

Massaging Data

- Parameters/scalars can also be set by assignment

- ┆ SCALAR rho ;
- ┆ rho = 10 ;

- Assignments are performed in order

- ┆ SCALAR a / 15 / ;
- ┆ a = a/3 ;
- ┆ a = a*2 ;

Massaging Data

- Here is an example where ALIAS helps:

- ┆ SET i / z1*z4 / ;
- ┆ ALIAS (i,j) ;
- ┆ PARAMETER s(i,j) A matrix ;
- ┆ s(i,j) = 1 ;
- ┆ s('z1',j) = 2 ;
- ┆ s(j,j) = 3 ;

Massaging Data

■ Let's see what happens during assignments:

■ $s(i,j)=1;$ $s('z1',j) = 2 ;$ $s(j,j) = 3 ;$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}$$

Displaying Results

■ Displaying results during execution can be useful

■ E.g., the following might be useful to see in the above program:

 | DISPLAY "s after one assignment",s ;

 | DISPLAY "s after two assignments",s ;

■ Note we can display titles and that domain lists are not specified in DISPLAY commands

Data Messaging

- $aX^b = a(X^b)$ or $(aX)^b$?
- Usually, one takes the first answer as being correct
- Why is this, and are there any general rules?

Data Messaging

- Arithmetic is a lot like Excel, BASIC, C and other languages
 - ┆ Precedence of operations is:
 - Exponentiation (**)
 - Multiplication (*) and division (/)
 - Addition (+) and subtraction (-)
 - ┆ All numbers are treated as real with a few extensions

Data Messaging

- With real numbers, a computer calculates y^x as $\exp(x \cdot \log(y))$
 - ┆ This is a problem if $y \leq 0$!
 - ┆ However y^x makes sense when $y \leq 0$ when x is an integer (a whole number)
- GAMS provides a special function: $y^x = \text{POWER}(y,x)$ that works for integer x and all y

Data Messaging

- Other functions operate over sets
 - ┆ SET row / r1,r2 / , col / c1*c3 / ;
 - ┆ TABLE data(row,col) A bunch of numbers
 - ┆ c1 c2 c3
 - ┆ r1 1 2 5
 - ┆ r2 3 3 1 ;
 - ┆ PARAMETER rowsum(row) ;
 - ┆ rowsum(row) = SUM(col,data(row,col)) ;
 - ┆ SCALAR matsum ;
 - ┆ matsum = SUM((row,col),data(row,col)) ;

Data Messaging

■ Other functions that operate over sets:

- | PROD(i,s(i)) yields the product over the set i of the values in the parameter s(i)
- | SMIN(j,s(j)) yields the minimum value in s(j) ranging over the set j
- | SMAX is analogous to SMIN, but produces the maximum rather than the minimum

Data Messaging

■ The following provides examples of PROD, SMIN, and SMAX:

- | SET j / 1*3 / ;
- | PARAMETER x(j) / 1 5, 2 3, 3 2 / ;
- | SCALAR xprod,xmin,xmax ;
- | xprod = PROD(j,x(j)) ;
- | xmin = SMIN(j,x(j)) ;
- | xmax = SMAX(j,x(j)) ;

■ Results: xprod = 30, xmin = 2, xmax = 5

Data Messaging

I GAMS extends arithmetic as follows

Values		Operation		
x	y	x**y	POWER(x,y)	x/y
2	2	4	4	1
-2	2	UNDF	4	-1
2	2.1	4.29	UNDF	0.95
NA	2.5	NA	NA	NA
3	0	1	1	UNDF
INF	2	INF	INF	INF
2	INF	UNDF	UNDF	0

Purdue University Ag. Econ. 652
Lecture 4

11

Data Messaging

I Conditional assignments can also be used

```
| SET j /1*5/, m(j) /1,3,5/;  
| ALIAS (j,jj) ;  
| PARAMETERS a(j),g(j),s(j) ;  
| a(j) = ord(j) ; g(j) = a(j) ;  
| a(j) = 1$(ord(j) le 3) ;  
| g(j)$ (ord(j) le 3) = 1 ;  
| s(j) = (g(j)-1)/sum(jj,g(jj)-2) ;  
| a(m) = (1/g(m))$ ((ord(m) eq 2) or (ord(m) eq 3)) ;  
| s(j)$s(j) = 1/s(j) ;
```

Purdue University Ag. Econ. 652
Lecture 4

12

Data Messaging

- | Line 4: $a(j) = \text{ord}(j)$; $g(j) = a(j)$;
- | After line 4: $a(j) = [1,2,3,4,5]$, $g(j) = [1,2,3,4,5]$
- | Line 5: $a(j) = 1 \text{ if } (\text{ord}(j) \leq 3)$;
- | After line 5: $a(j) = [1,1,1,0,0]$, $g(j) = [1,2,3,4,5]$
- | Line 6: $g(j) \text{ if } (\text{ord}(j) \leq 3) = 1$;
- | After line 6: $a(j) = [1,1,1,0,0]$, $g(j) = [1,1,1,4,5]$
- | Line 7: $s(j) = (g(j)-1)/\text{sum}(j,j,g(j))-2$;
- | After line 7: $a(j) = [1,1,1,0,0]$, $g(j) = [1,1,1,4,5]$,
- | $s(j) = [0,0,0,1.5,2]$

Data Messaging

- | Line 8:
- | $a(m) = (1/g(m)) \text{ if } ((\text{ord}(m) \text{ eq } 2) \text{ or } (\text{ord}(m) \text{ eq } 3))$;
- | After line 8: $a(j) = [0,1,1,0,0.2]$, $g(j) = [1,1,1,4,5]$,
- | $s(j) = [0,0,0,1.5,2]$
- | Line 9: $s(j) \text{ if } s(j) = 1/s(j)$;
- | After line 9: $a(j) = [1,1,1,0,0.2]$, $g(j) = [1,1,1,4,5]$,
- | $s(j) = [0,0,0,0.667,0.5]$

Data Messaging

I Summary

- | Conditional statements involving EQ, LE, GE, LT, GT may all be used
- | Compound conditions can use AND, OR and NOT
- | If the condition appears on the right-hand side of the assignment and is false the preceding token vanishes
- | If the condition appears on the left-hand side of the assignment and is false the assignment is not performed
- | Numbers have logical value: only zero is FALSE

Data Messaging

I IF ... THEN ... ELSE structure is useful when a list of operations depends on one condition:

- | IF (alpha GT beta,
- | s(i) = t(i)**2 ;
- | g(i) = s(i) + t(i) ;
- | ELSE
- | s(i) = 0 ;
- | g(i) = 7 ;
- |);

Variables

- Unlike parameters, variables are “concepts” rather than numbers
- Values for variables are typically computed by solving a model
- A variable’s value may also be initialized prior to solving a model

Variables

- Variables are declared via a VARIABLES statement:
 - | VARIABLES x(time),y(time,space),z ;
 - | VARIABLES
 - | bananas(tree) Number of bananas per tree
 - | snakes(pit,t) Number of snakes in pit at time t ;
- Including units in comments is a good idea!

Variables

- In addition to values, variables have bounds

- | One way to set the bounds is through the declaration:

- | POSITIVE VARIABLE x ;

- | x will be bounded between 0 and +INF

- The word POSITIVE determines the bounds

Variables

- Other variable types and their bounds:

Variable Type	Lower	Upper
NEGATIVE	-INF	0
FREE (Default)	-INF	+INF
INTEGER	0	100
BINARY	0	1

- Integer and binary types must also be whole numbers

Variables

- We refer to attributes (e.g., bounds, values) via suffixes:

Suffix	Meaning
.LO	Lower bound
.UP	Upper bound
.L	Current level
.M	Marginal (i.e., penalty cost)

Variables

- A variable with a suffix appended acts just like a parameter:

- | $x.lo(j) = 7$;
- | $land.up = 100$;
- | $area.l(crop,t) = base.l(crop,t)$;

- The exception is the additional bound type .FX which refers to both upper and lower bounds simultaneously

Equations

■ These are declared much like variables:

| EQUATIONS

| cost Cost function

| prd(t) Production in period t

| bal(t,s) Balance in period t at site s ;

Equations

■ Structure of equations is defined by a special type of assignment:

| cost .. total =E= sum(t,output(t)*costs(t)) ;

■ Form is:

| <eqn. name> .. <expression> <rel> <expression>;

Equations

■ The different types of relations are:

Relation	Meaning
=E=	=
=L=	≤
=G=	≥

■ Note that there are no strict inequalities – those make sense mathematically, but not for numerical mathematical programming

Equations

■ Equations can be indexed:

| $\text{prd}(t) \dots \text{SUM}(g, \text{supply}(g,t)) =L= \text{demand}(t) ;$

■ Conditions can be used to define equations:

| $\text{prd}(t) \dots \text{SUM}(g, \text{supply}(g,t)) =L=$
| $\text{demand}(t) \$(\text{ORD}(t) \text{ gt } 1) ;$

Equations

- Conditions can also be used to suppress equations:
 - | $\text{prd}(t)\$(\text{ORD}(t) \text{ gt } 1) ..$
 - | $\text{SUM}(g, \text{supply}(g,t)) =L= \text{demand}(t) ;$
- Conditions for *equation assignments* work the same as for parameter assignments
- This only works with “\$” not with IF ... ELSE

Equations

- If the condition appears on the right-hand side of the equation assignment, the preceding token disappears when the condition is false
- If the condition appears on the left-hand side of the equation assignment, the equation is not generated (assignment not done) when the the condition is false

Equations

- Like variables, suffixes are used to refer
 - | Level (.L)
 - | Lower bound (.LO)
 - | Upper bound (.UP)
 - | Marginal (.M) which is also called the shadow price

Model Statement

- The model declaration serves two functions:
 - | Naming the model
 - | Providing a list of the equations in the model
- Examples of valid model commands:
 - | MODEL economy / bal,cost / ;
 - | MODEL route A vehicle routing model / ALL / ;

Solve Statement

- The solve statement instructs GAMS to assemble the equations for a particular model based on current parameter values and solve
- It also tells GAMS what type of model it is (and therefore what to use to solve it)
- Finally, the solve statement indicates what variable is to be optimized and whether it should be minimized or maximized

Solve Statement

- Here are some legitimate solve statements:
 - | SOLVE economy USING LP MAXIMIZING gdp ;
 - | SOLVE route USING NLP MINIMIZING fuelcost ;
- The general form is:
 - | SOLVE <model name> USING <model type>
 - | <MINIMIZING or MAXIMIZING> <variable name> ;

Solve Statement

■ Principle model types are:

Type	Features
LP	Linear program
MIP	Mixed integer program
RMIP	Relaxed mixed integer program
NLP	Nonlinear program
DNLP	Nondifferentiable nonlinear program
MCP	Mixed complementarity problem

More on Sets

■ Set membership can also be achieved by direct assignment:

SET firms	All firms / $i1*i2000$ /
efirms(firms)	Efficient firms ;
...	
efirms(firms)=YES\$(profit.l(firms) EQ opt.l(firms)) ;	

More on Sets

- In some models it is useful to be able to refer to the “current”, “next” or “previous” element of a set
 - ┆ The current element is referred to as $x(j)$
 - ┆ The next element is $x(j+1)$ this is called a “lead”
 - ┆ The previous element is $x(j-1)$ this is called a “lag”

- This only works if the set is ordered

More on Sets

- Complications come at the ends of the set
 - ┆ What does $x(j-1)$ mean if j is the first element in the set?
 - ┆ What does $x(j+1)$ mean if j is the last element in the set?

- The answer in both cases is zero!

- These are called “end-off” lags and leads

More on Sets

- Sometimes lagging from the first set element should give the *last* element, and
- Leading from the last element should give the *first* element
- (This is like time on a clock 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2,...)
- These are called *circular* leads and lags

Purdue University Ag. Econ. 652
Lecture 4

37

More on Sets

- Circular leads and lags are specified by doubling the lead or lag operator

Purdue University Ag. Econ. 652
Lecture 4

38

Data Messaging

- When a block of commands needs to be done in sequence across a set, use LOOP:

```
| SET j / 1*4 / ;  
| PARAMETERS a(j),b(j) ;  
| b(j) = 2 ;  
| LOOP(j,  
|   a(j) = a(j-1) + b(j) ;  
| );
```

- At the end, $a(j) = [2,4,6,8]$