Course Materials from GAMS 2 Class

Conditional Compilation

Bruce A. McCarl

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

mccarl@tamu.edu
mccarl@bihs.net
agrinet.tamu.edu/mccarl

409-693-5694
409-845-1706

Conditional Compilation
Basics

GAMS provides features which allow one to change the basic structure of model being utilized at compile time.

These features allow one to

- Simplify maintenance of models which share common features but have significant differences with the differences involving features which
  - Could not simultaneously exist in a compiled GAMS code, or
  - Add memory, execution time or solver requirements which are desirable to avoid if not needed.
- Develop utilities which may be used across a wide variety of applications in different contexts.

The above statements are a bit obtuse and are best understood with examples

First I will introduce the language elements then I will provide case specific examples

Conditional Compilation
Control variables (control.gms)

One can use either control variables or the parameters of the batinclude statement to cause compile time changes in program structure.

Control variables defined in three different ways. Difference involves manner in which control variables are accessible or retained in included code.

```
$set varname value
$setlocal varname value
$setglobal varname value
```

where varname is any user chosen variable name
value is an optional text or numerical value

Variables defined with

$setglobal	available throughout the code
$setlocal	available only in code module where defined
$Set	available in code module where defined and any code included therein

Note $show shows all active variables at point where command is placed but is not always accurate on setlocal control variables as they are all listed but can only be used in module where defined

Conditional Compilation
Control variables (control.gms)

The sequence

$set it 1	sets a variable called it to 1
$setlocal yy	creates a variable called yy
$setglobal gg what	sets variable gg to “what”
$include includ	includes the file includ.gms
$show	shows all control variables

where includ.gms is

```
$set inincs
$setlocal inincsl
$setglobal inincsg
$show
```

This has 2 instances of $show, the first in the lst file reveals

```
LEVEL SETVAL    TYPE      NUM TEXT
-------------------------------------
    1 ININCSL     LOCAL         1
    1 ININCS      SCOPED        1
    0 YY          LOCAL         1
    0 IT          SCOPED        1 1
    0 GG          GLOBAL        1 what
    1 ININCSG     GLOBAL        1
```

The second

```
LEVEL  SETVAL      TYPE         NUM TEXT
-------------------------------------
    0 YY          LOCAL         1
    0 IT          SCOPED        1 1
    0 GG          GLOBAL        1 what
    1 ININCSG     GLOBAL        1
```

Note only the item defined as $setglobal in the included file carries over. Also values assigned are in the text field

Conditional Compilation
$IF and $IF Not (basicif.gms)

Conditional control is defined by utilizing
$IF and $IF NOT

The basic form of this is
$IF condition GAMS command
$IF NOT condition GAMS command

where condition is a conditional expression and
GAMS command is a one line GAMS instruction

Example (basicif.gms)

```
scalar x /1/;
scalar y /1/;
$setglobal gg
$setglobal tt doit
$if setglobal gg display x;
$if not setglobal gg display y;
$if "%tt%" == "doit" x=x*2;
$if not "%tt%" == "doit" y=y/4;
```

Effect on compiled code

```
1  scalar x /1/;
2  scalar y /1/;
5  display x;
7  x=x*2;
```

Note the lines in blue are suppressed because the $IF fails
%varnam% retrieves text in variable

Conditional Compilation
Overcoming One line if (goto.gms)

The one GAMS statement limits ones options. This can be overcome via use of

$GOTO labelname
$LABEL labelname

Example (goto.gms)

```
scalar y /1/;
$setglobal gg
$if setglobal gg $goto yesgg
y=y+3;
$label yesgg
display y;
*after yesgg
$if not setglobal gg $goto nogg
y=y/14;
display y;
$label nogg
```

Effect on code

```
1  scalar y /1/;
4  display y;
5  *after yesgg
7  y=y/14;
8  display y;
```

Note red line is suppressed because $IF causes $GOTO to occur

Conditional Compilation

Fancy forms of IF

Quite a few other forms of $IF are possible. Here are the ones I know about

$if errorfree	have compile errors occurred
$if declared item	has item been declared in a set, parameter, table etc statement
$if defined item	if this item is defined with data somewhere
$if dimension 0 itemname	is this item of dimension zero (a scalar)
$if dimension 1 itemname	is this item of dimension 1 (a parameter with one index set -- a(i))
$if dimension 3 itemname	is this item of dimension 3 (a parameter a(i,j,k)) cases 0-10 are allowed
$if settype itemname	is this item a set
$if partype itemname	is this item a parameter
$if vartype itemname	is this item a variable
$if equtype itemname	is this item an equation
$if exist filename	does this file exist
$if "%varname%" == "text"	does this control variable's text equal text in quotes (note 2 =)

Note item can be any GAMS item not just control variables

Conditional Compilation

Other things you can do

$abort message	stops compilation and writes message
$log message	send message to log file
$error message	generates compile error with message
$exit	exits compilation
$goto %varname%	goes to place specified by variable
$call filename	executes program during compilation but does not return info from program
$execute filename	executes program during model run but does not return info from program
$onlisting	turns on listing
$onmulti	allows data item redefinition
$onempty	allows empty items
$setargs args	sets arguements for a program call or execute
$onuni	suppresses checking of index validity
$shift	DOS shift command
$echo	Copies text to file

For examples see

GNUPLTXY.gms , xldump.gms in inclib or
program files/GAMSide/inclib for rutherford's stuff

Conditional Compilation

Nonlinear vs linear illustration (nlp-lp.gms)

Here we treat a model as either linear or nonlinear depending on control variable

```
$setglobal nonlin yes
*$setglobal nonlin no
variables            z   objective
positive variables   x   decision variables;
equations            obj
                     xlim;
$if %nonlin% == yes $goto nonlin
        obj..   z=e=3*x;
$goto around
$label nonlin
        obj..   z=e=3*x-3*x**2;
$label around
        xlim..   x=l=4;
model cond /all/;
$if %nonlin% == yes solve cond using nlp maximizing z;
$if not %nonlin% == yes solve cond using lp maximizing z
```

When NONLIN is set to yes we get

```
 3  variables            z   objective
 4  positive variables   x   decision variables;
 5  equations            obj
 6                       xlim;
 8          obj..   z=e=3*x;
10          xlim..   x=l=4;
11  model cond /all/;
13  solve cond using lp maximizing z;
```

Otherwise

```
 3  variables            z   objective
 4  positive variables   x   decision variables;
 5  equations            obj
 6                       xlim;
 8          obj..   z=e=3*x;
10          xlim..   x=l=4;
11  model cond /all/;
13  solve cond using lp maximizing z;
```

Conditional Compilation

Multi - single mode commodity transport (mode.gms)

Here we have our model considering modes or not depending on control variable

```
$setglobal mode
Sets Source       plants   / Seattle, "San Diego" /
     Destinaton   markets          / "New York", Chicago, Topeka / ;
Parameters  Supply(Source)   Supply at each source
           /seattle  350, "san diego" 600  /
            Need(Destinaton) Demand at each market
           /"new york" 325, chicago   300,  topeka 275  / ;
Table distance(Source,Destinaton)  distance in thousands of miles
                    "new york"      chicago      topeka
      seattle          2.5           1.7          1.8
      "San diego"      2.5           1.8          1.4  ;
$if setglobal mode $goto mode
Scalar  prmilecst   freight cost in $ per case per 1000 miles /90/
        loadcost    freight loading cost in $ per case /25/        ;
Parameter trancost(Source,Destinaton)  transport cost in dollars per case ;
   trancost(Source,Destinaton) =
         loadcost + prmilecst * distance(Source,Destinaton) ;
$goto around
$label mode
set mode /truck,train/
parameter prmilecst(mode)  /truck 90,train 70/
          loadcost(mode)   /truck 25,train 100/        ;
Parameter trancost(Source,Destinaton,mode)  transport cost  ;
   trancost(Source,Destinaton,mode) =
         loadcost(mode) + prmilecst(mode) * distance(Source,Destinaton) ;
$label around
Positive Variable
$if setglobal mode transport(Source,Destinaton,mode) shipment quantities in cases;
$if not setglobal mode transport(Source,Destinaton) shipment quantities in cases;
Variable    totalcost total transportation costs in dollars ;
Equations   Costsum                total transport cost -- objective function
    Supplybal(Source)       supply limit at source plants
    Demandbal(Destinaton)   demand at destinations ;
$if not setglobal mode $goto nomode
Costsum ..  totalcost  =e=  sum((Source,Destinaton),
          sum(mode,trancost(Source,Destinaton,mode)
                  *transport(Source,Destinaton,mode)));
Supplybal(Source) ..
     sum((destinaton,mode), transport(Source,Destinaton,mode))
            =l=  supply(Source) ;
demandbal(Destinaton) ..
   sum((Source,mode), transport(Source,Destinaton,mode))
           =g=  need(Destinaton) ;
$goto modset
$label nomode
Costsum ..  totalcost  =e=  sum((Source,Destinaton),
                trancost(Source,Destinaton)
                *transport(Source,Destinaton));
Supplybal(Source) ..
     sum((destinaton), transport(Source,Destinaton))
            =l=  supply(Source) ;
demandbal(Destinaton) ..
   sum((Source), transport(Source,Destinaton))
           =g=  need(Destinaton) ;
$label modset
  Model tranport /all/ ;
  Solve tranport using lp minimizing totalcost ;
```

Conditional Compilation

Put sets or parameters (putcond.gms)

Here we put out a set or a parameter depending on input item type

```
file at
put at
set a1 set to be put/item1 first,item2 second/
parameter r(a1) parameter to be put  /item1 5,item2 6/
$batinclude outit a1
$batinclude outit r
```

where outit.gms is

```
$if not "a%1" == "a" $goto start
$error Error in outit: item to be printed is not specified.
$label start
$if declared %1    $goto declared
$error Error in outit: identfier %1 is undeclared.
$exit
$label declared
$if defined %1     $goto defined
$error Error in outit: identfier %1 is undefined.
$exit
$label defined
$if settype %1  $goto doset
$if partype %1  $goto dopar
$error Error in outit: identfier %1 is not a set or a parameter.
$exit
$label doset
put /' set %1  ' %1.ts /
loop(%1,put '    Element called ' %1.tl ' defined as ' %1.te(%1) /)
put  /
$goto end
$label dopar
$if not dimension 1 %1 $goto badnews
$if not declared wkset1 alias(wkset1,*);
$if not declared wkset2 set wkset2(wkset1);
wkset2(wkset1)=no;
$onuni
wkset2(wkset1)$%1(wkset1)=yes;
display wkset2;
put /' Parameter %1  ' %1.ts  /
loop(wkset2,put '    Element ' wkset2.tl ' equals ' %1(wkset2) /)
put  /
$offuni
$goto end
$label badnews
$error Error in outit: identfier %1 is not a one dimensional parameter.
$label end
```

Conditional Compilation

Put sets or parameters (putcond.gms)

Which becomes

```
       1  file at
       2  put at
       3  set a1 set to be put/item1 first,item2 second/
       4  parameter r(a1) parameter to be put  /item1 5,item2 6/
BATINCLUDE C:\GAMS\ADVCLASS\CLASS\EXAMPLE\CONDCOMP\OUTIT.GMS
      10  put /' set a1  ' a1.ts /
      11  loop(a1,put 'Element called' a1.tl 'definedas' a1.te(a1) /)
      12  put  /
BATINCLUDE C:\GAMS\ADVCLASS\CLASS\EXAMPLE\CONDCOMP\OUTIT.GMS
      21  alias(wkset1,*);
      22  set wkset2(wkset1);
      23  wkset2(wkset1)=no;
      25  wkset2(wkset1)$r(wkset1)=yes;
      26  display wkset2;
      27  put /' Parameter r  ' r.ts  /
      28  loop(wkset2,put ' Element' wkset2.tl 'equals ' r(wkset2) /)
      29  put  /
```

Note lines 21-25 figure out the set elements r is defined over and put it in set wkset2

Much more of this can be found in Rutherfords files on

http://nash.colorado.edu/tomruth