GAMS solution of a model for each relevant period which gives crop acreage along with a selection of rotation, tillage method and fertilization level. Also suppose a companion program is being run which computes fertilizer carryover, yields under different management regimes and erosion rates for the next period. This yield data would then be an input to the GAMS model by including a file of GAMS instructions which contains those numbers. This would entail the construction of an interfacing program which translated the crop simulation yield output into GAMS instructions. Suppose this has been done and we have the file in a format like that below

```
* the first 5 lines and the last would be user defined and
*     would not come from companion program
set crops      Crop names          /corn, wheat/
set tillage    tillage methods     /conventional, notill/
set fert       Fertilization level /fert1*fert3/
parameter
  yields(crops,tillage,fert) crop yields from simulator   /
```

```
* the following lines would be from the other program
      corn.conventional.fert1      92.
      corn.conventional.fert2      120.
      corn.conventional.fert3      140.
      corn.notill.fert1   87
      corn.notill.fert2   117
      corn.notill.fert3   136
      wheat.conventional.fert1     42
      wheat.conventional.fert2     48
      wheat.conventional.fert3     54
      wheat.notill.fert1   42
      wheat.notill.fert2   48
      wheat.notill.fert3   54
* the line below would not be from other program
    /;
```

When implementing this one would most likely use the include file sequence as follows.

First,define the file to be included with the name fromprog.gms with the following content

```
      corn.conventional.fert1      92.
      corn.conventional.fert2      120.
      corn.conventional.fert3      140.
      corn.notill.fert1   87
      corn.notill.fert2   117
      corn.notill.fert3   136
      wheat.conventional.fert1     42
      wheat.conventional.fert2     48
      wheat.conventional.fert3     54
      wheat.notill.fert1   42
      wheat.notill.fert2   48
      wheat.notill.fert3   54
```

The redefine the GAMS program (includ.gms) above to be

```
set crops      Crop names          /corn, wheat/
set tillage    tillage methods     /conventional, notill/
set fert       Fertilization level /fert1*fert3/
parameter
  yields(crops,tillage,fert) crop yields from simulator   /
$include fromprog
 /;
```

In turn whenever this GAMS program was run the yields in the file fromprog.gms would be incorporated. There are several items relevant to the above procedure which are meritorious of mention.

1) When communicating between a program and GAMS one must use some pre-agreed upon set element names and definitions for the items under those names. In the above example this would entail a precise definition of the fertilizer, crop and tillage item names as well as what is meant by the alternative fertilizer levels.

2) The data must be included in an acceptable GAMS format. Tables can be used providing alignment under column labels is assured. We generally use the parameter format as given above as this frequently simplifies programming in the interface module since alignment is not a concern rather for each item one has to enter the set element names separated by periods followed by the numerical value.

3) The file included is brought in at compile time for the GAMS program. Thus, within any single GAMS program interaction is not possible as all of the files are included at compile time. The only way around this is through the use of save and restart files as will be discussed below.

4) Multiple includes of the same data may require a more complex structure as discussed in the immediately following paragraphs.

17.1.1.2 Including files with substitutable parameters

One very useful feature within GAMS it the ability to use substitutable parameters in include files. A user can include a file to alter the particular items being included. Let's suppose for illustrative purposes that we have two data sets that we wish to merge into an overall composite set. In particular suppose in file1 we have data for yields and costs of crops in time period one, while in file2 we have data again on costs of yields of the crops for time period two. We could use the following BATINCLUDE files to merge these data

```
Set      periods          /period1,period2/
 set     crops   /corn,wheat/
set      items   /yield,cost/
parameter       Alldata(period,crops,items) merged data;
$batinclude mergedat 1
$batinclude mergedat 2
```

where the bat include file looked like the following.

```
table dataset%1(crops,items) data set %1
$include file%1
;
alldata("period%1",crops,items)=dataset%1(crops,items);
```

and file1 appears as follows

```
          yield   cost
corn      120     56
wheat     40      35
```

while file2 looks like

```
          yield   cost
corn      140     86
wheat     60      45
```

The net effect of the BATINCLUDE syntax is that the included file mergedat is called up with the number following it name in the BATINCLUDE statement substituted where ever the %1 appears.

The first batinclude causes GAMS to see the file

```
table dataset1(crops,items) data set 1
$include file1
;
alldata("period1",crops,items)=dataset1(crops,items);
```

while the second causes execution of

```
table dataset2(crops,items) data set 2
$include file2
;
alldata("period2",crops,items)=dataset2(crops,items);
```

Note in both of these cases the file appears with the calling parameter from the BATINCLUDE statement (which appears after the name of the file to be incorporated) substituted in place of the %1.

The total effect of the above statements is that a GAMS program of the form in the following LST file segment is executed.

```
     1  Set     periods             /period1,period2/
     2  set     crops               /corn,wheat/
     3  set     items               /yield,cost/
     4  parameter   Alldata(periods,crops,items) merged data;
BATINCLUDE F:\MCCARL\574PROJ\EXAMPLES\CH17\MERGEDAT.GMS
     6  table dataset1(crops,items) data set 1
INCLUDE    F:\MCCARL\574PROJ\EXAMPLES\CH17\FILE1.GMS
     8                  yield   cost
     9          corn    120     56
    10          wheat   40      35
    11  ;
    12  alldata("period1",crops,items)=dataset1(crops,items);
BATINCLUDE F:\MCCARL\574PROJ\EXAMPLES\CH17\MERGEDAT.GMS
    14  table dataset2(crops,items) data set 2
INCLUDE    F:\MCCARL\574PROJ\EXAMPLES\CH17\FILE2.GMS
    16                  yield   cost
    17          corn    140     86
    18          wheat   60      45
    19  ;
    20  alldata("period2",crops,items)=dataset2(crops,items);
```

Notice that in this LST file GAMS has marked exactly in which files it is incorporating at each stage.

The BATINCLUDE command can be a valuable part of a GAMS modelers repertoire especially regarding output as will be discussed later. The LIBINCLUDE and SYSINCLUDE syntaxes operate in an essential identical manner with the only difference being the locations from which the files are sought. Under the BATINCLUDE syntax, files are assumed to be local in the current working directory, whereas under the other two syntax forms special file locations are utilized.

One additional note involves operation whenever more than one substitutable parameter is present. Namely, when more than one parameter is used one enters the string %1 wherever the first substitutable parameter is to be placed, %2 wherever the second one appears, %3 wherever the third and so on. In turn GAMS replaces these items and executes the commands just just as if original commands had been typed with the substituted characters within them.

The items substituted can be numbers, text characters or quoted strings including spaces or special characters. For example (batinc2.gms) using a command of the form

$batinclude fileit 10 one "this is it" "a+b*c"

would result in a 10 being placed in where ever %1 is used the string "one" wherever %2 appears and a+b*c used where ever a %4 appeared.

If we were to use such syntax in a more general setting like the following:

```
scalar a  /10/
       b  /5/
       c  /4/
       one /0/
       d   /2/  ;
$batinclude fileit  10 one "this is it" "a+b*c"
$batinclude fileit  20 d "this is next" "a+b+c"
```

In conjunction with in a fileit.gms file of the form.

```
%2 = %1 * (%4);
display '%3',%2 ;
```

Then the resultant file that GAMS executes is that in the LST file below.

```
    1  scalar a  /10/
    2         b  /5/
    3         c  /4/
    4         one /0/
    5         d   /2/  ;
BATINCLUDE F:\MCCARL\574PROJ\EXAMPLES\CH17\FILEIT.GMS
    7  one = 10 * (a+b*c);
    8  display 'this is it',one ;
```

```
BATINCLUDE F:\MCCARL\574PROJ\EXAMPLES\CH17\FILEIT.GMS
   10  d = 20 * (a+b+c);
   11  display 'this is next',d ;
```

This shows we can substitute formulas, numbers, output labels and many other things in context of GAMS execution when using BATINCLUDE or one of the other include formats which allow substitutable parameters.

17.1.1.3 Including files from other GAMS programs

One trick which is commonly used in GAMS programming is to incorporate results from one GAMS program into another. This strategy is frequently used to incorporate a basis (see the discussion on Rutherford's webSite or the chapter 11 discussion of GAMSBAS). One also may also wish to include numerical results from one GAMS program into another. We have used this strategy with large models when we

1) wished to include results from one run into another
2) wanted to pass solution information on to other members of modeling exercise located at a distant location.
3) had a costly model to run and wished to preserve solutions for possible future analyses.

We will not cover the mechanism for bringing such instructions into the program as this is done via the $INCLUDE syntax which has already been discussed. For an example refer to Chapter 11 where the basis file that is written is being included back into the GAMS program. We also defer the discussion of how we write the data into a GAMS readable file until section 17..

One other strategy and that is possible is that one can properly format the display in a GAMS program and then cut and paste the results into a file which is subsequently included in other GAMS programs. This is most easily done using a display formatting option statement such

as

option dataitem:0:0:5;display dataitem;

which in the chapter 14 example results in a numeric display of the form

```
----     10 PARAMETER DATA – option allows 0 row by 5 columns (:0:0:5)
index11.index21.index31.index41 2,    index11.index21.index31.index42 2
index11.index21.index32.index41 2,    index11.index21.index32.index42 2
index11.index22.index31.index41 2,    index11.index22.index31.index42 2
index11.index22.index32.index41 2,    index11.index22.index32.index42 2
index12.index21.index31.index41 2,    index12.index21.index31.index42 2
index12.index21.index32.index41 2,    index12.index21.index32.index42 2
index12.index22.index31.index41 2,    index12.index22.index31.index42 2
index12.index22.index32.index41 2,    index12.index22.index32.index42 2
```

More generally, this display is of the form.

option itemname: n1:0:n2; display itemname;

where itemname is the thing to to be displayed ;
n1 is the number of decimal points one wishes in this display; and
n2 is the number of dimensions or set indices used in the parameter.

Thus if we had parameter DATA(A, B,C) and wanted four decimal places, we would use the syntax

option data:4:0:3;display data;

When this syntax is run the GAMS LST file would look much like that above and one could copy the results and paste them in a new file called toimport. Then in a target file where the results are to be included one would employ syntax like the following.

```
set a /1,2,3/
set b / T,E,f/
set c /are, you, there/
parameter importdat(a, b,c) imported data /
$include toimport
/;
```

This would result in the data included in the target file.

We generally employ this option strategy (one where we allow zero row items and a number of the column items equal to the number of sets defining the data item) for preparing data

to be exported. This allows us to generate data in a format readily incorporated a parameter definition. Consequently, we do not have to worry about the issues of column alignment that would need to be faced when using a table syntax. Nor do we need to worry about data continuation when the table width does not readily fit across the page. More discussion of the option command for output formatting appears in chapter 14 .

17.1.2 Special Sources

Data may also be included from a number of a special purpose programs. In particular herein we discuss incorporation of data from spreadsheets, the matlab software, zip files, and files from the ???? word processor. Most of these procedures were developed by Rutherford while the matlab interface was developed by Michael Ferris.

17.1.2.1 Incorporation of data from spreadsheets

Data can be imported from spreadsheets using a program called SSLINK developed by Rutherford. The witeup for this program is on the WebSite. To include sees the general format is one includes a statement in a GAMS program of the form a

worksheets

17.1.2.2 Incorporation of data from MATLAB

matlab

17.1.2.3 Incorporation of data from ZIP and PRM files

zip

prm

17.1.3 Web based cgi

17.1.4 Toward more specific interfaces

visual basic

17.2 output

17.2.1 through put

general formatting

special , delimited formatting

to GAMS

spreadsheet imports

databases

17.2.2 special purpose links

17.2.2.1 Spreadsheets

Using Put

Other Means SSLINK

17.2.2.2 Graphics Programs

gnuplot

matlab

17.2.2.3 Other special interfaces

17.2.3 Interfacing other ways

pgm

zip files

17.3 Web based interface

17.4 Interactive interfaces

17.4.1 Interactive compiled program

17.4.2 save restart methods

```
*#########################
*Five examples in GNUPLOT
*#########################
$title  Example 1: Directing GNUPLOT Output to Screen or File

set year /1971*1992/, crop /corn,wheat,sorghum,soybeans/;

Table Acreage(year,crop)  Acreage by crop and year
          corn    soybeans       wheat     sorghum

1971       64044       42701       47613       16296
1972       57412       46871       47241       13354
1973       61884       55794       53837       15855
1974       65345       52364       65585       13947
1975       67438       53572       69104       15342
1976       71238       49373       70755       14713
1977       70860       57600       66246       14096
1978       70264       63334       57046       13566
1979       72381       70560       61920       12895
1980       73020       67847       70982       12493
1981       74497       66174       80809       13706
1982       72693       69445       77929       14303
1983       51457       62532       61379        9988
1984       71841       66103       66917       15344
1985       75224       61603       64934       16823
1986       69159       58310       60723       13907
1987       59505       57190       55975       10568
1988       58250       57390       53189        9067
1989       64703       59555       62189       11143
1990       66952       56521       69243        9131
1991       68852       58031       57703        9922
1992       72145       58404       62387       12199
;

*1) show on screen

$setglobal gp_xl year
$libinclude c:\gams25\gnuplot Acreage


*2) save it on ex2.gif with the title and x and y axis label

$setglobal gp_title "Graphic of U.S. Acreage in 1000 acres"
$setglobal gp_key 'bottom right'
```

```
$setglobal gp_xl year
$setglobal gp_xlabel "Year"
$setglobal gp_ylabel "Thousand Acreages"
$setglobal gp_output 'ex2.gif'
$setglobal gp_term 'gif'
$libinclude c:\gams25\gnuplot Acreage
```

*3) save it on ex21.gif with the title, x and y label and GRID

```
$setglobal gp_title "Graphic of U.S. Acreage in 1000 acres"
$setglobal gp_key 'bottom right'
$setglobal gp_grid   'yes'
$setglobal gp_xl year
$setglobal gp_xlabel "Year"
$setglobal gp_ylabel "Thousand Acreages"
$setglobal gp_output 'ex21.gif'
$setglobal gp_term 'gif'
$libinclude c:\gams25\gnuplot Acreage
```

*4) save it on ex22.gif with the title, x and y label and a Range

```
$setglobal gp_title "Graphic of U.S. Acreage in 1000 acres"
$setglobal gp_key 'bottom right'
$setglobal gp_xrange     '[11:20]'
$setglobal gp_xl year
$setglobal gp_xlabel "Year"
$setglobal gp_ylabel "Thousand Acreages"
$setglobal gp_output 'ex22.gif'
$setglobal gp_term 'gif'
$libinclude c:\gams25\gnuplot Acreage
```

*5) save it on ex23.gif with the linespoints

```
$setglobal gp_title "Graphic of U.S. Acreage in 1000 acres"
$setglobal gp_key 'bottom right'
$setglobal gp_style 'linespoints'
$setglobal gp_xl year
$setglobal gp_xlabel "Year"
$setglobal gp_ylabel "Thousand Acreages"
$setglobal gp_output 'ex23.gif'
$setglobal gp_term 'gif'
$libinclude c:\gams25\gnuplot Acreage
```

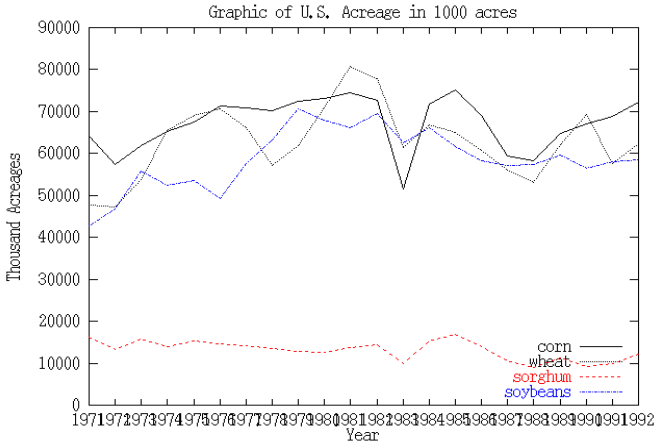


Figure 1: Ex2.gif

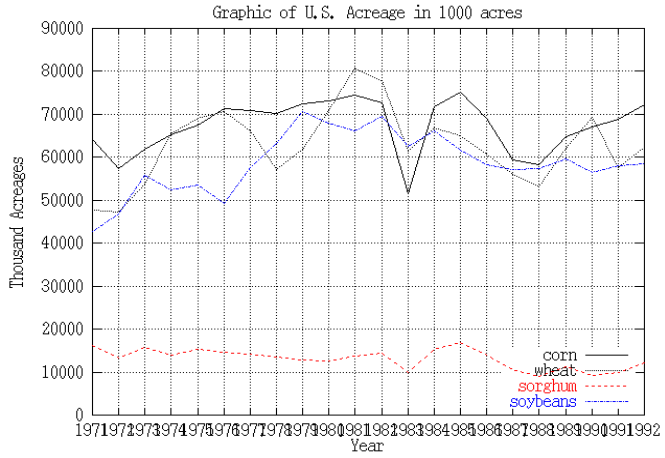


Figure 2: Ex21.gif

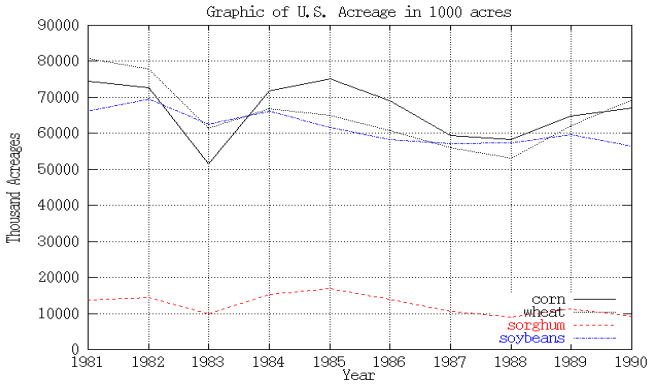


Figure 3:

Ex22.gif

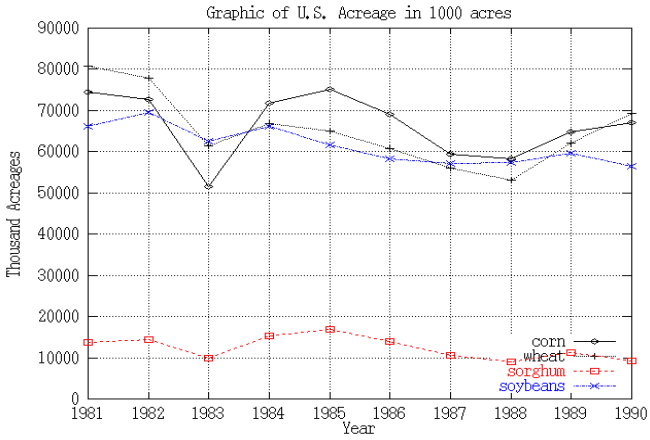


Figure 4: Ex23.gif

*################################
*Examples in SSDUMP

```
*################################
$title example #1:SSDUMP
set year  /1971*1992/
   crop  /corn, soybeans, wheat, sorghum/
   subreg  /txhiplains, txrolingpl, txcntblack, txeast, txedplat,
            txsouth, txtranspec/;

Table TXACRE(year,subreg,crop)  Texas historical acreage by crops

                      corn    soybeans       wheat     sorghum

1971.TXHIPLAINS        244          51        1029        2926
1971.TXROLINGPL                                317         521
1971.TXCNTBLACK        163           9         124         934
1971.TXEAST             34           9                      67
1971.TXEDPLAT            2                       8         106
1971.TXSOUTH            64           2           8         580
1971.TXTRANSPEC                                  7           8
1972.TXHIPLAINS        199          58        1185        2486
1972.TXROLINGPL                                609         518
1972.TXCNTBLACK        142          35         142         988
1972.TXEAST             30          29           1          59
1972.TXEDPLAT            1                      28         100
1972.TXSOUTH            45           3          23         601
1972.TXTRANSPEC                                  8           8
1973.TXHIPLAINS        367         131        2186        3030
1973.TXROLINGPL                                817         645
1973.TXCNTBLACK        133          29         268        1261
1973.TXEAST             27          59           4          43
1973.TXEDPLAT                                   64         137
1973.TXSOUTH            72           7          52         893
1973.TXTRANSPEC                                  5           8
1974.TXHIPLAINS        541          88        1665        2175
1974.TXROLINGPL           1                    1090         400
1974.TXCNTBLACK        125          24         393        1207
1974.TXEAST             34          34           6          74
1974.TXEDPLAT            1                      72         131
1974.TXSOUTH            61           4          66         997
1974.TXTRANSPEC                                  5           9
1975.TXHIPLAINS        771         127        3060        2629
1975.TXROLINGPL                                1840         617
1975.TXCNTBLACK        139          43         519        1498
1975.TXEAST             41          40           3          60
1975.TXEDPLAT            2                     154         138
1975.TXSOUTH            91           5          92        1171
1975.TXTRANSPEC                                 26           2
1976.TXHIPLAINS       1210          75        2125        1890
1976.TXROLINGPL          4                    1665         479
1976.TXCNTBLACK        132          48         690        1270
1976.TXEAST             21          53          17          48
1976.TXEDPLAT            2                      74         149
1976.TXSOUTH            71                      81         757
1976.TXTRANSPEC                                 40
1977.TXHIPLAINS       1211         152        2337        1320
1977.TXROLINGPL          3                    1580         335
1977.TXCNTBLACK        165          74         555        1328
1977.TXEAST             37         103          19          30
1977.TXEDPLAT            5                     124         121
1977.TXSOUTH            119           8          62         666
1977.TXTRANSPEC                                 19           8
1978.TXHIPLAINS        907         188        1285        1370
1978.TXROLINGPL          2                     834         280
1978.TXCNTBLACK        199          51         474        1351
```

1978.TXEAST	33	103	18	31
1978.TXEDPLAT	3		46	88
1978.TXSOUTH	144	6	33	640
1978.TXTRANSPEC			5	8
1979.TXHIPLAINS	700	217	2393	1230
1979.TXROLINGPL	3		1370	290
1979.TXCNTBLACK	209	69	620	1188
1979.TXEAST	37	110	17	24
1979.TXEDPLAT	3		115	121
1979.TXSOUTH	173	16	72	731
1979.TXTRANSPEC			5	8
1980.TXHIPLAINS	735	65	2595	1137
1980.TXROLINGPL	3		1255	168
1980.TXCNTBLACK	226	37	1116	983
1980.TXEAST	28	122	62	27
1980.TXEDPLAT	2		85	79
1980.TXSOUTH	154	5	46	626
1980.TXTRANSPEC			23	
1981.TXHIPLAINS	522	58	2750	1239
1981.TXROLINGPL	1		1864	195
1981.TXCNTBLACK	212	22	1511	894
1981.TXEAST	31	73	142	41
1981.TXEDPLAT	1		145	88
1981.TXSOUTH	187		87	822
1981.TXTRANSPEC			38	9
1982.TXHIPLAINS	523	445	2520	2180
1982.TXROLINGPL	2		1605	357
1982.TXCNTBLACK	237	55	1437	1062
1982.TXEAST	25	58	102	44
1982.TXEDPLAT	2		178	109
1982.TXSOUTH	167	2	108	682
1982.TXTRANSPEC			18	2
1983.TXHIPLAINS	400	108	1985	940
1983.TXROLINGPL	1		1275	237
1983.TXCNTBLACK	266	49	1069	657
1983.TXEAST	23	34	84	30
1983.TXEDPLAT	1		97	89
1983.TXSOUTH	203		51	589
1983.TXTRANSPEC			15	
1984.TXHIPLAINS	532	40	2330	1295
1984.TXROLINGPL	3		1260	174
1984.TXCNTBLACK	415	32	1172	862
1984.TXEAST	35	44	66	26
1984.TXEDPLAT	1		90	72
1984.TXSOUTH	220		45	609
1984.TXTRANSPEC			10	8
1985.TXHIPLAINS	517	75	2620	1567
1985.TXROLINGPL	6		1680	287
1985.TXCNTBLACK	361	17	1175	861
1985.TXEAST	43	20	129	25
1985.TXEDPLAT	28	1	128	87
1985.TXSOUTH	216		93	591
1985.TXTRANSPEC			6	8
1986.TXHIPLAINS	467	56	2340	1468
1986.TXROLINGPL	5		1400	249
1986.TXCNTBLACK	403	26	845	764
1986.TXEAST	56	32	40	51
1986.TXEDPLAT	6		82	79
1986.TXSOUTH	163		67	476
1986.TXTRANSPEC			7	8
1987.TXHIPLAINS	412	40	1730	895
1987.TXROLINGPL	3		1113	191
1987.TXCNTBLACK	383	31	552	607
1987.TXEAST	30	21	43	36
1987.TXEDPLAT	3		87	68
1987.TXSOUTH	170		63	361
1987.TXTRANSPEC			3	8

```
1988.TXHIPLAINS         423          61        1472         668
1988.TXROLINGPL           3                    930         123
1988.TXCNTBLACK         392          25         629         499
1988.TXEAST              36          31          45          21
1988.TXEDPLAT             3                     58          56
1988.TXSOUTH            222                     54         282
1988.TXTRANSPEC                                  3           9
1989.TXHIPLAINS         592         178         748        1208
1989.TXROLINGPL           3                   1225         132
1989.TXCNTBLACK         369          49         774         643
1989.TXEAST              36          42          55          30
1989.TXEDPLAT             3                    121          63
1989.TXSOUTH            182           5          66         308
1989.TXTRANSPEC                                  2           2
1990.TXHIPLAINS         621          28        1736         662
1990.TXROLINGPL           7                   1354         126
1990.TXCNTBLACK         363          44         833         645
1990.TXEAST              34          34          53          23
1990.TXEDPLAT             3                    127          63
1990.TXSOUTH            230           2          83         367
1991.TXHIPLAINS         642          24        1157         738
1991.TXROLINGPL           7                    903         141
1991.TXCNTBLACK         375          38         556         720
1991.TXEAST              35          29          35          26
1991.TXEDPLAT             3                     85          70
1991.TXSOUTH            238           2          56         409
1991.TXTRANSPEC                                                1
1992.TXHIPLAINS         694          56        1571        1146
1992.TXROLINGPL           8                   1225         218
1992.TXCNTBLACK         405          87         754        1117
1992.TXEAST              38          66          48          40
1992.TXEDPLAT             3           2         115         109
1992.TXSOUTH            257           4          76         635
1992.TXTRANSPEC                                                2
;
```

parameter subacre(year,subreg) subreg acres ;
subacre(year,subreg)
  =sum(crop, txacre(year,subreg,crop));

parameter totacre(year)   total texas acres ;
totacre(year)
  =sum(subreg, subacre(year,subreg));

*save the total Texas acres in one-dimensional array
$libinclude c:\gams25\ssdump totacre acre1.wk1

*save the Texas subregion acres in two-dimensional array
$libinclude c:\gams25\ssdump subacre acre2.wk1

*save the Texas acres by crop in three-dimensional array
$libinclude c:\gams25\ssdump txacre acre3.wk1

* The Spreadsheet of Acre1.wk1

* The Spreadsheet of Acre2.wk1

* The Spreadsheet of Acre3.wk1

```
*##########################
*Examples in SSIMPORT
*##########################
title example #2:SSIMPORT
set year  /1971*1992/
    crop  /corn, soybeans, wheat, sorghum/
    subreg  /txhiplains, txrolingpl, txcntblack, txeast, txedplat,
             txsouth, txtranspec/;

parameter TXACRE(year,subreg,crop)  Texas historical acreage by crops;

$libinclude c:\gams25\ssimport txacre im3.wk1 a1..f154
option txacre:0:2:1;display txacre;

parameter subacre(year,subreg)   Texas subreg acres;

$libinclude c:\gams25\ssimport subacre im2.wk1 a1..h23
option subacre:0:1:1;display subacre;

parameter totacre(year)   total texas acres;
$libinclude c:\gams25\ssimport totacre im1.wk1 a1..b22
option totacre:0:0:1;display totacre;
```

The Example2 SSIMPORT list file

```
LIBINCLUDE C:\GAMS25\SSIMPORT.GMS
  67  parameter txacre /
INCLUDE    C:\GAMS25\225A\GAMSLINK.SCR
  69  *  Tuesday, June 30 1998   15:28:18
  70  *  C:\GAMS25\IM3.WK1  TXACRE
  71  1971.TXCNTBLACK.CORN     1.63000000000000E+0002
  72  1971.TXCNTBLACK.SORGHUM  9.34000000000000E+0002
  73  1971.TXCNTBLACK.SOYBEANS 9.00000000000000E+0000
  74  1971.TXCNTBLACK.WHEAT    1.24000000000000E+0002
  75  1971.TXEAST.CORN         3.40000000000000E+0001
  76  1971.TXEAST.SORGHUM      6.70000000000000E+0001
  77  1971.TXEAST.SOYBEANS     9.00000000000000E+0000
  78  1971.TXEDPLAT.CORN       2.00000000000000E+0000
  79  1971.TXEDPLAT.SORGHUM    1.06000000000000E+0002
  80  1971.TXEDPLAT.WHEAT      8.00000000000000E+0000
  81  1971.TXHIPLAINS.CORN     2.44000000000000E+0002
  82  1971.TXHIPLAINS.SORGHUM  2.92600000000000E+0003
  83  1971.TXHIPLAINS.SOYBEANS 5.10000000000000E+0001
  84  1971.TXHIPLAINS.WHEAT    1.02900000000000E+0003
  85  1971.TXROLINGPL.SORGHUM  5.21000000000000E+0002
  86  1971.TXROLINGPL.WHEAT    3.17000000000000E+0002
  87  1971.TXSOUTH.CORN        6.40000000000000E+0001
  88  1971.TXSOUTH.SORGHUM     5.80000000000000E+0002
  89  1971.TXSOUTH.SOYBEANS    2.00000000000000E+0000
  90  1971.TXSOUTH.WHEAT       8.00000000000000E+0000
  91  1971.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
  92  1971.TXTRANSPEC.WHEAT    7.00000000000000E+0000
  93  1972.TXCNTBLACK.CORN     1.42000000000000E+0002
  94  1972.TXCNTBLACK.SORGHUM  9.88000000000000E+0002
  95  1972.TXCNTBLACK.SOYBEANS 3.50000000000000E+0001
  96  1972.TXCNTBLACK.WHEAT    1.42000000000000E+0002
  97  1972.TXEAST.CORN         3.00000000000000E+0001
  98  1972.TXEAST.SORGHUM      5.90000000000000E+0001
  99  1972.TXEAST.SOYBEANS     2.90000000000000E+0001
 100  1972.TXEAST.WHEAT        1.00000000000000E+0000
 101  1972.TXEDPLAT.CORN       1.00000000000000E+0000
 102  1972.TXEDPLAT.SORGHUM    1.00000000000000E+0002
 103  1972.TXEDPLAT.WHEAT      2.80000000000000E+0001
 104  1972.TXHIPLAINS.CORN     1.99000000000000E+0002
 105  1972.TXHIPLAINS.SORGHUM  2.48600000000000E+0003
 106  1972.TXHIPLAINS.SOYBEANS 5.80000000000000E+0001
 107  1972.TXHIPLAINS.WHEAT    1.18500000000000E+0003
 108  1972.TXROLINGPL.SORGHUM  5.18000000000000E+0002
 109  1972.TXROLINGPL.WHEAT    6.09000000000000E+0002
 110  1972.TXSOUTH.CORN        4.50000000000000E+0001
 111  1972.TXSOUTH.SORGHUM     6.01000000000000E+0002
 112  1972.TXSOUTH.SOYBEANS    3.00000000000000E+0000
 113  1972.TXSOUTH.WHEAT       2.30000000000000E+0001
 114  1972.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
 115  1972.TXTRANSPEC.WHEAT    8.00000000000000E+0000
 116  1973.TXCNTBLACK.CORN     1.33000000000000E+0002
 117  1973.TXCNTBLACK.SORGHUM  1.26100000000000E+0003
 118  1973.TXCNTBLACK.SOYBEANS 2.90000000000000E+0001
 119  1973.TXCNTBLACK.WHEAT    2.68000000000000E+0002
 120  1973.TXEAST.CORN         2.70000000000000E+0001
 121  1973.TXEAST.SORGHUM      4.30000000000000E+0001
 122  1973.TXEAST.SOYBEANS     5.90000000000000E+0001
 123  1973.TXEAST.WHEAT        4.00000000000000E+0000
 124  1973.TXEDPLAT.SORGHUM    1.37000000000000E+0002
 125  1973.TXEDPLAT.WHEAT      6.40000000000000E+0001
 126  1973.TXHIPLAINS.CORN     3.67000000000000E+0002
 127  1973.TXHIPLAINS.SORGHUM  3.03000000000000E+0003
 128  1973.TXHIPLAINS.SOYBEANS 1.31000000000000E+0002
 129  1973.TXHIPLAINS.WHEAT    2.18600000000000E+0003
 130  1973.TXROLINGPL.SORGHUM  6.45000000000000E+0002
 131  1973.TXROLINGPL.WHEAT    8.17000000000000E+0002
```

```
132  1973.TXSOUTH.CORN         7.20000000000000E+0001
133  1973.TXSOUTH.SORGHUM      8.93000000000000E+0002
134  1973.TXSOUTH.SOYBEANS     7.00000000000000E+0000
135  1973.TXSOUTH.WHEAT        5.20000000000000E+0001
136  1973.TXTRANSPEC.SORGHUM   8.00000000000000E+0000
137  1973.TXTRANSPEC.WHEAT     5.00000000000000E+0000
138  1974.TXCNTBLACK.CORN      1.25000000000000E+0002
139  1974.TXCNTBLACK.SORGHUM   1.20700000000000E+0003
140  1974.TXCNTBLACK.SOYBEANS  2.40000000000000E+0001
141  1974.TXCNTBLACK.WHEAT     3.93000000000000E+0002
142  1974.TXEAST.CORN          3.40000000000000E+0001
143  1974.TXEAST.SORGHUM       7.40000000000000E+0001
144  1974.TXEAST.SOYBEANS      3.40000000000000E+0001
145  1974.TXEAST.WHEAT         6.00000000000000E+0000
146  1974.TXEDPLAT.CORN        1.00000000000000E+0000
147  1974.TXEDPLAT.SORGHUM     1.31000000000000E+0002
148  1974.TXEDPLAT.WHEAT       7.20000000000000E+0001
149  1974.TXHIPLAINS.CORN      5.41000000000000E+0002
150  1974.TXHIPLAINS.SORGHUM   2.17500000000000E+0003
151  1974.TXHIPLAINS.SOYBEANS  8.80000000000000E+0001
152  1974.TXHIPLAINS.WHEAT     1.66500000000000E+0003
153  1974.TXROLINGPL.CORN      1.00000000000000E+0000
154  1974.TXROLINGPL.SORGHUM   4.00000000000000E+0002
155  1974.TXROLINGPL.WHEAT     1.09000000000000E+0003
156  1974.TXSOUTH.CORN         6.10000000000000E+0001
157  1974.TXSOUTH.SORGHUM      9.97000000000000E+0002
158  1974.TXSOUTH.SOYBEANS     4.00000000000000E+0000
159  1974.TXSOUTH.WHEAT        6.60000000000000E+0001
160  1974.TXTRANSPEC.SORGHUM   9.00000000000000E+0000
161  1974.TXTRANSPEC.WHEAT     5.00000000000000E+0000
162  1975.TXCNTBLACK.CORN      1.39000000000000E+0002
163  1975.TXCNTBLACK.SORGHUM   1.49800000000000E+0003
164  1975.TXCNTBLACK.SOYBEANS  4.30000000000000E+0001
165  1975.TXCNTBLACK.WHEAT     5.19000000000000E+0002
166  1975.TXEAST.CORN          4.10000000000000E+0001
167  1975.TXEAST.SORGHUM       6.00000000000000E+0001
168  1975.TXEAST.SOYBEANS      4.00000000000000E+0001
169  1975.TXEAST.WHEAT         3.00000000000000E+0000
170  1975.TXEDPLAT.CORN        2.00000000000000E+0000
171  1975.TXEDPLAT.SORGHUM     1.38000000000000E+0002
172  1975.TXEDPLAT.WHEAT       1.54000000000000E+0002
173  1975.TXHIPLAINS.CORN      7.71000000000000E+0002
174  1975.TXHIPLAINS.SORGHUM   2.62900000000000E+0003
175  1975.TXHIPLAINS.SOYBEANS  1.27000000000000E+0002
176  1975.TXHIPLAINS.WHEAT     3.06000000000000E+0003
177  1975.TXROLINGPL.SORGHUM   6.17000000000000E+0002
178  1975.TXROLINGPL.WHEAT     1.84000000000000E+0003
179  1975.TXSOUTH.CORN         9.10000000000000E+0001
180  1975.TXSOUTH.SORGHUM      1.17100000000000E+0003
181  1975.TXSOUTH.SOYBEANS     5.00000000000000E+0000
182  1975.TXSOUTH.WHEAT        9.20000000000000E+0001
183  1975.TXTRANSPEC.SORGHUM   2.00000000000000E+0000
184  1975.TXTRANSPEC.WHEAT     2.60000000000000E+0001
185  1976.TXCNTBLACK.CORN      1.32000000000000E+0002
186  1976.TXCNTBLACK.SORGHUM   1.27000000000000E+0003
187  1976.TXCNTBLACK.SOYBEANS  4.80000000000000E+0001
188  1976.TXCNTBLACK.WHEAT     6.90000000000000E+0002
189  1976.TXEAST.CORN          2.10000000000000E+0001
190  1976.TXEAST.SORGHUM       4.80000000000000E+0001
191  1976.TXEAST.SOYBEANS      5.30000000000000E+0001
192  1976.TXEAST.WHEAT         1.70000000000000E+0001
193  1976.TXEDPLAT.CORN        2.00000000000000E+0000
194  1976.TXEDPLAT.SORGHUM     1.49000000000000E+0002
195  1976.TXEDPLAT.WHEAT       7.40000000000000E+0001
196  1976.TXHIPLAINS.CORN      1.21000000000000E+0003
197  1976.TXHIPLAINS.SORGHUM   1.89000000000000E+0003
198  1976.TXHIPLAINS.SOYBEANS  7.50000000000000E+0001
```

```
199  1976.TXHIPLAINS.WHEAT    2.12500000000000E+0003
200  1976.TXROLINGPL.CORN     4.00000000000000E+0000
201  1976.TXROLINGPL.SORGHUM  4.79000000000000E+0002
202  1976.TXROLINGPL.WHEAT    1.66500000000000E+0003
203  1976.TXSOUTH.CORN        7.10000000000000E+0001
204  1976.TXSOUTH.SORGHUM     7.57000000000000E+0002
205  1976.TXSOUTH.WHEAT       8.10000000000000E+0001
206  1976.TXTRANSPEC.WHEAT    4.00000000000000E+0001
207  1977.TXCNTBLACK.CORN     1.65000000000000E+0002
208  1977.TXCNTBLACK.SORGHUM  1.32800000000000E+0003
209  1977.TXCNTBLACK.SOYBEANS 7.40000000000000E+0001
210  1977.TXCNTBLACK.WHEAT    5.55000000000000E+0002
211  1977.TXEAST.CORN         3.70000000000000E+0001
212  1977.TXEAST.SORGHUM      3.00000000000000E+0001
213  1977.TXEAST.SOYBEANS     1.03000000000000E+0002
214  1977.TXEAST.WHEAT        1.90000000000000E+0001
215  1977.TXEDPLAT.CORN       5.00000000000000E+0000
216  1977.TXEDPLAT.SORGHUM    1.21000000000000E+0002
217  1977.TXEDPLAT.WHEAT      1.24000000000000E+0002
218  1977.TXHIPLAINS.CORN     1.21100000000000E+0003
219  1977.TXHIPLAINS.SORGHUM  1.32000000000000E+0003
220  1977.TXHIPLAINS.SOYBEANS 1.52000000000000E+0002
221  1977.TXHIPLAINS.WHEAT    2.33700000000000E+0003
222  1977.TXROLINGPL.CORN     3.00000000000000E+0000
223  1977.TXROLINGPL.SORGHUM  3.35000000000000E+0002
224  1977.TXROLINGPL.WHEAT    1.58000000000000E+0003
225  1977.TXSOUTH.CORN        1.19000000000000E+0002
226  1977.TXSOUTH.SORGHUM     6.66000000000000E+0002
227  1977.TXSOUTH.SOYBEANS    8.00000000000000E+0000
228  1977.TXSOUTH.WHEAT       6.20000000000000E+0001
229  1977.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
230  1977.TXTRANSPEC.WHEAT    1.90000000000000E+0001
231  1978.TXCNTBLACK.CORN     1.99000000000000E+0002
232  1978.TXCNTBLACK.SORGHUM  1.35100000000000E+0003
233  1978.TXCNTBLACK.SOYBEANS 5.10000000000000E+0001
234  1978.TXCNTBLACK.WHEAT    4.74000000000000E+0002
235  1978.TXEAST.CORN         3.30000000000000E+0001
236  1978.TXEAST.SORGHUM      3.10000000000000E+0001
237  1978.TXEAST.SOYBEANS     1.03000000000000E+0002
238  1978.TXEAST.WHEAT        1.80000000000000E+0001
239  1978.TXEDPLAT.CORN       3.00000000000000E+0000
240  1978.TXEDPLAT.SORGHUM    8.80000000000000E+0001
241  1978.TXEDPLAT.WHEAT      4.60000000000000E+0001
242  1978.TXHIPLAINS.CORN     9.07000000000000E+0002
243  1978.TXHIPLAINS.SORGHUM  1.37000000000000E+0003
244  1978.TXHIPLAINS.SOYBEANS 1.88000000000000E+0002
245  1978.TXHIPLAINS.WHEAT    1.28500000000000E+0003
246  1978.TXROLINGPL.CORN     2.00000000000000E+0000
247  1978.TXROLINGPL.SORGHUM  2.80000000000000E+0002
248  1978.TXROLINGPL.WHEAT    8.34000000000000E+0002
249  1978.TXSOUTH.CORN        1.44000000000000E+0002
250  1978.TXSOUTH.SORGHUM     6.40000000000000E+0002
251  1978.TXSOUTH.SOYBEANS    6.00000000000000E+0000
252  1978.TXSOUTH.WHEAT       3.30000000000000E+0001
253  1978.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
254  1978.TXTRANSPEC.WHEAT    5.00000000000000E+0000
255  1979.TXCNTBLACK.CORN     2.09000000000000E+0002
256  1979.TXCNTBLACK.SORGHUM  1.18800000000000E+0003
257  1979.TXCNTBLACK.SOYBEANS 6.90000000000000E+0001
258  1979.TXCNTBLACK.WHEAT    6.20000000000000E+0002
259  1979.TXEAST.CORN         3.70000000000000E+0001
260  1979.TXEAST.SORGHUM      2.40000000000000E+0001
261  1979.TXEAST.SOYBEANS     1.10000000000000E+0002
262  1979.TXEAST.WHEAT        1.70000000000000E+0001
263  1979.TXEDPLAT.CORN       3.00000000000000E+0000
264  1979.TXEDPLAT.SORGHUM    1.21000000000000E+0002
265  1979.TXEDPLAT.WHEAT      1.15000000000000E+0002
```

```
266  1979.TXHIPLAINS.CORN     7.00000000000000E+0002
267  1979.TXHIPLAINS.SORGHUM  1.23000000000000E+0003
268  1979.TXHIPLAINS.SOYBEANS 2.17000000000000E+0002
269  1979.TXHIPLAINS.WHEAT    2.39300000000000E+0003
270  1979.TXROLINGPL.CORN     3.00000000000000E+0000
271  1979.TXROLINGPL.SORGHUM  2.90000000000000E+0002
272  1979.TXROLINGPL.WHEAT    1.37000000000000E+0003
273  1979.TXSOUTH.CORN        1.73000000000000E+0002
274  1979.TXSOUTH.SORGHUM     7.31000000000000E+0002
275  1979.TXSOUTH.SOYBEANS    1.60000000000000E+0001
276  1979.TXSOUTH.WHEAT       7.20000000000000E+0001
277  1979.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
278  1979.TXTRANSPEC.WHEAT    5.00000000000000E+0000
279  1980.TXCNTBLACK.CORN     2.26000000000000E+0002
280  1980.TXCNTBLACK.SORGHUM  9.83000000000000E+0002
281  1980.TXCNTBLACK.SOYBEANS 3.70000000000000E+0001
282  1980.TXCNTBLACK.WHEAT    1.11600000000000E+0003
283  1980.TXEAST.CORN         2.80000000000000E+0001
284  1980.TXEAST.SORGHUM      2.70000000000000E+0001
285  1980.TXEAST.SOYBEANS     1.22000000000000E+0002
286  1980.TXEAST.WHEAT        6.20000000000000E+0001
287  1980.TXEDPLAT.CORN       2.00000000000000E+0000
288  1980.TXEDPLAT.SORGHUM    7.90000000000000E+0001
289  1980.TXEDPLAT.WHEAT      8.50000000000000E+0001
290  1980.TXHIPLAINS.CORN     7.35000000000000E+0002
291  1980.TXHIPLAINS.SORGHUM  1.13700000000000E+0003
292  1980.TXHIPLAINS.SOYBEANS 6.50000000000000E+0001
293  1980.TXHIPLAINS.WHEAT    2.59500000000000E+0003
294  1980.TXROLINGPL.CORN     3.00000000000000E+0000
295  1980.TXROLINGPL.SORGHUM  1.68000000000000E+0002
296  1980.TXROLINGPL.WHEAT    1.25500000000000E+0003
297  1980.TXSOUTH.CORN        1.54000000000000E+0002
298  1980.TXSOUTH.SORGHUM     6.26000000000000E+0002
299  1980.TXSOUTH.SOYBEANS    5.00000000000000E+0000
300  1980.TXSOUTH.WHEAT       4.60000000000000E+0001
301  1980.TXTRANSPEC.WHEAT    2.30000000000000E+0001
302  1981.TXCNTBLACK.CORN     2.12000000000000E+0002
303  1981.TXCNTBLACK.SORGHUM  8.94000000000000E+0002
304  1981.TXCNTBLACK.SOYBEANS 2.20000000000000E+0001
305  1981.TXCNTBLACK.WHEAT    1.51100000000000E+0003
306  1981.TXEAST.CORN         3.10000000000000E+0001
307  1981.TXEAST.SORGHUM      4.10000000000000E+0001
308  1981.TXEAST.SOYBEANS     7.30000000000000E+0001
309  1981.TXEAST.WHEAT        1.42000000000000E+0002
310  1981.TXEDPLAT.CORN       1.00000000000000E+0000
311  1981.TXEDPLAT.SORGHUM    8.80000000000000E+0001
312  1981.TXEDPLAT.WHEAT      1.45000000000000E+0002
313  1981.TXHIPLAINS.CORN     5.22000000000000E+0002
314  1981.TXHIPLAINS.SORGHUM  1.23900000000000E+0003
315  1981.TXHIPLAINS.SOYBEANS 5.80000000000000E+0001
316  1981.TXHIPLAINS.WHEAT    2.75000000000000E+0003
317  1981.TXROLINGPL.CORN     1.00000000000000E+0000
318  1981.TXROLINGPL.SORGHUM  1.95000000000000E+0002
319  1981.TXROLINGPL.WHEAT    1.86400000000000E+0003
320  1981.TXSOUTH.CORN        1.87000000000000E+0002
321  1981.TXSOUTH.SORGHUM     8.22000000000000E+0002
322  1981.TXSOUTH.WHEAT       8.70000000000000E+0001
323  1981.TXTRANSPEC.SORGHUM  9.00000000000000E+0000
324  1981.TXTRANSPEC.WHEAT    3.80000000000000E+0001
325  1982.TXCNTBLACK.CORN     2.37000000000000E+0002
326  1982.TXCNTBLACK.SORGHUM  1.06200000000000E+0003
327  1982.TXCNTBLACK.SOYBEANS 5.50000000000000E+0001
328  1982.TXCNTBLACK.WHEAT    1.43700000000000E+0003
329  1982.TXEAST.CORN         2.50000000000000E+0001
330  1982.TXEAST.SORGHUM      4.40000000000000E+0001
331  1982.TXEAST.SOYBEANS     5.80000000000000E+0001
332  1982.TXEAST.WHEAT        1.02000000000000E+0002
```

```
333  1982.TXEDPLAT.CORN       2.00000000000000E+0000
334  1982.TXEDPLAT.SORGHUM    1.09000000000000E+0002
335  1982.TXEDPLAT.WHEAT      1.78000000000000E+0002
336  1982.TXHIPLAINS.CORN     5.23000000000000E+0002
337  1982.TXHIPLAINS.SORGHUM  2.18000000000000E+0003
338  1982.TXHIPLAINS.SOYBEANS 4.45000000000000E+0002
339  1982.TXHIPLAINS.WHEAT    2.52000000000000E+0003
340  1982.TXROLINGPL.CORN     2.00000000000000E+0000
341  1982.TXROLINGPL.SORGHUM  3.57000000000000E+0002
342  1982.TXROLINGPL.WHEAT    1.60500000000000E+0003
343  1982.TXSOUTH.CORN        1.67000000000000E+0002
344  1982.TXSOUTH.SORGHUM     6.82000000000000E+0002
345  1982.TXSOUTH.SOYBEANS    2.00000000000000E+0000
346  1982.TXSOUTH.WHEAT       1.08000000000000E+0002
347  1982.TXTRANSPEC.SORGHUM  2.00000000000000E+0000
348  1982.TXTRANSPEC.WHEAT    1.80000000000000E+0001
349  1983.TXCNTBLACK.CORN     2.66000000000000E+0002
350  1983.TXCNTBLACK.SORGHUM  6.57000000000000E+0002
351  1983.TXCNTBLACK.SOYBEANS 4.90000000000000E+0001
352  1983.TXCNTBLACK.WHEAT    1.06900000000000E+0003
353  1983.TXEAST.CORN         2.30000000000000E+0001
354  1983.TXEAST.SORGHUM      3.00000000000000E+0001
355  1983.TXEAST.SOYBEANS     3.40000000000000E+0001
356  1983.TXEAST.WHEAT        8.40000000000000E+0001
357  1983.TXEDPLAT.CORN       1.00000000000000E+0000
358  1983.TXEDPLAT.SORGHUM    8.90000000000000E+0001
359  1983.TXEDPLAT.WHEAT      9.70000000000000E+0001
360  1983.TXHIPLAINS.CORN     4.00000000000000E+0002
361  1983.TXHIPLAINS.SORGHUM  9.40000000000000E+0002
362  1983.TXHIPLAINS.SOYBEANS 1.08000000000000E+0002
363  1983.TXHIPLAINS.WHEAT    1.98500000000000E+0003
364  1983.TXROLINGPL.CORN     1.00000000000000E+0000
365  1983.TXROLINGPL.SORGHUM  2.37000000000000E+0002
366  1983.TXROLINGPL.WHEAT    1.27500000000000E+0003
367  1983.TXSOUTH.CORN        2.03000000000000E+0002
368  1983.TXSOUTH.SORGHUM     5.89000000000000E+0002
369  1983.TXSOUTH.WHEAT       5.10000000000000E+0001
370  1983.TXTRANSPEC.WHEAT    1.50000000000000E+0001
371  1984.TXCNTBLACK.CORN     4.15000000000000E+0002
372  1984.TXCNTBLACK.SORGHUM  8.62000000000000E+0002
373  1984.TXCNTBLACK.SOYBEANS 3.20000000000000E+0001
374  1984.TXCNTBLACK.WHEAT    1.17200000000000E+0003
375  1984.TXEAST.CORN         3.50000000000000E+0001
376  1984.TXEAST.SORGHUM      2.60000000000000E+0001
377  1984.TXEAST.SOYBEANS     4.40000000000000E+0001
378  1984.TXEAST.WHEAT        6.60000000000000E+0001
379  1984.TXEDPLAT.CORN       1.00000000000000E+0000
380  1984.TXEDPLAT.SORGHUM    7.20000000000000E+0001
381  1984.TXEDPLAT.WHEAT      9.00000000000000E+0001
382  1984.TXHIPLAINS.CORN     5.32000000000000E+0002
383  1984.TXHIPLAINS.SORGHUM  1.29500000000000E+0003
384  1984.TXHIPLAINS.SOYBEANS 4.00000000000000E+0001
385  1984.TXHIPLAINS.WHEAT    2.33000000000000E+0003
386  1984.TXROLINGPL.CORN     3.00000000000000E+0000
387  1984.TXROLINGPL.SORGHUM  1.74000000000000E+0002
388  1984.TXROLINGPL.WHEAT    1.26000000000000E+0003
389  1984.TXSOUTH.CORN        2.20000000000000E+0002
390  1984.TXSOUTH.SORGHUM     6.09000000000000E+0002
391  1984.TXSOUTH.WHEAT       4.50000000000000E+0001
392  1984.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
393  1984.TXTRANSPEC.WHEAT    1.00000000000000E+0001
394  1985.TXCNTBLACK.CORN     3.61000000000000E+0002
395  1985.TXCNTBLACK.SORGHUM  8.61000000000000E+0002
396  1985.TXCNTBLACK.SOYBEANS 1.70000000000000E+0001
397  1985.TXCNTBLACK.WHEAT    1.17500000000000E+0003
398  1985.TXEAST.CORN         4.30000000000000E+0001
399  1985.TXEAST.SORGHUM      2.50000000000000E+0001
```

```
400  1985.TXEAST.SOYBEANS     2.00000000000000E+0001
401  1985.TXEAST.WHEAT        1.29000000000000E+0002
402  1985.TXEDPLAT.CORN       2.80000000000000E+0001
403  1985.TXEDPLAT.SORGHUM    8.70000000000000E+0001
404  1985.TXEDPLAT.SOYBEANS   1.00000000000000E+0000
405  1985.TXEDPLAT.WHEAT      1.28000000000000E+0002
406  1985.TXHIPLAINS.CORN     5.17000000000000E+0002
407  1985.TXHIPLAINS.SORGHUM  1.56700000000000E+0003
408  1985.TXHIPLAINS.SOYBEANS 7.50000000000000E+0001
409  1985.TXHIPLAINS.WHEAT    2.62000000000000E+0003
410  1985.TXROLINGPL.CORN     6.00000000000000E+0000
411  1985.TXROLINGPL.SORGHUM  2.87000000000000E+0002
412  1985.TXROLINGPL.WHEAT    1.68000000000000E+0003
413  1985.TXSOUTH.CORN        2.16000000000000E+0002
414  1985.TXSOUTH.SORGHUM     5.91000000000000E+0002
415  1985.TXSOUTH.WHEAT       9.30000000000000E+0001
416  1985.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
417  1985.TXTRANSPEC.WHEAT    6.00000000000000E+0000
418  1986.TXCNTBLACK.CORN     4.03000000000000E+0002
419  1986.TXCNTBLACK.SORGHUM  7.64000000000000E+0002
420  1986.TXCNTBLACK.SOYBEANS 2.60000000000000E+0001
421  1986.TXCNTBLACK.WHEAT    8.45000000000000E+0002
422  1986.TXEAST.CORN         5.60000000000000E+0001
423  1986.TXEAST.SORGHUM      5.10000000000000E+0001
424  1986.TXEAST.SOYBEANS     3.20000000000000E+0001
425  1986.TXEAST.WHEAT        4.00000000000000E+0001
426  1986.TXEDPLAT.CORN       6.00000000000000E+0000
427  1986.TXEDPLAT.SORGHUM    7.90000000000000E+0001
428  1986.TXEDPLAT.WHEAT      8.20000000000000E+0001
429  1986.TXHIPLAINS.CORN     4.67000000000000E+0002
430  1986.TXHIPLAINS.SORGHUM  1.46800000000000E+0003
431  1986.TXHIPLAINS.SOYBEANS 5.60000000000000E+0001
432  1986.TXHIPLAINS.WHEAT    2.34000000000000E+0003
433  1986.TXROLINGPL.CORN     5.00000000000000E+0000
434  1986.TXROLINGPL.SORGHUM  2.49000000000000E+0002
435  1986.TXROLINGPL.WHEAT    1.40000000000000E+0003
436  1986.TXSOUTH.CORN        1.63000000000000E+0002
437  1986.TXSOUTH.SORGHUM     4.76000000000000E+0002
438  1986.TXSOUTH.WHEAT       6.70000000000000E+0001
439  1986.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
440  1986.TXTRANSPEC.WHEAT    7.00000000000000E+0000
441  1987.TXCNTBLACK.CORN     3.83000000000000E+0002
442  1987.TXCNTBLACK.SORGHUM  6.07000000000000E+0002
443  1987.TXCNTBLACK.SOYBEANS 3.10000000000000E+0001
444  1987.TXCNTBLACK.WHEAT    5.52000000000000E+0002
445  1987.TXEAST.CORN         3.00000000000000E+0001
446  1987.TXEAST.SORGHUM      3.60000000000000E+0001
447  1987.TXEAST.SOYBEANS     2.10000000000000E+0001
448  1987.TXEAST.WHEAT        4.30000000000000E+0001
449  1987.TXEDPLAT.CORN       3.00000000000000E+0000
450  1987.TXEDPLAT.SORGHUM    6.80000000000000E+0001
451  1987.TXEDPLAT.WHEAT      8.70000000000000E+0001
452  1987.TXHIPLAINS.CORN     4.12000000000000E+0002
453  1987.TXHIPLAINS.SORGHUM  8.95000000000000E+0002
454  1987.TXHIPLAINS.SOYBEANS 4.00000000000000E+0001
455  1987.TXHIPLAINS.WHEAT    1.73000000000000E+0003
456  1987.TXROLINGPL.CORN     3.00000000000000E+0000
457  1987.TXROLINGPL.SORGHUM  1.91000000000000E+0002
458  1987.TXROLINGPL.WHEAT    1.11300000000000E+0003
459  1987.TXSOUTH.CORN        1.70000000000000E+0002
460  1987.TXSOUTH.SORGHUM     3.61000000000000E+0002
461  1987.TXSOUTH.WHEAT       6.30000000000000E+0001
462  1987.TXTRANSPEC.SORGHUM  8.00000000000000E+0000
463  1987.TXTRANSPEC.WHEAT    3.00000000000000E+0000
464  1988.TXCNTBLACK.CORN     3.92000000000000E+0002
465  1988.TXCNTBLACK.SORGHUM  4.99000000000000E+0002
466  1988.TXCNTBLACK.SOYBEANS 2.50000000000000E+0001
```

```
467  1988.TXCNTBLACK.WHEAT    6.29000000000000E+0002
468  1988.TXEAST.CORN         3.60000000000000E+0001
469  1988.TXEAST.SORGHUM      2.10000000000000E+0001
470  1988.TXEAST.SOYBEANS     3.10000000000000E+0001
471  1988.TXEAST.WHEAT        4.50000000000000E+0001
472  1988.TXEDPLAT.CORN       3.00000000000000E+0000
473  1988.TXEDPLAT.SORGHUM    5.60000000000000E+0001
474  1988.TXEDPLAT.WHEAT      5.80000000000000E+0001
475  1988.TXHIPLAINS.CORN     4.23000000000000E+0002
476  1988.TXHIPLAINS.SORGHUM  6.68000000000000E+0002
477  1988.TXHIPLAINS.SOYBEANS 6.10000000000000E+0001
478  1988.TXHIPLAINS.WHEAT    1.47200000000000E+0003
479  1988.TXROLINGPL.CORN     3.00000000000000E+0000
480  1988.TXROLINGPL.SORGHUM  1.23000000000000E+0002
481  1988.TXROLINGPL.WHEAT    9.30000000000000E+0002
482  1988.TXSOUTH.CORN        2.22000000000000E+0002
483  1988.TXSOUTH.SORGHUM     2.82000000000000E+0002
484  1988.TXSOUTH.WHEAT       5.40000000000000E+0001
485  1988.TXTRANSPEC.SORGHUM  9.00000000000000E+0000
486  1988.TXTRANSPEC.WHEAT    3.00000000000000E+0000
487  1989.TXCNTBLACK.CORN     3.69000000000000E+0002
488  1989.TXCNTBLACK.SORGHUM  6.43000000000000E+0002
489  1989.TXCNTBLACK.SOYBEANS 4.90000000000000E+0001
490  1989.TXCNTBLACK.WHEAT    7.74000000000000E+0002
491  1989.TXEAST.CORN         3.60000000000000E+0001
492  1989.TXEAST.SORGHUM      3.00000000000000E+0001
493  1989.TXEAST.SOYBEANS     4.20000000000000E+0001
494  1989.TXEAST.WHEAT        5.50000000000000E+0001
495  1989.TXEDPLAT.CORN       3.00000000000000E+0000
496  1989.TXEDPLAT.SORGHUM    6.30000000000000E+0001
497  1989.TXEDPLAT.WHEAT      1.21000000000000E+0002
498  1989.TXHIPLAINS.CORN     5.92000000000000E+0002
499  1989.TXHIPLAINS.SORGHUM  1.20800000000000E+0003
500  1989.TXHIPLAINS.SOYBEANS 1.78000000000000E+0002
501  1989.TXHIPLAINS.WHEAT    7.48000000000000E+0002
502  1989.TXROLINGPL.CORN     3.00000000000000E+0000
503  1989.TXROLINGPL.SORGHUM  1.32000000000000E+0002
504  1989.TXROLINGPL.WHEAT    1.22500000000000E+0003
505  1989.TXSOUTH.CORN        1.82000000000000E+0002
506  1989.TXSOUTH.SORGHUM     3.08000000000000E+0002
507  1989.TXSOUTH.SOYBEANS    5.00000000000000E+0000
508  1989.TXSOUTH.WHEAT       6.60000000000000E+0001
509  1989.TXTRANSPEC.SORGHUM  2.00000000000000E+0000
510  1989.TXTRANSPEC.WHEAT    2.00000000000000E+0000
511  1990.TXCNTBLACK.CORN     3.63000000000000E+0002
512  1990.TXCNTBLACK.SORGHUM  6.45000000000000E+0002
513  1990.TXCNTBLACK.SOYBEANS 4.40000000000000E+0001
514  1990.TXCNTBLACK.WHEAT    8.33000000000000E+0002
515  1990.TXEAST.CORN         3.40000000000000E+0001
516  1990.TXEAST.SORGHUM      2.30000000000000E+0001
517  1990.TXEAST.SOYBEANS     3.40000000000000E+0001
518  1990.TXEAST.WHEAT        5.30000000000000E+0001
519  1990.TXEDPLAT.CORN       3.00000000000000E+0000
520  1990.TXEDPLAT.SORGHUM    6.30000000000000E+0001
521  1990.TXEDPLAT.WHEAT      1.27000000000000E+0002
522  1990.TXHIPLAINS.CORN     6.21000000000000E+0002
523  1990.TXHIPLAINS.SORGHUM  6.62000000000000E+0002
524  1990.TXHIPLAINS.SOYBEANS 2.80000000000000E+0001
525  1990.TXHIPLAINS.WHEAT    1.73600000000000E+0003
526  1990.TXROLINGPL.CORN     7.00000000000000E+0000
527  1990.TXROLINGPL.SORGHUM  1.26000000000000E+0002
528  1990.TXROLINGPL.WHEAT    1.35400000000000E+0003
529  1990.TXSOUTH.CORN        2.30000000000000E+0002
530  1990.TXSOUTH.SORGHUM     3.67000000000000E+0002
531  1990.TXSOUTH.SOYBEANS    2.00000000000000E+0000
532  1990.TXSOUTH.WHEAT       8.30000000000000E+0001
533  1991.TXCNTBLACK.CORN     3.75000000000000E+0002
```

```
 534  1991.TXCNTBLACK.SORGHUM  7.20000000000000E+0002
 535  1991.TXCNTBLACK.SOYBEANS 3.80000000000000E+0001
 536  1991.TXCNTBLACK.WHEAT    5.56000000000000E+0002
 537  1991.TXEAST.CORN         3.50000000000000E+0001
 538  1991.TXEAST.SORGHUM      2.60000000000000E+0001
 539  1991.TXEAST.SOYBEANS     2.90000000000000E+0001
 540  1991.TXEAST.WHEAT        3.50000000000000E+0001
 541  1991.TXEDPLAT.CORN       3.00000000000000E+0000
 542  1991.TXEDPLAT.SORGHUM    7.00000000000000E+0001
 543  1991.TXEDPLAT.WHEAT      8.50000000000000E+0001
 544  1991.TXHIPLAINS.CORN     6.42000000000000E+0002
 545  1991.TXHIPLAINS.SORGHUM  7.38000000000000E+0002
 546  1991.TXHIPLAINS.SOYBEANS 2.40000000000000E+0001
 547  1991.TXHIPLAINS.WHEAT    1.15700000000000E+0003
 548  1991.TXROLINGPL.CORN     7.00000000000000E+0000
 549  1991.TXROLINGPL.SORGHUM  1.41000000000000E+0002
 550  1991.TXROLINGPL.WHEAT    9.03000000000000E+0002
 551  1991.TXSOUTH.CORN        2.38000000000000E+0002
 552  1991.TXSOUTH.SORGHUM     4.09000000000000E+0002
 553  1991.TXSOUTH.SOYBEANS    2.00000000000000E+0000
 554  1991.TXSOUTH.WHEAT       5.60000000000000E+0001
 555  1991.TXTRANSPEC.SORGHUM  1.00000000000000E+0000
 556  1992.TXCNTBLACK.CORN     4.05000000000000E+0002
 557  1992.TXCNTBLACK.SORGHUM  1.11700000000000E+0003
 558  1992.TXCNTBLACK.SOYBEANS 8.70000000000000E+0001
 559  1992.TXCNTBLACK.WHEAT    7.54000000000000E+0002
 560  1992.TXEAST.CORN         3.80000000000000E+0001
 561  1992.TXEAST.SORGHUM      4.00000000000000E+0001
 562  1992.TXEAST.SOYBEANS     6.60000000000000E+0001
 563  1992.TXEAST.WHEAT        4.80000000000000E+0001
 564  1992.TXEDPLAT.CORN       3.00000000000000E+0000
 565  1992.TXEDPLAT.SORGHUM    1.09000000000000E+0002
 566  1992.TXEDPLAT.SOYBEANS   2.00000000000000E+0000
 567  1992.TXEDPLAT.WHEAT      1.15000000000000E+0002
 568  1992.TXHIPLAINS.CORN     6.94000000000000E+0002
 569  1992.TXHIPLAINS.SORGHUM  1.14600000000000E+0003
 570  1992.TXHIPLAINS.SOYBEANS 5.60000000000000E+0001
 571  1992.TXHIPLAINS.WHEAT    1.57100000000000E+0003
 572  1992.TXROLINGPL.CORN     8.00000000000000E+0000
 573  1992.TXROLINGPL.SORGHUM  2.18000000000000E+0002
 574  1992.TXROLINGPL.WHEAT    1.22500000000000E+0003
 575  1992.TXSOUTH.CORN        2.57000000000000E+0002
 576  1992.TXSOUTH.SORGHUM     6.35000000000000E+0002
 577  1992.TXSOUTH.SOYBEANS    4.00000000000000E+0000
 578  1992.TXSOUTH.WHEAT       7.60000000000000E+0001
 579  1992.TXTRANSPEC.SORGHUM  2.00000000000000E+0000
 580  /;
 581  option txacre:0:2:1;display txacre;
 582
 583  parameter subacre(year,subreg)   Texas subreg acres;
 584


LIBINCLUDE C:\GAMS25\SSIMPORT.GMS
 635  parameter subacre /
INCLUDE    C:\GAMS25\225A\GAMSLINK.SCR
 637  *  Tuesday, June 30 1998   15:28:19
 638  *  C:\GAMS25\IM2.WK1  SUBACRE
 639  1971.TXCNTBLACK               1.23000000000000E+0003
 640  1971.TXEAST                   1.10000000000000E+0002
 641  1971.TXEDPLAT                 1.16000000000000E+0002
 642  1971.TXHIPLAINS               4.25000000000000E+0003
 643  1971.TXROLINGPL               8.38000000000000E+0002
 644  1971.TXSOUTH                  6.54000000000000E+0002
 645  1971.TXTRANSPEC               1.50000000000000E+0001
 646  1972.TXCNTBLACK               1.30700000000000E+0003
 647  1972.TXEAST                   1.19000000000000E+0002
```

648	1972.TXEDPLAT	1.29000000000000E+0002
649	1972.TXHIPLAINS	3.92800000000000E+0003
650	1972.TXROLINGPL	1.12700000000000E+0003
651	1972.TXSOUTH	6.72000000000000E+0002
652	1972.TXTRANSPEC	1.60000000000000E+0001
653	1973.TXCNTBLACK	1.69100000000000E+0003
654	1973.TXEAST	1.33000000000000E+0002
655	1973.TXEDPLAT	2.01000000000000E+0002
656	1973.TXHIPLAINS	5.71400000000000E+0003
657	1973.TXROLINGPL	1.46200000000000E+0003
658	1973.TXSOUTH	1.02400000000000E+0003
659	1973.TXTRANSPEC	1.30000000000000E+0001
660	1974.TXCNTBLACK	1.74900000000000E+0003
661	1974.TXEAST	1.48000000000000E+0002
662	1974.TXEDPLAT	2.04000000000000E+0002
663	1974.TXHIPLAINS	4.46900000000000E+0003
664	1974.TXROLINGPL	1.49100000000000E+0003
665	1974.TXSOUTH	1.12800000000000E+0003
666	1974.TXTRANSPEC	1.40000000000000E+0001
667	1975.TXCNTBLACK	2.19900000000000E+0003
668	1975.TXEAST	1.44000000000000E+0002
669	1975.TXEDPLAT	2.94000000000000E+0002
670	1975.TXHIPLAINS	6.58700000000000E+0003
671	1975.TXROLINGPL	2.45700000000000E+0003
672	1975.TXSOUTH	1.35900000000000E+0003
673	1975.TXTRANSPEC	2.80000000000000E+0001
674	1976.TXCNTBLACK	2.14000000000000E+0003
675	1976.TXEAST	1.39000000000000E+0002
676	1976.TXEDPLAT	2.25000000000000E+0002
677	1976.TXHIPLAINS	5.30000000000000E+0003
678	1976.TXROLINGPL	2.14800000000000E+0003
679	1976.TXSOUTH	9.09000000000000E+0002
680	1976.TXTRANSPEC	4.00000000000000E+0001
681	1977.TXCNTBLACK	2.12200000000000E+0003
682	1977.TXEAST	1.89000000000000E+0002
683	1977.TXEDPLAT	2.50000000000000E+0002
684	1977.TXHIPLAINS	5.02000000000000E+0003
685	1977.TXROLINGPL	1.91800000000000E+0003
686	1977.TXSOUTH	8.55000000000000E+0002
687	1977.TXTRANSPEC	2.70000000000000E+0001
688	1978.TXCNTBLACK	2.07500000000000E+0003
689	1978.TXEAST	1.85000000000000E+0002
690	1978.TXEDPLAT	1.37000000000000E+0002
691	1978.TXHIPLAINS	3.75000000000000E+0003
692	1978.TXROLINGPL	1.11600000000000E+0003
693	1978.TXSOUTH	8.23000000000000E+0002
694	1978.TXTRANSPEC	1.30000000000000E+0001
695	1979.TXCNTBLACK	2.08600000000000E+0003
696	1979.TXEAST	1.88000000000000E+0002
697	1979.TXEDPLAT	2.39000000000000E+0002
698	1979.TXHIPLAINS	4.54000000000000E+0003
699	1979.TXROLINGPL	1.66300000000000E+0003
700	1979.TXSOUTH	9.92000000000000E+0002
701	1979.TXTRANSPEC	1.30000000000000E+0001
702	1980.TXCNTBLACK	2.36200000000000E+0003
703	1980.TXEAST	2.39000000000000E+0002
704	1980.TXEDPLAT	1.66000000000000E+0002
705	1980.TXHIPLAINS	4.53200000000000E+0003
706	1980.TXROLINGPL	1.42600000000000E+0003
707	1980.TXSOUTH	8.31000000000000E+0002
708	1980.TXTRANSPEC	2.30000000000000E+0001
709	1981.TXCNTBLACK	2.63900000000000E+0003
710	1981.TXEAST	2.87000000000000E+0002
711	1981.TXEDPLAT	2.34000000000000E+0002
712	1981.TXHIPLAINS	4.56900000000000E+0003
713	1981.TXROLINGPL	2.06000000000000E+0003
714	1981.TXSOUTH	1.09600000000000E+0003

```
715  1981.TXTRANSPEC              4.70000000000000E+0001
716  1982.TXCNTBLACK              2.79100000000000E+0003
717  1982.TXEAST                  2.29000000000000E+0002
718  1982.TXEDPLAT                2.89000000000000E+0002
719  1982.TXHIPLAINS              5.66800000000000E+0003
720  1982.TXROLINGPL              1.96400000000000E+0003
721  1982.TXSOUTH                 9.59000000000000E+0002
722  1982.TXTRANSPEC              2.00000000000000E+0001
723  1983.TXCNTBLACK              2.04100000000000E+0003
724  1983.TXEAST                  1.71000000000000E+0002
725  1983.TXEDPLAT                1.87000000000000E+0002
726  1983.TXHIPLAINS              3.43300000000000E+0003
727  1983.TXROLINGPL              1.51300000000000E+0003
728  1983.TXSOUTH                 8.43000000000000E+0002
729  1983.TXTRANSPEC              1.50000000000000E+0001
730  1984.TXCNTBLACK              2.48100000000000E+0003
731  1984.TXEAST                  1.71000000000000E+0002
732  1984.TXEDPLAT                1.63000000000000E+0002
733  1984.TXHIPLAINS              4.19700000000000E+0003
734  1984.TXROLINGPL              1.43700000000000E+0003
735  1984.TXSOUTH                 8.74000000000000E+0002
736  1984.TXTRANSPEC              1.80000000000000E+0001
737  1985.TXCNTBLACK              2.41400000000000E+0003
738  1985.TXEAST                  2.17000000000000E+0002
739  1985.TXEDPLAT                2.44000000000000E+0002
740  1985.TXHIPLAINS              4.77900000000000E+0003
741  1985.TXROLINGPL              1.97300000000000E+0003
742  1985.TXSOUTH                 9.00000000000000E+0002
743  1985.TXTRANSPEC              1.40000000000000E+0001
744  1986.TXCNTBLACK              2.03800000000000E+0003
745  1986.TXEAST                  1.79000000000000E+0002
746  1986.TXEDPLAT                1.67000000000000E+0002
747  1986.TXHIPLAINS              4.33100000000000E+0003
748  1986.TXROLINGPL              1.65400000000000E+0003
749  1986.TXSOUTH                 7.06000000000000E+0002
750  1986.TXTRANSPEC              1.50000000000000E+0001
751  1987.TXCNTBLACK              1.57300000000000E+0003
752  1987.TXEAST                  1.30000000000000E+0002
753  1987.TXEDPLAT                1.58000000000000E+0002
754  1987.TXHIPLAINS              3.07700000000000E+0003
755  1987.TXROLINGPL              1.30700000000000E+0003
756  1987.TXSOUTH                 5.94000000000000E+0002
757  1987.TXTRANSPEC              1.10000000000000E+0001
758  1988.TXCNTBLACK              1.54500000000000E+0003
759  1988.TXEAST                  1.33000000000000E+0002
760  1988.TXEDPLAT                1.17000000000000E+0002
761  1988.TXHIPLAINS              2.62400000000000E+0003
762  1988.TXROLINGPL              1.05600000000000E+0003
763  1988.TXSOUTH                 5.58000000000000E+0002
764  1988.TXTRANSPEC              1.20000000000000E+0001
765  1989.TXCNTBLACK              1.83500000000000E+0003
766  1989.TXEAST                  1.63000000000000E+0002
767  1989.TXEDPLAT                1.87000000000000E+0002
768  1989.TXHIPLAINS              2.72600000000000E+0003
769  1989.TXROLINGPL              1.36000000000000E+0003
770  1989.TXSOUTH                 5.61000000000000E+0002
771  1989.TXTRANSPEC              4.00000000000000E+0000
772  1990.TXCNTBLACK              1.88500000000000E+0003
773  1990.TXEAST                  1.44000000000000E+0002
774  1990.TXEDPLAT                1.93000000000000E+0002
775  1990.TXHIPLAINS              3.04700000000000E+0003
776  1990.TXROLINGPL              1.48700000000000E+0003
777  1990.TXSOUTH                 6.82000000000000E+0002
778  1991.TXCNTBLACK              1.68900000000000E+0003
779  1991.TXEAST                  1.25000000000000E+0002
780  1991.TXEDPLAT                1.58000000000000E+0002
781  1991.TXHIPLAINS              2.56100000000000E+0003
```

```
 782  1991.TXROLINGPL               1.05100000000000E+0003
 783  1991.TXSOUTH                  7.05000000000000E+0002
 784  1991.TXTRANSPEC               1.00000000000000E+0000
 785  1992.TXCNTBLACK               2.36300000000000E+0003
 786  1992.TXEAST                   1.92000000000000E+0002
 787  1992.TXEDPLAT                 2.29000000000000E+0002
 788  1992.TXHIPLAINS               3.46700000000000E+0003
 789  1992.TXROLINGPL               1.45100000000000E+0003
 790  1992.TXSOUTH                  9.72000000000000E+0002
 791  1992.TXTRANSPEC               2.00000000000000E+0000
 792  /;


 793  option subacre:0:1:1;display subacre;
 794
 795  parameter totacre(year)   total texas acres;
LIBINCLUDE C:\GAMS25\SSIMPORT.GMS
 846  parameter totacre /
INCLUDE    C:\GAMS25\225A\GAMSLINK.SCR
 848  *  Tuesday, June 30 1998   15:28:20
 849  *  C:\GAMS25\IM1.WK1  TOTACRE
 850  1971                     7.21300000000000E+0003
 851  1972                     7.29800000000000E+0003
 852  1973                     1.02380000000000E+0004
 853  1974                     9.20300000000000E+0003
 854  1975                     1.30680000000000E+0004
 855  1976                     1.09010000000000E+0004
 856  1977                     1.03810000000000E+0004
 857  1978                     8.09900000000000E+0003
 858  1979                     9.72100000000000E+0003
 859  1980                     9.57900000000000E+0003
 860  1981                     1.09320000000000E+0004
 861  1982                     1.19200000000000E+0004
 862  1983                     8.20300000000000E+0003
 863  1984                     9.34100000000000E+0003
 864  1985                     1.05410000000000E+0004
 865  1986                     9.09000000000000E+0003
 866  1987                     6.85000000000000E+0003
 867  1988                     6.04500000000000E+0003
 868  1989                     6.83600000000000E+0003
 869  1990                     7.43800000000000E+0003
 870  1991                     6.29000000000000E+0003
 871  1992                     8.67600000000000E+0003
 872  /;
 873  option totacre:0:0:1;display totacre;

----    581 PARAMETER TXACRE        Texas historical acreage by crops

                        corn    soybeans       wheat     sorghum

1971.txhiplains          244          51        1029        2926
1971.txrolingpl                                317         521
1971.txcntblack          163           9         124         934
1971.txeast               34           9                      67
1971.txedplat              2                       8         106
1971.txsouth              64           2           8         580
1971.txtranspec                                    7           8
1972.txhiplains          199          58        1185        2486
1972.txrolingpl                                609         518
1972.txcntblack          142          35         142         988
1972.txeast               30          29           1          59
1972.txedplat              1                      28         100
1972.txsouth              45           3          23         601
1972.txtranspec                                    8           8
1973.txhiplains          367         131        2186        3030
1973.txrolingpl                                817         645
1973.txcntblack          133          29         268        1261
```

1973.txeast	27	59	4	43
1973.txedplat			64	137
1973.txsouth	72	7	52	893
1973.txtranspec			5	8
1974.txhiplains	541	88	1665	2175
1974.txrolingpl	1		1090	400
1974.txcntblack	125	24	393	1207
1974.txeast	34	34	6	74
1974.txedplat	1		72	131
1974.txsouth	61	4	66	997
1974.txtranspec			5	9
1975.txhiplains	771	127	3060	2629
1975.txrolingpl			1840	617
1975.txcntblack	139	43	519	1498
1975.txeast	41	40	3	60
1975.txedplat	2		154	138
1975.txsouth	91	5	92	1171
1975.txtranspec			26	2
1976.txhiplains	1210	75	2125	1890
1976.txrolingpl	4		1665	479
1976.txcntblack	132	48	690	1270
1976.txeast	21	53	17	48
1976.txedplat	2		74	149
1976.txsouth	71		81	757
1976.txtranspec			40	
1977.txhiplains	1211	152	2337	1320
1977.txrolingpl	3		1580	335
1977.txcntblack	165	74	555	1328
1977.txeast	37	103	19	30
1977.txedplat	5		124	121
1977.txsouth	119	8	62	666
1977.txtranspec			19	8
1978.txhiplains	907	188	1285	1370
1978.txrolingpl	2		834	280
1978.txcntblack	199	51	474	1351
1978.txeast	33	103	18	31
1978.txedplat	3		46	88
1978.txsouth	144	6	33	640
1978.txtranspec			5	8
1979.txhiplains	700	217	2393	1230
1979.txrolingpl	3		1370	290
1979.txcntblack	209	69	620	1188
1979.txeast	37	110	17	24
1979.txedplat	3		115	121
1979.txsouth	173	16	72	731
1979.txtranspec			5	8
1980.txhiplains	735	65	2595	1137
1980.txrolingpl	3		1255	168
1980.txcntblack	226	37	1116	983
1980.txeast	28	122	62	27
1980.txedplat	2		85	79
1980.txsouth	154	5	46	626
1980.txtranspec			23	
1981.txhiplains	522	58	2750	1239
1981.txrolingpl	1		1864	195
1981.txcntblack	212	22	1511	894
1981.txeast	31	73	142	41
1981.txedplat	1		145	88
1981.txsouth	187		87	822
1981.txtranspec			38	9
1982.txhiplains	523	445	2520	2180
1982.txrolingpl	2		1605	357
1982.txcntblack	237	55	1437	1062
1982.txeast	25	58	102	44
1982.txedplat	2		178	109
1982.txsouth	167	2	108	682
1982.txtranspec			18	2

1983.txhiplains	400	108	1985	940
1983.txrolingpl	1		1275	237
1983.txcntblack	266	49	1069	657
1983.txeast	23	34	84	30
1983.txedplat	1		97	89
1983.txsouth	203		51	589
1983.txtranspec			15	
1984.txhiplains	532	40	2330	1295
1984.txrolingpl	3		1260	174
1984.txcntblack	415	32	1172	862
1984.txeast	35	44	66	26
1984.txedplat	1		90	72
1984.txsouth	220		45	609
1984.txtranspec			10	8
1985.txhiplains	517	75	2620	1567
1985.txrolingpl	6		1680	287
1985.txcntblack	361	17	1175	861
1985.txeast	43	20	129	25
1985.txedplat	28	1	128	87
1985.txsouth	216		93	591
1985.txtranspec			6	8
1986.txhiplains	467	56	2340	1468
1986.txrolingpl	5		1400	249
1986.txcntblack	403	26	845	764
1986.txeast	56	32	40	51
1986.txedplat	6		82	79
1986.txsouth	163		67	476
1986.txtranspec			7	8
1987.txhiplains	412	40	1730	895
1987.txrolingpl	3		1113	191
1987.txcntblack	383	31	552	607
1987.txeast	30	21	43	36
1987.txedplat	3		87	68
1987.txsouth	170		63	361
1987.txtranspec			3	8
1988.txhiplains	423	61	1472	668
1988.txrolingpl	3		930	123
1988.txcntblack	392	25	629	499
1988.txeast	36	31	45	21
1988.txedplat	3		58	56
1988.txsouth	222		54	282
1988.txtranspec			3	9
1989.txhiplains	592	178	748	1208
1989.txrolingpl	3		1225	132
1989.txcntblack	369	49	774	643
1989.txeast	36	42	55	30
1989.txedplat	3		121	63
1989.txsouth	182	5	66	308
1989.txtranspec			2	2
1990.txhiplains	621	28	1736	662
1990.txrolingpl	7		1354	126
1990.txcntblack	363	44	833	645
1990.txeast	34	34	53	23
1990.txedplat	3		127	63
1990.txsouth	230	2	83	367
1991.txhiplains	642	24	1157	738
1991.txrolingpl	7		903	141
1991.txcntblack	375	38	556	720
1991.txeast	35	29	35	26
1991.txedplat	3		85	70
1991.txsouth	238	2	56	409
1991.txtranspec				1
1992.txhiplains	694	56	1571	1146
1992.txrolingpl	8		1225	218
1992.txcntblack	405	87	754	1117
1992.txeast	38	66	48	40
1992.txedplat	3	2	115	109

```
1992.txsouth            257           4          76         635
1992.txtranspec                                                 2


----    793 PARAMETER SUBACRE       Texas subreg acres

      txhiplains  txrolingpl  txcntblack      txeast    txedplat     txsouth

1971        4250         838        1230         110         116         654
1972        3928        1127        1307         119         129         672
1973        5714        1462        1691         133         201        1024
1974        4469        1491        1749         148         204        1128
1975        6587        2457        2199         144         294        1359
1976        5300        2148        2140         139         225         909
1977        5020        1918        2122         189         250         855
1978        3750        1116        2075         185         137         823
1979        4540        1663        2086         188         239         992
1980        4532        1426        2362         239         166         831
1981        4569        2060        2639         287         234        1096
1982        5668        1964        2791         229         289         959
1983        3433        1513        2041         171         187         843
1984        4197        1437        2481         171         163         874
1985        4779        1973        2414         217         244         900
1986        4331        1654        2038         179         167         706
1987        3077        1307        1573         130         158         594
1988        2624        1056        1545         133         117         558
1989        2726        1360        1835         163         187         561
1990        3047        1487        1885         144         193         682
1991        2561        1051        1689         125         158         705
1992        3467        1451        2363         192         229         972

   +  txtranspec

1971          15
1972          16
1973          13
1974          14
1975          28
1976          40
1977          27
1978          13
1979          13
1980          23
1981          47
1982          20
1983          15
1984          18
1985          14
1986          15
1987          11
1988          12
1989           4
1991           1
1992           2


----    873 PARAMETER TOTACRE       total texas acres

1971  7213
1972  7298
1973 10238
1974  9203
1975 13068
1976 10901
1977 10381
1978  8099
1979  9721
```

```
1980  9579
1981 10932
1982 11920
1983  8203
1984  9341
1985 10541
1986  9090
1987  6850
1988  6045
1989  6836
1990  7438
1991  6290
1992  8676
```

```
##############################
*Examples for Readable data file
*##############################
set year  /1971*1992/
   crop  /corn, soybeans, wheat, sorghum/
   subreg  /txhiplains, txrolingpl, txcntblack, txeast, txedplat,
            txsouth, txtranspec/;

Table TXACRE(year,subreg,crop)  Texas historical acreage by crops

                       corn    soybeans       wheat     sorghum

1971.TXHIPLAINS         244          51        1029        2926
1971.TXROLINGPL                                317         521
1971.TXCNTBLACK         163           9         124         934
1971.TXEAST              34           9                      67
1971.TXEDPLAT             2                       8         106
1971.TXSOUTH             64           2           8         580
1971.TXTRANSPEC                                    7           8
1972.TXHIPLAINS         199          58        1185        2486
1972.TXROLINGPL                                609         518
1972.TXCNTBLACK         142          35         142         988
1972.TXEAST              30          29           1          59
1972.TXEDPLAT             1                      28         100
1972.TXSOUTH             45           3          23         601
1972.TXTRANSPEC                                    8           8
1973.TXHIPLAINS         367         131        2186        3030
1973.TXROLINGPL                                817         645
1973.TXCNTBLACK         133          29         268        1261
1973.TXEAST              27          59           4          43
1973.TXEDPLAT                                     64         137
1973.TXSOUTH             72           7          52         893
1973.TXTRANSPEC                                    5           8
1974.TXHIPLAINS         541          88        1665        2175
1974.TXROLINGPL           1                     1090         400
1974.TXCNTBLACK         125          24         393        1207
1974.TXEAST              34          34           6          74
1974.TXEDPLAT             1                      72         131
1974.TXSOUTH             61           4          66         997
1974.TXTRANSPEC                                    5           9
1975.TXHIPLAINS         771         127        3060        2629
1975.TXROLINGPL                                1840         617
1975.TXCNTBLACK         139          43         519        1498
1975.TXEAST              41          40           3          60
1975.TXEDPLAT             2                     154         138
1975.TXSOUTH             91           5          92        1171
1975.TXTRANSPEC                                   26           2
1976.TXHIPLAINS        1210          75        2125        1890
1976.TXROLINGPL           4                     1665         479
1976.TXCNTBLACK         132          48         690        1270
1976.TXEAST              21          53          17          48
1976.TXEDPLAT             2                      74         149
1976.TXSOUTH             71                      81         757
1976.TXTRANSPEC                                   40
1977.TXHIPLAINS        1211         152        2337        1320
1977.TXROLINGPL           3                     1580         335
1977.TXCNTBLACK         165          74         555        1328
1977.TXEAST              37         103          19          30
1977.TXEDPLAT             5                     124         121
1977.TXSOUTH             119           8          62         666
1977.TXTRANSPEC                                   19           8
1978.TXHIPLAINS          907         188        1285        1370
1978.TXROLINGPL            2                     834         280
```

1978.TXCNTBLACK	199	51	474	1351
1978.TXEAST	33	103	18	31
1978.TXEDPLAT	3		46	88
1978.TXSOUTH	144	6	33	640
1978.TXTRANSPEC			5	8
1979.TXHIPLAINS	700	217	2393	1230
1979.TXROLINGPL	3		1370	290
1979.TXCNTBLACK	209	69	620	1188
1979.TXEAST	37	110	17	24
1979.TXEDPLAT	3		115	121
1979.TXSOUTH	173	16	72	731
1979.TXTRANSPEC			5	8
1980.TXHIPLAINS	735	65	2595	1137
1980.TXROLINGPL	3		1255	168
1980.TXCNTBLACK	226	37	1116	983
1980.TXEAST	28	122	62	27
1980.TXEDPLAT	2		85	79
1980.TXSOUTH	154	5	46	626
1980.TXTRANSPEC			23	
1981.TXHIPLAINS	522	58	2750	1239
1981.TXROLINGPL	1		1864	195
1981.TXCNTBLACK	212	22	1511	894
1981.TXEAST	31	73	142	41
1981.TXEDPLAT	1		145	88
1981.TXSOUTH	187		87	822
1981.TXTRANSPEC			38	9
1982.TXHIPLAINS	523	445	2520	2180
1982.TXROLINGPL	2		1605	357
1982.TXCNTBLACK	237	55	1437	1062
1982.TXEAST	25	58	102	44
1982.TXEDPLAT	2		178	109
1982.TXSOUTH	167	2	108	682
1982.TXTRANSPEC			18	2
1983.TXHIPLAINS	400	108	1985	940
1983.TXROLINGPL	1		1275	237
1983.TXCNTBLACK	266	49	1069	657
1983.TXEAST	23	34	84	30
1983.TXEDPLAT	1		97	89
1983.TXSOUTH	203		51	589
1983.TXTRANSPEC			15	
1984.TXHIPLAINS	532	40	2330	1295
1984.TXROLINGPL	3		1260	174
1984.TXCNTBLACK	415	32	1172	862
1984.TXEAST	35	44	66	26
1984.TXEDPLAT	1		90	72
1984.TXSOUTH	220		45	609
1984.TXTRANSPEC			10	8
1985.TXHIPLAINS	517	75	2620	1567
1985.TXROLINGPL	6		1680	287
1985.TXCNTBLACK	361	17	1175	861
1985.TXEAST	43	20	129	25
1985.TXEDPLAT	28	1	128	87
1985.TXSOUTH	216		93	591
1985.TXTRANSPEC			6	8
1986.TXHIPLAINS	467	56	2340	1468
1986.TXROLINGPL	5		1400	249
1986.TXCNTBLACK	403	26	845	764
1986.TXEAST	56	32	40	51
1986.TXEDPLAT	6		82	79
1986.TXSOUTH	163		67	476
1986.TXTRANSPEC			7	8
1987.TXHIPLAINS	412	40	1730	895
1987.TXROLINGPL	3		1113	191
1987.TXCNTBLACK	383	31	552	607
1987.TXEAST	30	21	43	36
1987.TXEDPLAT	3		87	68
1987.TXSOUTH	170		63	361

```
1987.TXTRANSPEC                                    3          8
1988.TXHIPLAINS          423          61       1472        668
1988.TXROLINGPL            3                     930        123
1988.TXCNTBLACK          392          25        629        499
1988.TXEAST               36          31         45         21
1988.TXEDPLAT              3                      58         56
1988.TXSOUTH             222                      54        282
1988.TXTRANSPEC                                    3          9
1989.TXHIPLAINS          592         178        748       1208
1989.TXROLINGPL            3                    1225        132
1989.TXCNTBLACK          369          49        774        643
1989.TXEAST               36          42         55         30
1989.TXEDPLAT              3                     121         63
1989.TXSOUTH             182           5         66        308
1989.TXTRANSPEC                                    2          2
1990.TXHIPLAINS          621          28       1736        662
1990.TXROLINGPL            7                    1354        126
1990.TXCNTBLACK          363          44        833        645
1990.TXEAST               34          34         53         23
1990.TXEDPLAT              3                     127         63
1990.TXSOUTH             230           2         83        367
1991.TXHIPLAINS          642          24       1157        738
1991.TXROLINGPL            7                     903        141
1991.TXCNTBLACK          375          38        556        720
1991.TXEAST               35          29         35         26
1991.TXEDPLAT              3                      85         70
1991.TXSOUTH             238           2         56        409
1991.TXTRANSPEC                                                1
1992.TXHIPLAINS          694          56       1571       1146
1992.TXROLINGPL            8                    1225        218
1992.TXCNTBLACK          405          87        754       1117
1992.TXEAST               38          66         48         40
1992.TXEDPLAT              3           2        115        109
1992.TXSOUTH             257           4         76        635
1992.TXTRANSPEC                                                2
;
```

```
parameter subacre(year,subreg)  subreg acres ;
subacre(year,subreg)
  =sum(crop, txacre(year,subreg,crop));

parameter totacre(year)     total texas acres ;
totacre(year)
  =sum(subreg, subacre(year,subreg));

file result1 /totacre.gms/;
put result1;

$libinclude c:\gams25\gams2prm totacre

file result2 /subacre.gms/;
put result2;

$libinclude c:\gams25\gams2txt subacre

file result3 /txacre.gms/;
put result3;
```

```
$libinclude c:\gams25\gams2txt txacre
```

*The Readable List files
1)Totacre.gms file
parameter totacre total texas acres/
*=>gams2prm totacre
* Called from C:\GAMS25\EXAM1, line 236
* 06/30/98 16:09:44
1971 7.21300000000000E+03
1972 7.29800000000000E+03
1973 1.02380000000000E+04
1974 9.20300000000000E+03
1975 1.30680000000000E+04
1976 1.09010000000000E+04
1977 1.03810000000000E+04
1978 8.09900000000000E+03
1979 9.72100000000000E+03
1980 9.57900000000000E+03
1981 1.09320000000000E+04
1982 1.19200000000000E+04
1983 8.20300000000000E+03
1984 9.34100000000000E+03
1985 1.05410000000000E+04
1986 9.09000000000000E+03
1987 6.85000000000000E+03
1988 6.04500000000000E+03
1989 6.83600000000000E+03
1990 7.43800000000000E+03
1991 6.29000000000000E+03
1992 8.67600000000000E+03
/;

2)Subacre.gms file
```
*=>gams2txt subacre
* Called from C:\GAMS25\EXAM1, line 465
* 06/30/98 16:09:44
1971.txhiplains  4.25000000000000E+03
1971.txrolingpl  8.38000000000000E+02
1971.txcntblack  1.23000000000000E+03
1971.txeast  1.10000000000000E+02
1971.txedplat  1.16000000000000E+02
1971.txsouth  6.54000000000000E+02
1971.txtranspec  1.50000000000000E+01
1972.txhiplains  3.92800000000000E+03
1972.txrolingpl  1.12700000000000E+03
1972.txcntblack  1.30700000000000E+03
1972.txeast  1.19000000000000E+02
1972.txedplat  1.29000000000000E+02
1972.txsouth  6.72000000000000E+02
1972.txtranspec  1.60000000000000E+01
1973.txhiplains  5.71400000000000E+03
1973.txrolingpl  1.46200000000000E+03
```

```
1973.txcntblack  1.69100000000000E+03
1973.txeast  1.33000000000000E+02
1973.txedplat  2.01000000000000E+02
1973.txsouth  1.02400000000000E+03
1973.txtranspec  1.30000000000000E+01
1974.txhiplains  4.46900000000000E+03
1974.txrolingpl  1.49100000000000E+03
1974.txcntblack  1.74900000000000E+03
1974.txeast  1.48000000000000E+02
1974.txedplat  2.04000000000000E+02
1974.txsouth  1.12800000000000E+03
1974.txtranspec  1.40000000000000E+01
1975.txhiplains  6.58700000000000E+03
1975.txrolingpl  2.45700000000000E+03
1975.txcntblack  2.19900000000000E+03
1975.txeast  1.44000000000000E+02
1975.txedplat  2.94000000000000E+02
1975.txsouth  1.35900000000000E+03
1975.txtranspec  2.80000000000000E+01
1976.txhiplains  5.30000000000000E+03
1976.txrolingpl  2.14800000000000E+03
1976.txcntblack  2.14000000000000E+03
1976.txeast  1.39000000000000E+02
1976.txedplat  2.25000000000000E+02
1976.txsouth  9.09000000000000E+02
1976.txtranspec  4.00000000000000E+01
1977.txhiplains  5.02000000000000E+03
1977.txrolingpl  1.91800000000000E+03
1977.txcntblack  2.12200000000000E+03
1977.txeast  1.89000000000000E+02
1977.txedplat  2.50000000000000E+02
1977.txsouth  8.55000000000000E+02
1977.txtranspec  2.70000000000000E+01
1978.txhiplains  3.75000000000000E+03
1978.txrolingpl  1.11600000000000E+03
1978.txcntblack  2.07500000000000E+03
1978.txeast  1.85000000000000E+02
1978.txedplat  1.37000000000000E+02
1978.txsouth  8.23000000000000E+02
1978.txtranspec  1.30000000000000E+01
1979.txhiplains  4.54000000000000E+03
1979.txrolingpl  1.66300000000000E+03
1979.txcntblack  2.08600000000000E+03
1979.txeast  1.88000000000000E+02
1979.txedplat  2.39000000000000E+02
1979.txsouth  9.92000000000000E+02
1979.txtranspec  1.30000000000000E+01
1980.txhiplains  4.53200000000000E+03
1980.txrolingpl  1.42600000000000E+03
1980.txcntblack  2.36200000000000E+03
1980.txeast  2.39000000000000E+02
1980.txedplat  1.66000000000000E+02
1980.txsouth  8.31000000000000E+02
1980.txtranspec  2.30000000000000E+01
1981.txhiplains  4.56900000000000E+03
1981.txrolingpl  2.06000000000000E+03
1981.txcntblack  2.63900000000000E+03
1981.txeast  2.87000000000000E+02
1981.txedplat  2.34000000000000E+02
1981.txsouth  1.09600000000000E+03
1981.txtranspec  4.70000000000000E+01
1982.txhiplains  5.66800000000000E+03
1982.txrolingpl  1.96400000000000E+03
1982.txcntblack  2.79100000000000E+03
1982.txeast  2.29000000000000E+02
1982.txedplat  2.89000000000000E+02
1982.txsouth  9.59000000000000E+02
```

```
1982.txtranspec  2.00000000000000E+01
1983.txhiplains  3.43300000000000E+03
1983.txrolingpl  1.51300000000000E+03
1983.txcntblack  2.04100000000000E+03
1983.txeast  1.71000000000000E+02
1983.txedplat  1.87000000000000E+02
1983.txsouth  8.43000000000000E+02
1983.txtranspec  1.50000000000000E+01
1984.txhiplains  4.19700000000000E+03
1984.txrolingpl  1.43700000000000E+03
1984.txcntblack  2.48100000000000E+03
1984.txeast  1.71000000000000E+02
1984.txedplat  1.63000000000000E+02
1984.txsouth  8.74000000000000E+02
1984.txtranspec  1.80000000000000E+01
1985.txhiplains  4.77900000000000E+03
1985.txrolingpl  1.97300000000000E+03
1985.txcntblack  2.41400000000000E+03
1985.txeast  2.17000000000000E+02
1985.txedplat  2.44000000000000E+02
1985.txsouth  9.00000000000000E+02
1985.txtranspec  1.40000000000000E+01
1986.txhiplains  4.33100000000000E+03
1986.txrolingpl  1.65400000000000E+03
1986.txcntblack  2.03800000000000E+03
1986.txeast  1.79000000000000E+02
1986.txedplat  1.67000000000000E+02
1986.txsouth  7.06000000000000E+02
1986.txtranspec  1.50000000000000E+01
1987.txhiplains  3.07700000000000E+03
1987.txrolingpl  1.30700000000000E+03
1987.txcntblack  1.57300000000000E+03
1987.txeast  1.30000000000000E+02
1987.txedplat  1.58000000000000E+02
1987.txsouth  5.94000000000000E+02
1987.txtranspec  1.10000000000000E+01
1988.txhiplains  2.62400000000000E+03
1988.txrolingpl  1.05600000000000E+03
1988.txcntblack  1.54500000000000E+03
1988.txeast  1.33000000000000E+02
1988.txedplat  1.17000000000000E+02
1988.txsouth  5.58000000000000E+02
1988.txtranspec  1.20000000000000E+01
1989.txhiplains  2.72600000000000E+03
1989.txrolingpl  1.36000000000000E+03
1989.txcntblack  1.83500000000000E+03
1989.txeast  1.63000000000000E+02
1989.txedplat  1.87000000000000E+02
1989.txsouth  5.61000000000000E+02
1989.txtranspec  4.00000000000000E+00
1990.txhiplains  3.04700000000000E+03
1990.txrolingpl  1.48700000000000E+03
1990.txcntblack  1.88500000000000E+03
1990.txeast  1.44000000000000E+02
1990.txedplat  1.93000000000000E+02
1990.txsouth  6.82000000000000E+02
1991.txhiplains  2.56100000000000E+03
1991.txrolingpl  1.05100000000000E+03
1991.txcntblack  1.68900000000000E+03
1991.txeast  1.25000000000000E+02
1991.txedplat  1.58000000000000E+02
1991.txsouth  7.05000000000000E+02
1991.txtranspec  1.00000000000000E+00
1992.txhiplains  3.46700000000000E+03
1992.txrolingpl  1.45100000000000E+03
1992.txcntblack  2.36300000000000E+03
1992.txeast  1.92000000000000E+02
```

```
1992.txedplat  2.29000000000000E+02
1992.txsouth  9.72000000000000E+02
1992.txtranspec  2.00000000000000E+00
```

3)Txacre.gms file

```
*=>gams2txt txacre
* Called from C:\GAMS25\EXAM1, line 628
* 06/30/98 16:09:44
1971.txhiplains.corn  2.44000000000000E+02
1971.txhiplains.soybeans  5.10000000000000E+01
1971.txhiplains.wheat  1.02900000000000E+03
1971.txhiplains.sorghum  2.92600000000000E+03
1971.txrolingpl.wheat  3.17000000000000E+02
1971.txrolingpl.sorghum  5.21000000000000E+02
1971.txcntblack.corn  1.63000000000000E+02
1971.txcntblack.soybeans  9.00000000000000E+00
1971.txcntblack.wheat  1.24000000000000E+02
1971.txcntblack.sorghum  9.34000000000000E+02
1971.txeast.corn  3.40000000000000E+01
1971.txeast.soybeans  9.00000000000000E+00
1971.txeast.sorghum  6.70000000000000E+01
1971.txedplat.corn  2.00000000000000E+00
1971.txedplat.wheat  8.00000000000000E+00
1971.txedplat.sorghum  1.06000000000000E+02
1971.txsouth.corn  6.40000000000000E+01
1971.txsouth.soybeans  2.00000000000000E+00
1971.txsouth.wheat  8.00000000000000E+00
1971.txsouth.sorghum  5.80000000000000E+02
1971.txtranspec.wheat  7.00000000000000E+00
1971.txtranspec.sorghum  8.00000000000000E+00
1972.txhiplains.corn  1.99000000000000E+02
1972.txhiplains.soybeans  5.80000000000000E+01
1972.txhiplains.wheat  1.18500000000000E+03
1972.txhiplains.sorghum  2.48600000000000E+03
1972.txrolingpl.wheat  6.09000000000000E+02
1972.txrolingpl.sorghum  5.18000000000000E+02
1972.txcntblack.corn  1.42000000000000E+02
1972.txcntblack.soybeans  3.50000000000000E+01
1972.txcntblack.wheat  1.42000000000000E+02
1972.txcntblack.sorghum  9.88000000000000E+02
1972.txeast.corn  3.00000000000000E+01
1972.txeast.soybeans  2.90000000000000E+01
1972.txeast.wheat  1.00000000000000E+00
1972.txeast.sorghum  5.90000000000000E+01
1972.txedplat.corn  1.00000000000000E+00
1972.txedplat.wheat  2.80000000000000E+01
1972.txedplat.sorghum  1.00000000000000E+02
1972.txsouth.corn  4.50000000000000E+01
1972.txsouth.soybeans  3.00000000000000E+00
1972.txsouth.wheat  2.30000000000000E+01
1972.txsouth.sorghum  6.01000000000000E+02
1972.txtranspec.wheat  8.00000000000000E+00
1972.txtranspec.sorghum  8.00000000000000E+00
1973.txhiplains.corn  3.67000000000000E+02
1973.txhiplains.soybeans  1.31000000000000E+02
1973.txhiplains.wheat  2.18600000000000E+03
1973.txhiplains.sorghum  3.03000000000000E+03
1973.txrolingpl.wheat  8.17000000000000E+02
1973.txrolingpl.sorghum  6.45000000000000E+02
1973.txcntblack.corn  1.33000000000000E+02
1973.txcntblack.soybeans  2.90000000000000E+01
1973.txcntblack.wheat  2.68000000000000E+02
1973.txcntblack.sorghum  1.26100000000000E+03
1973.txeast.corn  2.70000000000000E+01
```

```
1973.txeast.soybeans  5.90000000000000E+01
1973.txeast.wheat  4.00000000000000E+00
1973.txeast.sorghum  4.30000000000000E+01
1973.txedplat.wheat  6.40000000000000E+01
1973.txedplat.sorghum  1.37000000000000E+02
1973.txsouth.corn  7.20000000000000E+01
1973.txsouth.soybeans  7.00000000000000E+00
1973.txsouth.wheat  5.20000000000000E+01
1973.txsouth.sorghum  8.93000000000000E+02
1973.txtranspec.wheat  5.00000000000000E+00
1973.txtranspec.sorghum  8.00000000000000E+00
1974.txhiplains.corn  5.41000000000000E+02
1974.txhiplains.soybeans  8.80000000000000E+01
1974.txhiplains.wheat  1.66500000000000E+03
1974.txhiplains.sorghum  2.17500000000000E+03
1974.txrolingpl.corn  1.00000000000000E+00
1974.txrolingpl.wheat  1.09000000000000E+03
1974.txrolingpl.sorghum  4.00000000000000E+02
1974.txcntblack.corn  1.25000000000000E+02
1974.txcntblack.soybeans  2.40000000000000E+01
1974.txcntblack.wheat  3.93000000000000E+02
1974.txcntblack.sorghum  1.20700000000000E+03
1974.txeast.corn  3.40000000000000E+01
1974.txeast.soybeans  3.40000000000000E+01
1974.txeast.wheat  6.00000000000000E+00
1974.txeast.sorghum  7.40000000000000E+01
1974.txedplat.corn  1.00000000000000E+00
1974.txedplat.wheat  7.20000000000000E+01
1974.txedplat.sorghum  1.31000000000000E+02
1974.txsouth.corn  6.10000000000000E+01
1974.txsouth.soybeans  4.00000000000000E+00
1974.txsouth.wheat  6.60000000000000E+01
1974.txsouth.sorghum  9.97000000000000E+02
1974.txtranspec.wheat  5.00000000000000E+00
1974.txtranspec.sorghum  9.00000000000000E+00
1975.txhiplains.corn  7.71000000000000E+02
1975.txhiplains.soybeans  1.27000000000000E+02
1975.txhiplains.wheat  3.06000000000000E+03
1975.txhiplains.sorghum  2.62900000000000E+03
1975.txrolingpl.wheat  1.84000000000000E+03
1975.txrolingpl.sorghum  6.17000000000000E+02
1975.txcntblack.corn  1.39000000000000E+02
1975.txcntblack.soybeans  4.30000000000000E+01
1975.txcntblack.wheat  5.19000000000000E+02
1975.txcntblack.sorghum  1.49800000000000E+03
1975.txeast.corn  4.10000000000000E+01
1975.txeast.soybeans  4.00000000000000E+01
1975.txeast.wheat  3.00000000000000E+00
1975.txeast.sorghum  6.00000000000000E+01
1975.txedplat.corn  2.00000000000000E+00
1975.txedplat.wheat  1.54000000000000E+02
1975.txedplat.sorghum  1.38000000000000E+02
1975.txsouth.corn  9.10000000000000E+01
1975.txsouth.soybeans  5.00000000000000E+00
1975.txsouth.wheat  9.20000000000000E+01
1975.txsouth.sorghum  1.17100000000000E+03
1975.txtranspec.wheat  2.60000000000000E+01
1975.txtranspec.sorghum  2.00000000000000E+00
1976.txhiplains.corn  1.21000000000000E+03
1976.txhiplains.soybeans  7.50000000000000E+01
1976.txhiplains.wheat  2.12500000000000E+03
1976.txhiplains.sorghum  1.89000000000000E+03
1976.txrolingpl.corn  4.00000000000000E+00
1976.txrolingpl.wheat  1.66500000000000E+03
1976.txrolingpl.sorghum  4.79000000000000E+02
1976.txcntblack.corn  1.32000000000000E+02
1976.txcntblack.soybeans  4.80000000000000E+01
```

```
1976.txcntblack.wheat  6.90000000000000E+02
1976.txcntblack.sorghum  1.27000000000000E+03
1976.txeast.corn  2.10000000000000E+01
1976.txeast.soybeans  5.30000000000000E+01
1976.txeast.wheat  1.70000000000000E+01
1976.txeast.sorghum  4.80000000000000E+01
1976.txedplat.corn  2.00000000000000E+00
1976.txedplat.wheat  7.40000000000000E+01
1976.txedplat.sorghum  1.49000000000000E+02
1976.txsouth.corn  7.10000000000000E+01
1976.txsouth.wheat  8.10000000000000E+01
1976.txsouth.sorghum  7.57000000000000E+02
1976.txtranspec.wheat  4.00000000000000E+01
1977.txhiplains.corn  1.21100000000000E+03
1977.txhiplains.soybeans  1.52000000000000E+02
1977.txhiplains.wheat  2.33700000000000E+03
1977.txhiplains.sorghum  1.32000000000000E+03
1977.txrolingpl.corn  3.00000000000000E+00
1977.txrolingpl.wheat  1.58000000000000E+03
1977.txrolingpl.sorghum  3.35000000000000E+02
1977.txcntblack.corn  1.65000000000000E+02
1977.txcntblack.soybeans  7.40000000000000E+01
1977.txcntblack.wheat  5.55000000000000E+02
1977.txcntblack.sorghum  1.32800000000000E+03
1977.txeast.corn  3.70000000000000E+01
1977.txeast.soybeans  1.03000000000000E+02
1977.txeast.wheat  1.90000000000000E+01
1977.txeast.sorghum  3.00000000000000E+01
1977.txedplat.corn  5.00000000000000E+00
1977.txedplat.wheat  1.24000000000000E+02
1977.txedplat.sorghum  1.21000000000000E+02
1977.txsouth.corn  1.19000000000000E+02
1977.txsouth.soybeans  8.00000000000000E+00
1977.txsouth.wheat  6.20000000000000E+01
1977.txsouth.sorghum  6.66000000000000E+02
1977.txtranspec.wheat  1.90000000000000E+01
1977.txtranspec.sorghum  8.00000000000000E+00
1978.txhiplains.corn  9.07000000000000E+02
1978.txhiplains.soybeans  1.88000000000000E+02
1978.txhiplains.wheat  1.28500000000000E+03
1978.txhiplains.sorghum  1.37000000000000E+03
1978.txrolingpl.corn  2.00000000000000E+00
1978.txrolingpl.wheat  8.34000000000000E+02
1978.txrolingpl.sorghum  2.80000000000000E+02
1978.txcntblack.corn  1.99000000000000E+02
1978.txcntblack.soybeans  5.10000000000000E+01
1978.txcntblack.wheat  4.74000000000000E+02
1978.txcntblack.sorghum  1.35100000000000E+03
1978.txeast.corn  3.30000000000000E+01
1978.txeast.soybeans  1.03000000000000E+02
1978.txeast.wheat  1.80000000000000E+01
1978.txeast.sorghum  3.10000000000000E+01
1978.txedplat.corn  3.00000000000000E+00
1978.txedplat.wheat  4.60000000000000E+01
1978.txedplat.sorghum  8.80000000000000E+01
1978.txsouth.corn  1.44000000000000E+02
1978.txsouth.soybeans  6.00000000000000E+00
1978.txsouth.wheat  3.30000000000000E+01
1978.txsouth.sorghum  6.40000000000000E+02
1978.txtranspec.wheat  5.00000000000000E+00
1978.txtranspec.sorghum  8.00000000000000E+00
1979.txhiplains.corn  7.00000000000000E+02
1979.txhiplains.soybeans  2.17000000000000E+02
1979.txhiplains.wheat  2.39300000000000E+03
1979.txhiplains.sorghum  1.23000000000000E+03
1979.txrolingpl.corn  3.00000000000000E+00
1979.txrolingpl.wheat  1.37000000000000E+03
```

1979.txrolingpl.sorghum 2.90000000000000E+02
1979.txcntblack.corn 2.09000000000000E+02
1979.txcntblack.soybeans 6.90000000000000E+01
1979.txcntblack.wheat 6.20000000000000E+02
1979.txcntblack.sorghum 1.18800000000000E+03
1979.txeast.corn 3.70000000000000E+01
1979.txeast.soybeans 1.10000000000000E+02
1979.txeast.wheat 1.70000000000000E+01
1979.txeast.sorghum 2.40000000000000E+01
1979.txedplat.corn 3.00000000000000E+00
1979.txedplat.wheat 1.15000000000000E+02
1979.txedplat.sorghum 1.21000000000000E+02
1979.txsouth.corn 1.73000000000000E+02
1979.txsouth.soybeans 1.60000000000000E+01
1979.txsouth.wheat 7.20000000000000E+01
1979.txsouth.sorghum 7.31000000000000E+02
1979.txtranspec.wheat 5.00000000000000E+00
1979.txtranspec.sorghum 8.00000000000000E+00
1980.txhiplains.corn 7.35000000000000E+02
1980.txhiplains.soybeans 6.50000000000000E+01
1980.txhiplains.wheat 2.59500000000000E+03
1980.txhiplains.sorghum 1.13700000000000E+03
1980.txrolingpl.corn 3.00000000000000E+00
1980.txrolingpl.wheat 1.25500000000000E+03
1980.txrolingpl.sorghum 1.68000000000000E+02
1980.txcntblack.corn 2.26000000000000E+02
1980.txcntblack.soybeans 3.70000000000000E+01
1980.txcntblack.wheat 1.11600000000000E+03
1980.txcntblack.sorghum 9.83000000000000E+02
1980.txeast.corn 2.80000000000000E+01
1980.txeast.soybeans 1.22000000000000E+02
1980.txeast.wheat 6.20000000000000E+01
1980.txeast.sorghum 2.70000000000000E+01
1980.txedplat.corn 2.00000000000000E+00
1980.txedplat.wheat 8.50000000000000E+01
1980.txedplat.sorghum 7.90000000000000E+01
1980.txsouth.corn 1.54000000000000E+02
1980.txsouth.soybeans 5.00000000000000E+00
1980.txsouth.wheat 4.60000000000000E+01
1980.txsouth.sorghum 6.26000000000000E+02
1980.txtranspec.wheat 2.30000000000000E+01
1981.txhiplains.corn 5.22000000000000E+02
1981.txhiplains.soybeans 5.80000000000000E+01
1981.txhiplains.wheat 2.75000000000000E+03
1981.txhiplains.sorghum 1.23900000000000E+03
1981.txrolingpl.corn 1.00000000000000E+00
1981.txrolingpl.wheat 1.86400000000000E+03
1981.txrolingpl.sorghum 1.95000000000000E+02
1981.txcntblack.corn 2.12000000000000E+02
1981.txcntblack.soybeans 2.20000000000000E+01
1981.txcntblack.wheat 1.51100000000000E+03
1981.txcntblack.sorghum 8.94000000000000E+02
1981.txeast.corn 3.10000000000000E+01
1981.txeast.soybeans 7.30000000000000E+01
1981.txeast.wheat 1.42000000000000E+02
1981.txeast.sorghum 4.10000000000000E+01
1981.txedplat.corn 1.00000000000000E+00
1981.txedplat.wheat 1.45000000000000E+02
1981.txedplat.sorghum 8.80000000000000E+01
1981.txsouth.corn 1.87000000000000E+02
1981.txsouth.wheat 8.70000000000000E+01
1981.txsouth.sorghum 8.22000000000000E+02
1981.txtranspec.wheat 3.80000000000000E+01
1981.txtranspec.sorghum 9.00000000000000E+00
1982.txhiplains.corn 5.23000000000000E+02
1982.txhiplains.soybeans 4.45000000000000E+02
1982.txhiplains.wheat 2.52000000000000E+03

1982.txhiplains.sorghum 2.18000000000000E+03
1982.txrolingpl.corn 2.00000000000000E+00
1982.txrolingpl.wheat 1.60500000000000E+03
1982.txrolingpl.sorghum 3.57000000000000E+02
1982.txcntblack.corn 2.37000000000000E+02
1982.txcntblack.soybeans 5.50000000000000E+01
1982.txcntblack.wheat 1.43700000000000E+03
1982.txcntblack.sorghum 1.06200000000000E+03
1982.txeast.corn 2.50000000000000E+01
1982.txeast.soybeans 5.80000000000000E+01
1982.txeast.wheat 1.02000000000000E+02
1982.txeast.sorghum 4.40000000000000E+01
1982.txedplat.corn 2.00000000000000E+00
1982.txedplat.wheat 1.78000000000000E+02
1982.txedplat.sorghum 1.09000000000000E+02
1982.txsouth.corn 1.67000000000000E+02
1982.txsouth.soybeans 2.00000000000000E+00
1982.txsouth.wheat 1.08000000000000E+02
1982.txsouth.sorghum 6.82000000000000E+02
1982.txtranspec.wheat 1.80000000000000E+01
1982.txtranspec.sorghum 2.00000000000000E+00
1983.txhiplains.corn 4.00000000000000E+02
1983.txhiplains.soybeans 1.08000000000000E+02
1983.txhiplains.wheat 1.98500000000000E+03
1983.txhiplains.sorghum 9.40000000000000E+02
1983.txrolingpl.corn 1.00000000000000E+00
1983.txrolingpl.wheat 1.27500000000000E+03
1983.txrolingpl.sorghum 2.37000000000000E+02
1983.txcntblack.corn 2.66000000000000E+02
1983.txcntblack.soybeans 4.90000000000000E+01
1983.txcntblack.wheat 1.06900000000000E+03
1983.txcntblack.sorghum 6.57000000000000E+02
1983.txeast.corn 2.30000000000000E+01
1983.txeast.soybeans 3.40000000000000E+01
1983.txeast.wheat 8.40000000000000E+01
1983.txeast.sorghum 3.00000000000000E+01
1983.txedplat.corn 1.00000000000000E+00
1983.txedplat.wheat 9.70000000000000E+01
1983.txedplat.sorghum 8.90000000000000E+01
1983.txsouth.corn 2.03000000000000E+02
1983.txsouth.wheat 5.10000000000000E+01
1983.txsouth.sorghum 5.89000000000000E+02
1983.txtranspec.wheat 1.50000000000000E+01
1984.txhiplains.corn 5.32000000000000E+02
1984.txhiplains.soybeans 4.00000000000000E+01
1984.txhiplains.wheat 2.33000000000000E+03
1984.txhiplains.sorghum 1.29500000000000E+03
1984.txrolingpl.corn 3.00000000000000E+00
1984.txrolingpl.wheat 1.26000000000000E+03
1984.txrolingpl.sorghum 1.74000000000000E+02
1984.txcntblack.corn 4.15000000000000E+02
1984.txcntblack.soybeans 3.20000000000000E+01
1984.txcntblack.wheat 1.17200000000000E+03
1984.txcntblack.sorghum 8.62000000000000E+02
1984.txeast.corn 3.50000000000000E+01
1984.txeast.soybeans 4.40000000000000E+01
1984.txeast.wheat 6.60000000000000E+01
1984.txeast.sorghum 2.60000000000000E+01
1984.txedplat.corn 1.00000000000000E+00
1984.txedplat.wheat 9.00000000000000E+01
1984.txedplat.sorghum 7.20000000000000E+01
1984.txsouth.corn 2.20000000000000E+02
1984.txsouth.wheat 4.50000000000000E+01
1984.txsouth.sorghum 6.09000000000000E+02
1984.txtranspec.wheat 1.00000000000000E+01
1984.txtranspec.sorghum 8.00000000000000E+00
1985.txhiplains.corn 5.17000000000000E+02

```
1985.txhiplains.soybeans  7.50000000000000E+01
1985.txhiplains.wheat  2.62000000000000E+03
1985.txhiplains.sorghum  1.56700000000000E+03
1985.txrolingpl.corn  6.00000000000000E+00
1985.txrolingpl.wheat  1.68000000000000E+03
1985.txrolingpl.sorghum  2.87000000000000E+02
1985.txcntblack.corn  3.61000000000000E+02
1985.txcntblack.soybeans  1.70000000000000E+01
1985.txcntblack.wheat  1.17500000000000E+03
1985.txcntblack.sorghum  8.61000000000000E+02
1985.txeast.corn  4.30000000000000E+01
1985.txeast.soybeans  2.00000000000000E+01
1985.txeast.wheat  1.29000000000000E+02
1985.txeast.sorghum  2.50000000000000E+01
1985.txedplat.corn  2.80000000000000E+01
1985.txedplat.soybeans  1.00000000000000E+00
1985.txedplat.wheat  1.28000000000000E+02
1985.txedplat.sorghum  8.70000000000000E+01
1985.txsouth.corn  2.16000000000000E+02
1985.txsouth.wheat  9.30000000000000E+01
1985.txsouth.sorghum  5.91000000000000E+02
1985.txtranspec.wheat  6.00000000000000E+00
1985.txtranspec.sorghum  8.00000000000000E+00
1986.txhiplains.corn  4.67000000000000E+02
1986.txhiplains.soybeans  5.60000000000000E+01
1986.txhiplains.wheat  2.34000000000000E+03
1986.txhiplains.sorghum  1.46800000000000E+03
1986.txrolingpl.corn  5.00000000000000E+00
1986.txrolingpl.wheat  1.40000000000000E+03
1986.txrolingpl.sorghum  2.49000000000000E+02
1986.txcntblack.corn  4.03000000000000E+02
1986.txcntblack.soybeans  2.60000000000000E+01
1986.txcntblack.wheat  8.45000000000000E+02
1986.txcntblack.sorghum  7.64000000000000E+02
1986.txeast.corn  5.60000000000000E+01
1986.txeast.soybeans  3.20000000000000E+01
1986.txeast.wheat  4.00000000000000E+01
1986.txeast.sorghum  5.10000000000000E+01
1986.txedplat.corn  6.00000000000000E+00
1986.txedplat.wheat  8.20000000000000E+01
1986.txedplat.sorghum  7.90000000000000E+01
1986.txsouth.corn  1.63000000000000E+02
1986.txsouth.wheat  6.70000000000000E+01
1986.txsouth.sorghum  4.76000000000000E+02
1986.txtranspec.wheat  7.00000000000000E+00
1986.txtranspec.sorghum  8.00000000000000E+00
1987.txhiplains.corn  4.12000000000000E+02
1987.txhiplains.soybeans  4.00000000000000E+01
1987.txhiplains.wheat  1.73000000000000E+03
1987.txhiplains.sorghum  8.95000000000000E+02
1987.txrolingpl.corn  3.00000000000000E+00
1987.txrolingpl.wheat  1.11300000000000E+03
1987.txrolingpl.sorghum  1.91000000000000E+02
1987.txcntblack.corn  3.83000000000000E+02
1987.txcntblack.soybeans  3.10000000000000E+01
1987.txcntblack.wheat  5.52000000000000E+02
1987.txcntblack.sorghum  6.07000000000000E+02
1987.txeast.corn  3.00000000000000E+01
1987.txeast.soybeans  2.10000000000000E+01
1987.txeast.wheat  4.30000000000000E+01
1987.txeast.sorghum  3.60000000000000E+01
1987.txedplat.corn  3.00000000000000E+00
1987.txedplat.wheat  8.70000000000000E+01
1987.txedplat.sorghum  6.80000000000000E+01
1987.txsouth.corn  1.70000000000000E+02
1987.txsouth.wheat  6.30000000000000E+01
1987.txsouth.sorghum  3.61000000000000E+02
```

```
1987.txtranspec.wheat  3.00000000000000E+00
1987.txtranspec.sorghum  8.00000000000000E+00
1988.txhiplains.corn  4.23000000000000E+02
1988.txhiplains.soybeans  6.10000000000000E+01
1988.txhiplains.wheat  1.47200000000000E+03
1988.txhiplains.sorghum  6.68000000000000E+02
1988.txrolingpl.corn  3.00000000000000E+00
1988.txrolingpl.wheat  9.30000000000000E+02
1988.txrolingpl.sorghum  1.23000000000000E+02
1988.txcntblack.corn  3.92000000000000E+02
1988.txcntblack.soybeans  2.50000000000000E+01
1988.txcntblack.wheat  6.29000000000000E+02
1988.txcntblack.sorghum  4.99000000000000E+02
1988.txeast.corn  3.60000000000000E+01
1988.txeast.soybeans  3.10000000000000E+01
1988.txeast.wheat  4.50000000000000E+01
1988.txeast.sorghum  2.10000000000000E+01
1988.txedplat.corn  3.00000000000000E+00
1988.txedplat.wheat  5.80000000000000E+01
1988.txedplat.sorghum  5.60000000000000E+01
1988.txsouth.corn  2.22000000000000E+02
1988.txsouth.wheat  5.40000000000000E+01
1988.txsouth.sorghum  2.82000000000000E+02
1988.txtranspec.wheat  3.00000000000000E+00
1988.txtranspec.sorghum  9.00000000000000E+00
1989.txhiplains.corn  5.92000000000000E+02
1989.txhiplains.soybeans  1.78000000000000E+02
1989.txhiplains.wheat  7.48000000000000E+02
1989.txhiplains.sorghum  1.20800000000000E+03
1989.txrolingpl.corn  3.00000000000000E+00
1989.txrolingpl.wheat  1.22500000000000E+03
1989.txrolingpl.sorghum  1.32000000000000E+02
1989.txcntblack.corn  3.69000000000000E+02
1989.txcntblack.soybeans  4.90000000000000E+01
1989.txcntblack.wheat  7.74000000000000E+02
1989.txcntblack.sorghum  6.43000000000000E+02
1989.txeast.corn  3.60000000000000E+01
1989.txeast.soybeans  4.20000000000000E+01
1989.txeast.wheat  5.50000000000000E+01
1989.txeast.sorghum  3.00000000000000E+01
1989.txedplat.corn  3.00000000000000E+00
1989.txedplat.wheat  1.21000000000000E+02
1989.txedplat.sorghum  6.30000000000000E+01
1989.txsouth.corn  1.82000000000000E+02
1989.txsouth.soybeans  5.00000000000000E+00
1989.txsouth.wheat  6.60000000000000E+01
1989.txsouth.sorghum  3.08000000000000E+02
1989.txtranspec.wheat  2.00000000000000E+00
1989.txtranspec.sorghum  2.00000000000000E+00
1990.txhiplains.corn  6.21000000000000E+02
1990.txhiplains.soybeans  2.80000000000000E+01
1990.txhiplains.wheat  1.73600000000000E+03
1990.txhiplains.sorghum  6.62000000000000E+02
1990.txrolingpl.corn  7.00000000000000E+00
1990.txrolingpl.wheat  1.35400000000000E+03
1990.txrolingpl.sorghum  1.26000000000000E+02
1990.txcntblack.corn  3.63000000000000E+02
1990.txcntblack.soybeans  4.40000000000000E+01
1990.txcntblack.wheat  8.33000000000000E+02
1990.txcntblack.sorghum  6.45000000000000E+02
1990.txeast.corn  3.40000000000000E+01
1990.txeast.soybeans  3.40000000000000E+01
1990.txeast.wheat  5.30000000000000E+01
1990.txeast.sorghum  2.30000000000000E+01
1990.txedplat.corn  3.00000000000000E+00
1990.txedplat.wheat  1.27000000000000E+02
1990.txedplat.sorghum  6.30000000000000E+01
```

1990.txsouth.corn 2.30000000000000E+02
1990.txsouth.soybeans 2.00000000000000E+00
1990.txsouth.wheat 8.30000000000000E+01
1990.txsouth.sorghum 3.67000000000000E+02
1991.txhiplains.corn 6.42000000000000E+02
1991.txhiplains.soybeans 2.40000000000000E+01
1991.txhiplains.wheat 1.15700000000000E+03
1991.txhiplains.sorghum 7.38000000000000E+02
1991.txrolingpl.corn 7.00000000000000E+00
1991.txrolingpl.wheat 9.03000000000000E+02
1991.txrolingpl.sorghum 1.41000000000000E+02
1991.txcntblack.corn 3.75000000000000E+02
1991.txcntblack.soybeans 3.80000000000000E+01
1991.txcntblack.wheat 5.56000000000000E+02
1991.txcntblack.sorghum 7.20000000000000E+02
1991.txeast.corn 3.50000000000000E+01
1991.txeast.soybeans 2.90000000000000E+01
1991.txeast.wheat 3.50000000000000E+01
1991.txeast.sorghum 2.60000000000000E+01
1991.txedplat.corn 3.00000000000000E+00
1991.txedplat.wheat 8.50000000000000E+01
1991.txedplat.sorghum 7.00000000000000E+01
1991.txsouth.corn 2.38000000000000E+02
1991.txsouth.soybeans 2.00000000000000E+00
1991.txsouth.wheat 5.60000000000000E+01
1991.txsouth.sorghum 4.09000000000000E+02
1991.txtranspec.sorghum 1.00000000000000E+00
1992.txhiplains.corn 6.94000000000000E+02
1992.txhiplains.soybeans 5.60000000000000E+01
1992.txhiplains.wheat 1.57100000000000E+03
1992.txhiplains.sorghum 1.14600000000000E+03
1992.txrolingpl.corn 8.00000000000000E+00
1992.txrolingpl.wheat 1.22500000000000E+03
1992.txrolingpl.sorghum 2.18000000000000E+02
1992.txcntblack.corn 4.05000000000000E+02
1992.txcntblack.soybeans 8.70000000000000E+01
1992.txcntblack.wheat 7.54000000000000E+02
1992.txcntblack.sorghum 1.11700000000000E+03
1992.txeast.corn 3.80000000000000E+01
1992.txeast.soybeans 6.60000000000000E+01
1992.txeast.wheat 4.80000000000000E+01
1992.txeast.sorghum 4.00000000000000E+01
1992.txedplat.corn 3.00000000000000E+00
1992.txedplat.soybeans 2.00000000000000E+00
1992.txedplat.wheat 1.15000000000000E+02
1992.txedplat.sorghum 1.09000000000000E+02
1992.txsouth.corn 2.57000000000000E+02
1992.txsouth.soybeans 4.00000000000000E+00
1992.txsouth.wheat 7.60000000000000E+01
1992.txsouth.sorghum 6.35000000000000E+02
1992.txtranspec.sorghum 2.00000000000000E+00

Chapter 18 Features to watch out for

There are a number of GAMS peculiarities that cause problems for users. In this chapter we present brief examples of each and cross-references treatments in other areas in the text.

18.1 Dynamic vs static calculations --What is and is not recomputed

As discussed in section 13.1.2 the dynamic/static nature of calculations can cause problems. Here we will elaborate on the points in that section. In GAMS a dynamic calculation is done every time a SOLVE statement is executed and is limited to the calculations contained in the equation specification statements (those with the **..** syntax). All other calculations are done only at the stage in the code where they appear -- they are not automatically repeated when the input data defining them changes. GAMS does not contain any mechanism to make a calculation dynamic and this can coupled with user frailties (some of us forget to recompute items when data are changed) can cause problems.

To illustrate this point we introduce an example - table 18.1 (nondyn.gms) - which has non dynamic calculations in lines 16, 17 and 18, which define model data. Therein the parameter “C” is calculated in line 16 is used in defining the objective function value in line 20 while the upper bounds calculated in line 18 are used whenever the model is solved. Now suppose we also set up an alternative version of the model in lines 21 through 23, wherein the calculation of the sum placed into “C” is embedded in objective function and the bounds are explicitly included as constraints. We then set up two versions of the model in lines 24 and 25 both of which should be the same. We solve the two models in lines 28 and 31 an use a report writing feature in lines 29 and 30 to save the solution. Both models at this stage yield the same solution.

Now come the problem. Suppose we then revise the data used in the calculation of “c” and the upper bounds. We do this in lines 35 and 36 and resolve the models. The resultant solutions

are

```
----     49 PARAMETER SOL
     trybefore  try2before    tryafter   try2after  tryafterc~
x1    1000.000    1000.000    1000.000      89.000      89.000
x2    2000.000    2000.000    2000.000    2000.000    2000.000
x3    3000.000    3000.000    3000.000    3000.000    3000.000

z    42000.000   42000.000   42000.000 1825623.000 1825623.000
```

Notice that the try model solution from before and after the data change is no different, whereas the try2 model gives differing solutions. Why? This occurs because the "C" and bounds calculations are automatically updated sice they appear in the equation specification in try2 but are not altered when try is solved. In fact the only way we can get the try model to change solution is by repeating the calculations after the data have been manipulated as in lines 43-45, then re solving.

Is this a GAMS bug? No it is not. Rather one must recognize that whenever calculated parameters are used in a model that one must be careful to update their calculations whenever their input data are altered. It also indicates that GAMS does not treat bounds in a parallel manner with the rest of models. Namely, the bound calculations are not dynamic and are not done whenever the model is set up, rather they are only done at the time the bound definition is executed. This leads the author to frequently include the bounds as equations to avoid such difficulties and let the presolve routines in OSL and CPLEX convert them into bounds. It would be nice if the GAMS developers included a dynamic calculation definition mechanism (which has been under discussion for years).

18.2 Fully omitted variables that won't leave

Another GAMS feature which causes difficulties and is more like a bug involves the possibility that a variable that is supposed to be eliminated by conditionals hangs around in the solution. There are actually two variants to this problem which are intertwined with the

SOLVEOPT option.

Consider the example in table 18.2. Here we set up a problem with six variables. Also note that the variables are conditionally present in the model, dependent on the values of limit and limit2. We then solve the model. Later, in lines 25 and 26, we set the limits down to zero on the variable, thus eliminating the variables and solve the model again. Also note that we introduce a report in lines 22 through 24, 28 through 30 and throughout the rest the code that stores the variables values.

Now, let's the table of solutions displayed in line 43.

```
----     43 PARAMETER SOL
                trybefore      tryaft   tryaftrep  tryaftrep2
z         .z        6.000       2.000       2.000       2.000
variablval.x1       1.000       1.000
variablval.x2       1.000       1.000       1.000       1.000
variablval.x3       1.000       1.000       1.000       1.000
secondvar .x1       1.000       1.000       1.000
secondvar .x2       1.000       1.000       1.000
secondvar .x3       1.000       1.000       1.000
```

Here note that in the first solve (trybefore) shows all six variables present, the objective function which is the sum of all the variables is equal to six. We then solve the second problem and here notice this is called tryafter in our solution table. Here the objective function equals two indicating that at optimality there are only two variables in solution, the model setup statistics also indicate that there are only two variables. But, now the problem, the solution summary table shows all six variables are nonzero. Why? GAMS contains a parameter called solveopt which tells it whether to merge a current solution in with a previously stored solution or replace that solution. By default the solution is merged and what happens is the value of the variables that were eliminated are retained from the last solve. GAMS does provide a provision to change this merge option, namely we can utilize the syntax in line 33 and tell it we are going to replace our solution.

But yet another problem occurs. When we solve now we get the solution labeled tryaftrep. Again the model has two variables, but the solution report has five. This reflects what we believe

is a bug in GAMS. Namely, under the replace option, when a variable is completely eliminated from the model its old solution values are still retained even under the REPLACE option. Notice however in the case of the variable variablval(“x1") this is not a problem as it is only involved with a partially eliminated variable and is rewritten to be zero. The only way we have found to fix this is for any variable that may be fully eliminated by conditionals is to insert a statement like line 38 where we manually zero the values. After this modification our report writing is finally right with just two variables present.

This illustrates then two general points. People who are doing repeated solves with items in loops as well as for use in basis for the GAMS base but then eliminating the variables, will need to carefully manage the variables other wise any report writing statements may have old variable values in them. A lack of attention to this may lead to one explaining to a modeling client why the implications of an omitted variable still appear in the model reports. (once the author eliminated a land transfer via conditionals in a looped solve sequence but the client found the solution report still showing land transfers) The way to manage is two fold, first in repeated solves always use the merge option. (The merge option exists because one might want to recursively solve for different time periods and not eliminate the solution from the earlier time period.) Furthermore if a all cases of a variable can be fully eliminated you need to zero out the variable levels. The problem is also exhibited for variable marginals.

Note the replace problem also exists for equations. In particular the shadow prices and levels on equations will remain for omitted equations if the merge option is active. However, the equations can freely be eliminated and they will disappear from the solution when replace is active (as the more extensive wontleav.gms file on the disk indicates).

The author thinks that GAMS corporation should undertake two code additions to correct

this problem. First, a parameter file entry should be allowed that sets the program to for now default to the REPLACE option rather than MERGE. Second, a new option called OVERWRITE should be defined which will zero out the marginals and levels for all variables which appear in any equation associated with a model even if they are removed by a conditional.

18.3 Partially omitted variables that stay

Conditionally omitted variables can be a problem in another way. In particular, when one tries to omit a variable using conditionals, but does not have the exact same conditional everywhere in the model, then the variable may be retained in the solution. This is illustrated with examples in section 13.1.5 and will not be repeated here. The general lesson is that when a variable is to be conditionally eliminated from the model one must make sure one has the same conditional everywhere. This may mean that it is desirable in the case of complicated conditionals to calculate a set which indicates when the variable is present and then condition on that set or sum over a tuple involving that set. For additional details refer to the coverage in section 13.1.5.

This is not a GAMS bug but rather a modeler oversight. GAMS could again help the user by providing a mechanism via which variables could be either activated or deactivated everywhere. This can currently de done by fixing variables at zero but that while effective leads to larger than necessary models.

18.4 Phantom Sums

One of the other problems that one must be careful of is the phantom sum problem. Namely, these authors have on more than one occasion made a mistake of summing over an item using an index which is not represented in that sum, which in effect multiplies the result by a constant. This is covered in section13.1.3, and one should make sure that the sets referenced by a sum appear somewhere within each term in the parentheses defining the sum.

This is again a modeler problem. It may be possible for GAMS to issue warnings when such sums are done.

18.5 Cumulative Data Changes

When solving in loops particularly using the procedures described in chapter seventeen, one needs to be careful to avoid cumulative data changes. In particular consider the example in table 18.3. There the parameter "limit" is changed during the loop, being multiplied by one under the first case, two under the second, and four under the third. However, while line 23 is deactivated the value of limit after line 24 is the value of the limit that preexisted after each go through of the loop. In this particular case what happens is the original value of limits is multiply by one in the first place, two in the second case, and effectively eight in the third case. This is reflected in the following small solution table, where note the variable values are cumulative.

```
----     29 PARAMETER SOL
         case1       case2       case3
x1       1.000       2.000       8.000
x2       1.000       2.000       8.000
x3       1.000       2.000       8.000
z        3.000       6.000      24.000
```

On the other hand if we activate line 23 then the solution table becomes as follows,

```
----     29 PARAMETER SOL
         case1       case2       case3
x1       1.000       2.000       4.000
x2       1.000       2.000       4.000
x3       1.000       2.000       4.000
z        3.000       6.000      12.000
```

The lesson is that one needs to be careful to reset the data particularly in a loop, unless one wants cumulative changes. More is discussed on this in Chapter 17. This phenomenon is not only relevant to loops, but rather if data are changed anywhere they are changed permanently and can only return the original values, if they are reset using a statement like line 23.

This is clearly a modeling problem and does not merit any action by the GAMS developers.

18.6 Memory Hogs

One can if one is not careful create the GAMS memory hogs. This occurs when one does not carefully manage all the possible index set combinations for an item. In turn a lot of irrelevant terms or variables or parameters are considered as discussed in chapter 10 particularly in section 10.2.1. One also must be careful with this in respect to bounds and scaling to make sure that these items are only defined for realistic items

Again no action by GAMS is mandated toward this item.

18.7 Bases in repeated solutions

A problem occasionally crops up involving an advanced basis in a repeated solution. In particular, if one is making fairly radical changes between alternative solves, then some form of advanced basis management may be necessary in order to make the solver work correctly. This is discussed in section 11.3.

18.8 Terms that should not be there

GAMS can define irrelevant terms during a model that the modeler does not feel should be present. One needs to be careful to use conditionals and to manage the variables that are present as discussed in sections 10.2.1 and 13.1.4.

Table 18.1 Example of Non-dynamic Calculations

```
 1  set varname /x1,x2,x3/
 2  set scalefac(varname) scaling factors
 3        /x1 1000,x2 500,x3 250/
 4  parameter limit(varname) /x1 1000,x2 2000,x3 3000/
 5  set sumitem items to sum over /1*3/
 6  table sumdata(sumitem,varname)
 7            x1        x2     x3
 8  1          4         4      5
 9  2          3         3      2;
10  variable           z  obj var
11  positive variables variablval(varname) variable values;
12  equations          obj objective function
13                     obj2 alternative objective function
14                     bound(varname) bounds via equations;
15  parameter c(varname) variable objective function coeficients;
16  c(varname)=sum(sumitem,sumdata(sumitem,varname));
17  variablval.scale(varname)=scalefac(varname);
18  variablval.up(varname)=limit(varname);
19  option lp=bdmlp;
20  obj.. z=e=sum(varname,c(varname)*variablval(varname));
21  obj2.. z=e=sum(varname,sum(sumitem,sumdata(sumitem,varname)
22                       *variablval(varname)));
23  bound(varname).. variablval(varname)=l=limit(varname);
24  model try /obj/
25  model try2 /obj2,bound/
26  option solprint=off;
27  parameter sol(*,*);
28  solve try using lp maximizing z;
29  sol("z","trybefore")=z.l;
30  sol(varname,"trybefore")=variablval.l(varname);
31  solve try2 using lp maximizing z;
32  sol("z","try2before")=z.l;
33  sol(varname,"try2before")=variablval.l(varname);
34  scalefac('x1')=100;
35  limit('x1')=89;
36  sumdata('2','x2')=898;
37  solve try using lp maximizing z;
38  sol("z","tryafter")=z.l;
39  sol(varname,"tryafter")=variablval.l(varname);
40  solve try2 using lp maximizing z;
41  sol("z","try2after")=z.l;
42  sol(varname,"try2after")=variablval.l(varname);
43  c(varname)=sum(sumitem,sumdata(sumitem,varname));
44  variablval.scale(varname)=scalefac(varname);
45  variablval.up(varname)=limit(varname);
46  solve try using lp maximizing z;
47  sol("z","tryaftercal")=z.l;
48  sol(varname,"tryaftercal")=variablval.l(varname);
49  display sol;
```

Table 18.2 Example of the variable that Would Not Leave

```
option limrow=0
option limcol=0;
set varname /x1,x2,x3/
parameter limit(varname)  /x1 2,  x2 2, x3 2/
          limit2(varname) /x1 1 , x2 1, x3 1/
variable           z  obj var
positive variables variablval(varname) variable values
                   secondvar(varname)  other variables;
equations          obj objective function
                   bound2(varname) bounds on secondvar
bound(varname) bounds via equations;
option lp=bdmlp;
obj.. z=e=sum(varname,variablval(varname)$limit(varname)
                     +secondvar(varname)$limit2(varname));
bound(varname)$limit(varname).. variablval(varname)=l=limit(varname);
bound2(varname)$limit2(varname).. secondvar(varname)=l=limit2(varname);
model try /all/
*option solprint=off;
parameter sol(*,*,*);
solve try using lp maximizing z;
sol("z","z","trybefore")=z.l;
sol("variablval",varname,"trybefore")=variablval.l(varname);
sol("secondvar",varname,"trybefore")=secondvar.l(varname);
limit2(varname)=0;
limit('x1')=0;
solve try using lp maximizing z;
sol("z","z","tryaft")=z.l;
sol("variablval",varname,"tryaft")=variablval.l(varname);
sol("secondvar",varname,"tryaft")=secondvar.l(varname);
display variablval.l;
display secondvar.l;
option solveopt=replace
solve try using lp maximizing z;
sol("z","z","tryaftrep")=z.l;
sol("variablval",varname,"tryaftrep")=variablval.l(varname);
sol("secondvar",varname,"tryaftrep")=secondvar.l(varname);
secondvar.l(varname)=0;
solve try using lp maximizing z;
sol("z","z","tryaftrep2")=z.l;
sol("variablval",varname,"tryaftrep2")=variablval.l(varname);
sol("secondvar",varname,"tryaftrep2")=secondvar.l(varname);
display sol;
```

Table 18.3 Example of Cumulative Data Changes in a Loop

```
option limrow=0
option limcol=0;
set varname /x1,x2,x3/
parameter limit(varname)  /x1 1,  x2 1, x3 1/
variable           z  obj var
positive variables variablval(varname) variable values;
equations          obj objective function
                   bound(varname) bounds on var;
option lp=bdmlp;
obj.. z=e=sum(varname,variablval(varname)$limit(varname));
bound(varname)$limit(varname)..
      variablval(varname)=l=limit(varname);
model try /all/
*option solprint=off;
parameter sol(*,*);
set loopover /case1,case2,case3/
parameter valuesto(loopover)
             /case1 1, case2 2, case3 4/
parameter slimit(varname) saved values of limit;
slimit(varname)=limit(varname);
loop(loopover,
*    limit(varname)=slimit(varname);
     limit(varname)=limit(varname)*valuesto(loopover);
     solve try using lp maximizing z;
     display limit;
     sol("z",loopover)=z.l;
     sol(varname,loopover)=variablval.l(varname); )
display sol;
```

18-10

References

Brooke, A., D. Kendrick, and A. Meeraus. GAMS: A User's Guide. Boyd and Fraser Publishers, Version 2.25, 1993.

Dillon, M., "Heuristic Selection of Advanced Bases for a Class of Linear Programming Models," Operations Research, 18(1970):90-100.

GAMS Development Corporation. "Sensitivity Analysis with GAMS/CPLEX and GAMS/OSL." Washington, DC, 1993.

GAMS Development Corporation. "Guide to the 'Put' Writing Facility." Washington, DC, 1990.

Greenberg, H.J. "A Primer for ANALYZE(c): A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions." Mathematics Department, University of Colorado at Denver, Denver, CO, February 1991.

Kutcher, G. and A. Meeraus, "Computational Considerations for Sectoral Programming Models," in The Book of CHAC: Programming Studies for Mexican Agriculture. R. Norton and L. Solis, (Eds.), Johns Hopkins Press for the World Bank, 1983.

Luenberger, D., Introduction to Linear and Nonlinear Programming, Addison Wesley, 1973.

McCarl, B.A. "GAMSCHECK USER DOCUMENTATION: A System for Examining the Structure and Solution Properties of Linear Programming Problems Solved using GAMS." Working Documentation, Department of Agricultural Economics, Texas A&M University, 1994.

McCarl, B.A. "Degeneracy, Duality and Shadow prices in Linear Programming," Canadian Journal of Agricultural Economics, 25(1977):70-73.

McCarl, B.A., W.V. Candler, D.H. Doster and P. Robbins, "Experiences with Farmer Oriented Linear Programming for Crop Planning," Canadian Journal of Agricultural Economics, 25(1977):17-30.

McCarl, B.A. and P. Nuthall, "Linear Programming for Repeated use in the Analysis of Agricultural Systems," Agricultural Systems, 8(1982):17-39.

Optimization Subroutine Library (OSL). IBM Guide and Reference, Release 2. Ed. J.A. George and J.W.H. Liu. IBM Corporation, 1990-91.

Orchard-Hays, W., Advanced Linear-Programming Computing Techniques, McGraw-Hill Book Company, 1968.

Williams, H., Model Building in Mathematical Programming, John Wiley & Sons, 1978.

Appendix I - Good GAMS modeling practices

There are a number of practices which may be followed which improve readability, and documentation of GAMS models. These notes discuss those practices. The topics of discussion involve naming conventions, setting up data, set specification, typing, indexing and conditionals.

I.1 Naming Conventions

The full benefit of GAMS use is not realized unless one includes explanatory names and labels for items. Development of variables of the form

"X(A,B,C)"

is vastly inferior to a potentially equivalent label like

"PRODUCTION (PLANT, ITEMS, PROCESS).

Furthermore, when specifying sets, set elements, data items, variables etc., one should take advantage of the up to 80 character names which can be entered. Modelers should include descriptions of the nature and units of data items and set elements. Such naming practices allow the models to be easily read and utilized by the author at a later time or by others. Not following such practices generally makes it difficult for even the modeler to go back and discover what was done at an earlier stage.

I.2 Setting up Data

Modelers should, to the extent possible, put in raw data and then compute model data requirements. GAMS has calculation procedures which allows subsequent data to be derived. For example, transportation costs are commonly estimated by applying a rate function to the distance traveled. In such a case the rate function and the distance should be entered rather than the result of external cost calculations. This allows calculations to be revised at a future time and documents assumptions as part of the permanent model record. Longer names should be used for data items

with explanations entered for data sources. Explanatory comments should be entered using either an asterisk for one line comments or the $ONTEXT/OFFTEXT syntax.

I.3 Specification of Sets

Often GAMS modelers need to decide when to use a single versus multiple sets. Sets are used to address a family of similar items. There are cases when it is convenient to have something in such a family and cases when it is not. These authors err on the side of being more extensive with set definitions. For example, in the GAMS Note 3, resource allocation problem in we prefer the second formulation where the process set refers to the three potential production methods without reference to the chair type as in the first formulation. Thus, we believe that sets should contain items treated similarly in the problem (i.e., resources like fertilizer, seed, and energy), but when there are two items crossed (i.e., monthly availability of land, labor, and water involves month and reserve) one should have two sets.[c] However GAMS is limited to ten set indices for any particular item.

One other set definition consideration involves the use of subsets. Sometimes it is desirable to have items which can be treated simultaneously in some places, but separately elsewhere. For example, when entering crop budgets one might wish to enter yield along with usage of inputs, land, labor, and water, in one spot yet treat those differently elsewhere (i.e., where variable inputs might be in one equation, yield balance in another, with water and labor availability in yet a third and fourth equation). In this case, the strategy that the authors have pursued involves the uses of subsets. In particular, in the ASM Sector Model (Chang, et al.) the authors have defined a set called ALLI and in that set are all data items for a crop budget. Subsequently, subsets of ALLI are defined which identify the yield item, input, and resources, etc. The example in Table XIV.1

[c] Generally, the number of sets should be small, but to the point where the model is easy to explain and not cumbersome to work with.

shows such a case. Note how after the data references appear to subsets where commodities, inputs and resources are indexed. This mechanism allows one to both organize the input and then deal with it efficiently in the model and report writer statements.

I.4 Typing of GAMS Models

GAMS models are submitted in the form of an ASCII file. Models can format such files to improve model readability. We recommend entering the model in a fixed order. Generally, we put the data related sets first, followed by data definitions organized by class of parameters and/or type of data. For example, all the data related to transportation includes all tables, parameters, and scalers would be together followed by any calculations on those parameters. Third, we introduce variable and equation definitions. Fourth, the individual equations should be defined algebraically. Fifth, the model and solve. Finally, report writing statements should appear.

We also recommend that the input be formatted so that it is visually easy to follow. For example, one should align set names, set descriptions and set element definitions. Also when entering the equation ("..") specifications one should indent and use spaces to improve readability. Namely, when one does a sum one should indents the algebraic expression underneath that.

These practices are in the following example in Table XIV.1. Note how the names, labels and data are aligned. Also note the spacing on the SUMS showing where they begin and end.

It is also good practice to not split variables between lines in equations, but rather to keep them together with all their index positions. One can also use indenting in conjunction with the loops to know where the loop begins and ends.

Another element of good GAMS practice involves use of sets, calculations and complex dollar sign specifications. It is highly desirable to make sure that GAMS only executes over valid combinations of set indices. The absence of a global condition on definition of variables (i.e., the

ability to enter a global dollar sign defining when a variable exists) means that in a number of models one has to repeatedly includes a condition identifying whether a variable exists. This can make for long expressions. Often, it is desirable to introduce avoid cumbersome notation by computing a parameter called something like "ISEXIST" which is nonzero when a variable should be defined and then including a dollar sign conditions on it throughout the remainder of the program (see Table IV.2). In this way cumbersome algebraic statements are avoided. Also, this speeds up execution as repeated calculations are not needed.

I.5 Subscript Ordering

One can speed up GAMS execution time. Models with items defined over several sets which have lots of members can be quite slow in performance. The use of $ conditions in such models is essential. For example the report writing equation

Y=SUM((A,B,C,D,E,F,G), (DATA(A)+IT(B,C)+Y(D,E)+W(F,G))*X.L(A,B,C,D,E,F,G))

will perform much faster with the addition of a $ condition as follows

Y=SUM((A,B,C,D,E,F,G)$X.L(A,B,C,D,E,F,G),

This will result in the calculation only being done when nonzero solution values for X are involved and will avoid excess work.[d] The ordering of subscripts is also important where the data arrays should be referenced in an order consistent with their definition. For example, summing the above in the order F,D,A,C,E,B,G would be much slower. One should compute intermediate products to avoid repetitive, complex calculations (i.e., one could add the DATA and IT items into another parameter ahead of time). GAMS also gives help in reporting particularly slow statements. During execution a report appears on the screen giving the line being executed and one can observe progress making notes of statements which are computed for a long time to see if they can

[d] Such a modification reduced run time for a report writer from 2 hours to 10 minutes..

be streamlined. Also one may use the undocumented PROFILE option which produces a report of the time spent on code segments[e].

I.6 Minimizing Model Size

GAMS can generate very large problems when models contain a lot of sets with a lot of elements. The statements above on speeding up GAMS are also relevant when setting up models. It is usually highly desirable to define $ conditions on equations and the sums leading to generation of variables to avoid unneeded model features.

[e] This is invoked by including the lane OPTION PROFILE=1 or including PROFILE=1 on the GAMS call.

Appendix II: SUMMATION NOTATION and GAMS

Summation notation is difficult for some students to use and follow. Here we present notes on the mechanics of summation notation usage and some rules for proper use. This discussion is cast within the GAMS framework with presentation equivalents of common summation expressions and error messages caused by improper summation. All of the GAMS statements used herein are shown in Table 1 and are in file NOTATION.

II.1 Summation Mechanics

Summation notation is a short hand way of expressing sums of algebraic terms involving subscripted items. In order to cover the mechanics of summation notation it is useful to have a set of subscripted items and associated numerical values. Thus, let us define some data

$$x_1 = 1 \quad y_{11} = 2 \quad y_{12} = 3$$

$$x_2 = 2 \quad y_{21} = 4 \quad y_{22} = 1$$

$$x_3 = 3 \quad y_{31} = 1 \quad y_{32} = 4.$$

Now let us define a variety of summation expressions.

II.1.1 Sum of an Item

Suppose we wished to sum all values of x. This would be written as

$$\sum_{i=1}^{3} x_i = x_1 + x_2 + x_3 = 1 + 2 + 3 = 6$$

or in GAMS

```
SUM1 = SUM(I, X(I));
```

For short hand purposes if i was to be summed over all possible values, we would write this as

$$\sum_i x_i .$$

We might also express a sum as follows which indicates all of the i are summed over except i=3

$$\sum_{\substack{i \\ i \neq 3}} x_i = 3.$$

In GAMS, this is more difficult to express where one has to write a conditional ($) operation or define a subset as follows

```
SUM1 = SUM(I$(ORD(I.NE.3)), X(I));
```

or

```
SET SUBSETI(I)      /1, 2/;

SUM1 = SUM(SUBSETI, X(SUBSETI(I));
```

II.1.2 Multiple Sums

Sums over two indices consider all combinations of those items

$$\sum_i \sum_j y_{ij} = y_{11} + y_{12} + y_{21} + y_{22} + y_{31} + y_{32} = 15.$$

The equivalent GAMS expression is

```
SUM2 = SUM((I,J), Y(I,J));
```

II.1.3 Sum of Two Items

Suppose we wished to sum over two items completely where they shared a subscript

$$\sum_{i=1}^{3} \left(x_i + \sum_{j=1}^{2} y_{ij} \right) = \sum_i \left(x_i + \sum_j y_{ij} \right) = \sum_i x_i + \sum_i \sum_j y_{ij}$$

$$= x_1 + y_{11} + y_{12} + x_2 + y_{21} + y_{22} + x_3 + y_{31} + y_{32} = 21.$$

The equivalent GAMS expression is as follows

```
SUM3 = SUM(I, X(I)+SUM(J, Y(I, J));
```

or

```
SUM3 = SUM(I, X(I)) + SUM((I,J), Y(I,J));
```

On the other hand, if we wished to sum the results only for the i^{th} element and call it A_i then

$$A_i = x_i + \sum_j y_{ij} = x_i + y_{i1} + y_{i2}$$

or in GAMS

```
A(I) = X(I) + SUM(J, Y(I,J));
```

which would yield a vector [6 , 7 , 8] of results.

Sums over common subscripts can be collapsed or taken apart

$$\sum_i (x_i + z_i) = \sum_i x_i + \sum_i z_i$$

or

```
SUM4 = SUM(I, X(I) +Z(I));
```

or

```
SUM4 = SUM(I, X(I)) + SUM(I, Z(I));
```

II.2 Summation Notation Rules

Certain rules apply when writing summation notation equations. The applicable rules depend on whether the final result is an unsubscripted scalar or a subscripted family of results determined by multiple equations.

II.2.1 For a Scaler Equation

All subscripts must be dealt with in each term. Thus, it is proper to define the equation

$$B1 = \sum_i \sum_j \sum_k p_{ijk} + \sum_m \sum_n q_{mn}.$$

However, the following equations are wrong

$$B2 = p_{ijk} + q_{mn}$$

$$B3 = \sum_{j}\sum_{i} p_{ijk} + \sum_{m}\sum_{n} q_{mn}.$$

In the case of the first equation, the result would really have the subscripts i,j,k,m,n, while the second equation result would have to have a k subscript on B3 or a sum over k to be proper. Equivalent GAMS commands for the above equation expressions are

```
EQB1..    B1 =E= SUM((I,J,K),P(I,J,K)) + SUM((M,N), Q(M,N));

EQB2..    B2 =E= P(I,J,K) + Q(M,N);

EQB3..    B3 =E= SUM((I,J), P(I,J,K)) + SUM((M,N), Q(M,N));
```

Here, the first equation expression is correct, while the last two equation expressions are incorrect. If you run GAMS with the above commands, you would encounter GAMS error messages $149 which says "UNCONTROLLED SET ENTERED AS CONSTANT" meaning that you have not somehow dealt with all the subscripts in the equation.

II.2.2 For a Family of Equations

Several rules apply when one is working with a family of equations.

1. The members of the family must be specified with an indication of the subscripts which define each equation. This is done by indicating all the conditions for which the equations exist in a "for" condition.

   For example, suppose we define an equation which sets all C's equal to

2. This is done by saying

   $C_i = 2$ for all i or $C_i = 2$ for i = 1,2, ... n.

Similarly, if we wish to set a 2 dimensional variable equal to a constant, we would state

$D_{ij} = 2$ for all i and j,

while stating that for each row of the matrix E_{ij} we have the same values F_i is defined by

$E1_{ij} = F_i$ for all i and j.

The equivalent GAMS commands for the above expressions are

```
EQUATIONS
EQC(I)      EQUATION C
EQD(I,J)    EQUATION D
EQE1(I,J)   EQUATION E1;
EQC(I)..    C(I) =E= 2;
EQD(I,J)..  D(I,J) =E= 2;
EQE1(I,J).. E1(I,J) =E= F(J);
```

On the other hand, it is wrong to state

$E2_{ij} = 2$

without conditions on i and j. The equivalent GAMS commands for the above incorrect expressions are

```
EQUATION

EQE2        EQUATION E2;
EQE2..      E2(I,J) =E= 2;
```

Here you would get error message $149 which says "UNCONTROLLED SET ENTERED AS CONSTANT."

2. When writing an equation with a for statement all subscripts which are not in the for statement must be summed over. Consequently, it is proper to write

$$\sum_j \sum_k p_{ijk} = G1_i \qquad \text{for all } i$$

$$\sum_k p_{ijk} = H1_i \qquad \text{for all } i \text{ and } j$$

but improper to write

$$p_{ijk} = G2_i \qquad \text{for all } i$$

$$\sum_k p_{ijk} = H2_i \qquad \text{for all } i.$$

The equivalent

GAMS commands for the above equations are

```
EQUATIONS

EQG1(I)      EQUATION G1
EQH1(I,J)    EQUATION H1
EQG2(I)      EQUATION G2
EQH2(I)      EQUATION H2;
EQG1(I)..    G1(I)   =E= SUM((J,K), P(I,J,K));
EQH1(I,J)..  H1(I,J) =E= SUM(K, P(I,J,K));
EQG2(I)..    G2(I)   =E= P(I,J,K);
EQH2(I)..    H2(I)   =E= SUM(K, P(I,J,K));
```

in which the first two equations are correct, while the last two equations are wrong and error messages $149

"UNCONTROLLED SET ENTERED AS CONSTANT"

would again be realized.

3. In any term of an equation, the result after executing the mathematical operations in that term must be of a dimension less than or equal to the family definition in the for statement. For example, it is proper to write

$$\sum_{j}\sum_{k} p_{ijk} = L1 \qquad \text{for all } i$$

$$\sum_{j}\sum_{k} r_{ijkm} + \sum_{j} s_{ijm} = N_{im} \qquad \text{for all } i \text{ and } m$$

but wrong to write

$$p_{ijk} = L2 \qquad \text{for all } i.$$

Thus, for the following expressions, the first two equations are appropriate but the last equation would give you error message

$149 "UNCONTROLLED SET ENTERED AS CONSTANT."

4. When the dimension is less than the family definition this implies the same term appears in multiple equations. For example, in the equation

$$2 + \sum_{j}\sum_{k} p_{ijk} + \sum_{j} s_{ijm} = O_{im} \qquad \text{for all i and m,}$$

the 2 term appears in every equation and the sum involving p is common when m varies. Equivalent GAMS commands are as follows

```
EQUATION

   EQO(I,M)   EQUATION O;

   EQO(I,M)..     2 + SUM((J,K), P(I,J,K)) + SUM(J, S(I,J,M)) =E=
O(I,M);
```

5. In an equation you can never sum over the parameter that determines the family of equations. It is certainly wrong to write

$$\sum_{k}\sum_{j}\sum_{i} p_{ijk} = W_i \qquad \text{for all i.}$$

Or, equivalently, the following expressions are wrong and will result in error message $125 which says "SET IS UNDER CONTROL ALREADY."

II.2.3 Defining Subscripts

In setting up a set of equations and variables use the following principles.

Define a subscript for each physical phenomena set which has multiple members,

i.e.,

Let i denote production processes of which there are I

j denote locations of which there are J

k denote products of which there are K

m denote sales locations of which that are M.

Equivalent GAMS commands are

```
SET   I      /1*20/
      J      /1*30/
      K      /1*5/
      M      /CHICAGO, BOSTON/;
```

Define different subscripts when you are either considering subsets of the subscript set or different physical phenomena.

II.2.4 Defining and Using Variables

1. Define a unique symbol with a subscript for each manipulatable item.

   For example:

   p_{ijk} = production using process i at location j while producing good k.

Or, equivalently,

PARAMETER P(I,J,K)

or

PARAMETER PRODUCTION(PROCESS, LOCATION, GOOD)

Here, for documentation purposes, the second expression is preferred.

2. Make sure that variable has the same subscript in each place it occurs.

Thus it is proper to write

$$\text{Max} \quad \sum_i \sum_j \sum_k t_{ijk}$$

$$\sum_i \sum_j t_{ijk} = 3 \qquad \text{for all } k$$

but wrong to write

$$\text{Max} \quad \sum_i \sum_j t_{ij}$$

$$\sum_i \sum_j t_{ijk} = 3 \qquad \text{for all } k$$

$$t_{ijk} \geq 0.$$

The second model would cause error message $148 indicating "DIMENSION DIFFERENT."

3. It is bad practice to define different items with the same symbol but varying subscripts. For example consider the following

u_{ij} = amount of tires transported from i to j and

u_{kj} = amount of chickens transported from k to j.

GAMS will not permit this, giving errors like $150 "Symbolic Equations Redefined."

II.3 Equations

Modelers should carefully identify the conditions under which each equation exists and use subscripts to identify those conditions. We do not think modelers should try to overly compact the families of equations. For example, it is OK to define

$$\sum_j a_{ij} x_j \le b_i \qquad \text{for all i,}$$

where a_{ij} is use of water by period and labor by period, where i denotes water periods and labor periods and b_i simultaneously contains water and labor availability by period.

But we find it is better to define

$$\sum_j d_{ij} x_j \le e_i$$
$$\sum_j f_{ij} x_j \le h_i$$

where i denotes period,

d_{ij} denotes water use and e_i water availability,

f_{ij} denotes labor use and h_i labor availability.

II.4 Cautions and Extensions

1. Be careful when you sum over terms which do not contain the subscript you are summing over. This is equivalent to multiplying a term by the number of items in the sum.

$$\sum_{j=1}^{N} x_i = Nx_i$$

Or, in GAMS $\sum_{j=1}^{3} X_2 = 3(2) = 6$

```
SUM5A = SUM(J, X("2"));
```

2. Be careful when you have a term in a family of equations which is of a lesser dimension than the family, this term will occur in each equation.

For example, the expression

$$\sum_j x_j = z_i \qquad \text{for } i = 1,2,3$$

implies that simultaneously

$$\sum_j x_j = z_1$$
$$\sum_j x_j = z_2$$
$$\sum_j x_j = z_3.$$

3. The same rules as outlined above apply to product cases

$$\prod_{i=1}^{3} x_i = x_1 * x_2 * x_3.$$

Or, equivalently,

PRODUCTX = PROD(I, X(I));

4. The following relationships also hold for summation

a. $\sum_i K x_i = K \sum_i x_i$

b. $\sum_{i=1}^{n} KP = K \sum_{i=1}^{n} P = KnP$

c. $\sum_i \sum_j (v_{ij} + y_{ij}) = \sum_i \sum_j v_{ij} + \sum_i \sum_j y_{ij}$

d. $\sum_i \sum_j (x_i + y_{ij}) = n \sum_i x_i + \sum_i \sum_j y_{ij}$ when $j = 1,2,...n$

Table II.I. Sample GAMS Commands for Summation Notation Expressions

```
 5
 6  SETS
 7      I /1*3/
 8      J /1*2/
 9      K /1*2/
10      M /1*2/
11      N /1*3/
12
13  PARAMETERS
14
15      X(I) /1 1,2 2,3 3/
16      Z(I) /1 2,2 4,3 6/
17
18  TABLE Y(I,J)
19
20              1    2
21    1         2    3
22    2         4    1
23    3         1    4;
24
25  TABLE V(I,J)
26
27              1    2
28    1         2    3
29    2         4    1
30    3         1    4;
31
32  TABLE P(I, J, K)
33
34          1.1  1.2  2.1  2.2
35    1       1    3    5    7
36    2       2    4    6    8
37    3       1    2    3    4 ;
38
39  TABLE Q(M, N)
40
41          1   2   3
42    1     1   5  10
43    2    10   5   1;
44
45  ***************************
46  ** AI.1.1 SUM OF AN ITEM **
47  ***************************
48
49  PARAMETER
50   SUM1      SUM OF AN ITEM;
51   SUM1     = SUM(I, X(I));
52   DISPLAY SUM1;
```

```
  53
  54  *************************
  55  ** AI.1.2 MULTIPLE SUMS **
Table II. 1 (continued)

  56  *************************
  57
  58  PARAMETER
  59   SUM2      MULTIPLE SUMS;
  60   SUM2     = SUM((I,J), Y(I,J));
  61   DISPLAY SUM2;
  62
  63  *****************************
  64  ** AI.1.3 SUM OF TWO ITEMS **
  65  *****************************
  66
  67  PARAMETERS
  68   SUM3A     SUM OF TWO ITEMS-1
  69   SUM3B     SUM OF TWO ITEMS-1
  70   A(I)      SUM OF TWO ITEMS-2
  71   SUM4A     SUM OF TWO ITEMS-3
  72   SUM4B     SUM OF TWO ITEMS-3;
  73   SUM3A    = SUM(I, X(I)+SUM(J, Y(I, J)));
  74   SUM3B    = SUM(I, X(I)) + SUM ((I,J), Y(I,J));
  75   A(I)     = X(I) + SUM(J, Y(I,J));
  76   SUM4A    = SUM(I, X(I)+Z(I));
  77   SUM4B    = SUM(I, X(I)) + SUM(I, Z(I));
  78   DISPLAY SUM3A, SUM3B, A, SUM4A, SUM4B;
  79
  80  *********************************
  81  ** AI.2.1 FOR A SCALER EQUATION **
  82  *********************************
  83
  84  PARAMETERS
  85   B1   SUM FOR A SCALER EQUATION-1;
  86   B1 = SUM((I,J,K), P(I,J,K)) + SUM((M,N), Q(M,N));
  87   DISPLAY B1;
  88
  89  * $ONTEXT
  90  * THE FOLLOWING SUMMATION NOTATIONS ARE INCORRECT
  91  * IF YOU TURN THESE COMMANDS ON, YOU WILL ENCOUNTER
  92  * ERROR MESSAGES
  93  * PARAMETERS
  94  *  B2   SUM FOR A SCALER EQUATION-2
  95  *  B3   SUM FOR A SCALER EQUATION-3;
  96  *  B2 = P(I,J,K) + Q(M,N);
  97  *  B3 = SUM((I,J), P(I,J,K)) + SUM((M,N), Q(M,N));
  98  *  DISPLAY B2, B3;
  99  * $OFFTEXT
 100
```

```
 101  ****************************************
 102  ** A.I.2.2 FOR A FAMILY OF EQUATIONS **
 103  ****************************************
 104
 105  VARIABLES      C(I), D(I,J), E1(I,J), F(J);
 106  EQUATIONS
 107                 EQC(I)    EQUATION C
 108                 EQD(I,J)  EQUATION D
 109                 EQE1(I,J) EQUATION E1;
 110       EQC(I)..        C(I) =E= 2;
 111       EQD(I,J)..      D(I,J) =E= 2;
 112       EQE1(I,J)..     E1(I,J) =E= F(J);
 113
 114  * $ONTEXT
 115  * THE FOLLOWING EXPRESSION IS INCORRECT
 116  * ERROR MESSAGES WILL BE ENCOUNTERED
 117  * VARIABLES E2(I,J);
 118  * EQUATION
 119  *          EQE2      EQUATION E2;
 120  *          EQE2..    E2(I,J) =E= 2;
 121  * $OFFTEXT
 122
 123  VARIABLES G1(I), H1(I,J);
 124  EQUATIONS
 125           EQG1(I)   EQUATION G1
 126           EQH1(I,J) EQUATION H1;
 127      EQG1(I)..      G1(I) =E= SUM((J,K), P(I,J,K));
 128      EQH1(I,J)..    H1(I,J) =E= SUM(K, P(I,J,K));
 129
 130  * $ONTEXT
 131  * THE FOLLOWING EXPRESSIONS ARE INCORRECT
 132  * ERROR MESSAGES WILL BE ENCOUNTERED
 133  * VARIABLES G2(I), H2(I);
 134  * EQUATIONS
 135  *           EQG2(I)   EQUATION G2
 136  *           EQH2(I)   EQUATION H2;
 137  *      EQG2(I)..      G2(I) =E= P(I,J,K);
 138  *      EQH2(I)..      H2(I) =E= SUM(K, P(I,J,K))
 139  * $OFFTEXT
 140
 141  VARIABLES L1(I), U(I,M), R(I,J,K,M), S(I,J,M);
 142  EQUATIONS
 143            EQL1(I)   EQUATION L1
 144            EQN(I,M)  EQUATION N;
 145       EQL1(I)..      L1(I) =E= SUM((J,K), P(I,J,K));
 146       EQN(I,M)..     U(I,M) =E= SUM((J,K),R(I,J,K,M)) +
SUM(J, S(I,J,M));
 147
 148  * $ONTEXT
 149  * THE FOLLOWING EXPRESSIONS ARE INCORRECT
```

```
 150  * ERROR MESSAGES WILL BE ENCOUNTERED
 151  * VARIABLES L2;
 152  * EQUATIONS
 153  *          EQL2(I)   EQUATION L2;
 154  *     EQL2(I)..      L2 =E= P(I,J,K);
 155  * OFFTEXT
 156
 157  VARIABLE O(I,M);
 158  EQUATION
 159          EQO(I,M) EQUATION O;
 160     EQO(I,M)..     2 + SUM((J,K), P(I,J,K)) + SUM(J,
S(I,J,M)) =E= O(I,M);
 161
 162
 163  * $ONTEXT
 164  * THE FOLLOWING EXPRESSION IS INCORRECT
 165  * GAMS ERROR MESSAGES WILL BE ENCOUNTERED
 166  * VARIABLE W(I);
 167  * EQUATION
 168  *          EQW(I)    EQUATION W;
 169  *     EQW(I)..  W(I) =E= SUM((I,J,K), P(I,J,K));
 170  * $OFFTEXT
 171
 172  ****************************************
 173  ** AI.4 DEFINING AND USING VARIABLES **
 174  ****************************************
 175
 176  VARIABLES
 177    OBJ1        OBJECTIVE FUNCTION VALUE
 178    T(I,J,K)   DECISION VARIABLE;
 179  EQUATIONS
 180    OBJFUNC1    OBJECTIVE FUNCTION
 181    CONST(K)  CONSTRAINT;
 182    OBJFUNC1..    OBJ1 =E= SUM((I,J,K), T(I,J,K));
 183    CONST(K)..  SUM((I,J), T(I,J,K)) =E= 3;
 184  MODEL EXAMPLE1 /ALL/;
 185  SOLVE EXAMPLE1 USING LP MAXIMIZING OBJ1;
 186  DISPLAY T.L;
 187
 188  * $ONTEXT
 189  * THE FOLLOWING COMMANDS ARE INCORRECT
 190  * THEY WILL RESULT IN ERROR MESSAGES
 191  * VARIABLES
 192  *   OBJ2        OBJECTIVE FUNCTION VALUE
 193  *   TT(I,J,K)   DECISION VARAIBLE;
 194  * POSITIVE VARIABLE TT;
 195  * EQUATIONS
 196  *   OBJFUNC2    OBJECTIVE FUNCTION
 197  *   CONSTT(K)   CONSTRAINT;
 198  *   OBJFUNC2..   OBJ2 =E= SUM((I,J), TT(I,J));
```

```
199  *   CONSTT(K)..  SUM((I,J), TT(I,J,K)) =E= 3;
200  * MODEL EXAMPLE2 /ALL/;
201  * SOLVE EXAMPLE2 USING LP MAXIMIZING OBJ2;
202  * DISPLAY TT.L;
203  * $OFFTEXT
204
205  *********************************
206  ** AI.6 CAUTIONS AND EXTENSIONS **
207  *********************************
208
209  PARAMETER
210    SUM5A  CAUTIONS AND EXTENSIONS-1;
211    SUM5A = SUM(J, X("2"));
212    DISPLAY SUM5A;
213
214  PARAMETER
215    PRODUCT6 CAUTIONS AND EXTENSIONS-2;
216    PRODUCT6 = PROD(I, X(I));
217    DISPLAY PRODUCT6;
218
219  PARAMETERS
220    SUM7A    CAUTIONS AND EXTENSIONS-3
221    SUM7B    CAUTIONS AND EXTENSIONS-3
222    SUM8A    CAUTIONS AND EXTENSIONS-4
223    SUM8B    CAUTIONS AND EXTENSIONS-4
224    SUM8C    CAUTIONS AND EXTENSIONS-4
225    SUM9A    CAUTIONS AND EXTENSIONS-5
226    SUM9B    CAUTIONS AND EXTENSIONS-5
227    SUM10A   CAUTIONS AND EXTENSIONS-6
228    SUM10B   CAUTIONS AND EXTENSIONS-6;
229    SUM7A =  SUM(I, 5*X(I));
230    SUM7B =  5*SUM(I, X(I));
231    SUM8A =  SUM(I, 5*10);
232    SUM8B =  5*SUM(I, 10);
233    SUM8C =  5*3*10;
234    SUM9A = SUM((I,J), V(I,J)+Y(I,J));
235    SUM9B = SUM((I,J), V(I,J)) + SUM((I,J), Y(I,J));
236    SUM10A = SUM((I,J), X(I)+Y(I,J));
237    SUM10B = 2*SUM(I, X(I)) + SUM((I,J), Y(I,J));
238    DISPLAY SUM7A, SUM7B, SUM8A, SUM8B, SUM8C,
239            SUM9A, SUM9B, SUM10A, SUM10B;
```

Appendix III GAMSMAP Usage

GAMSMAP is a program which creates a set of output identifying where in a program that certain items appear and where they are used. Examples of the output appear in chapter 11.

GAMSMAP is used via a two step process. First the base model must be run with the run time parameter rf=list invoked. Second GAMSMAP is run. The GAMSMAP output is then present on 4 output files.

File Name	Contents
gamsout	Contains a list of parameters that are computed during program execution and is designed to give one a list of items that are program outputs, although other things would also be included
gamsmap.sc0	Lists all files which GAMS executes during program execution. Also lists items which are defined but not used.
gamsmap.sc1	Lists all places where actions are undertaken on SETS, PARAMETERS, EQUATIONS, VARIABLES, and MODELS in the program showing where items are initially identified or defined(DECLARED,DEFINED), given values (ASSIGNED), used as indices (CONTROL), or used (REF). Also gives the items worked on by file in the program.
gamsmap.sc2.	Lists all items given input data. Intended to list data that are input to a model and where they appear.

In using GAMSMAP the program should not use save or restart files as the RF option only generates option on the files included in the current run.

Example for GAMSMAP:

Suppose we want to use GAMSMAP to get information on the chapter 5 ASM model example. To do this we first reconstruct the r.bat save / restart procedure into one composite model. We then form the file as follows:

```
$include allofit.gms
$include asmmodel.gms
$include asmsolvf.gms
$include asmrept.gms
```

Suppose we call this file ALLFILES.

After that, we execute gams with the command

```
gams ALLFILES rf=list
```

Then execute GAMSMAP.

This generates the results in Table 11-3 in the four files.