LINGO 8.0 Software Tutorial

Created by:
Anne Hummel
Kris Thornburg

Introduction to LINGO 8.0

- LINGO is a software tool designed to efficiently build and solve linear, nonlinear, and integer optimization models

Creating a LINGO Model

- An Optimization model consists of 3 parts
  - Objective Function
    - A single formula that describes exactly what the model should optimize
  - Variables
    - Quantities that can be changed to produce the optimal value of the objective function
  - Constraints
    - formulas that define the limits on the values of the variables

A Sample Model

```
! A cookie store can produce drop cookies and decorated
  cookies, which sell for $1 and $1.50 apiece,
  respectively.  The two bakers each work 8 hours per day
  and can produce up to 400 drop cookies and 200 decorated
  cookies.  It takes 1 minute to produce each drop cookie
  and 3 minutes to produce each decorated cookie.  What
  combination of cookies produced will maximize the
  baker's profit? ;

MAX = 1*Drop + 1.5*Deco;

Drop <= 400;
Deco <= 200;

1/60*Drop + 3/60*Deco <=16;
```

Things to notice

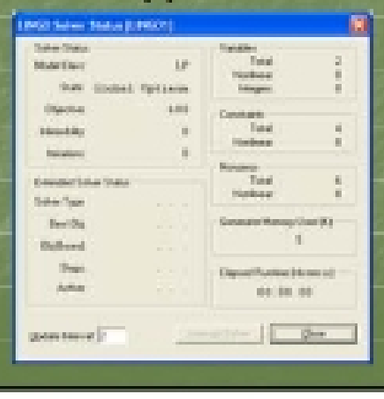
- Comments in the model are initiated with an exclamation point (!) and appear in green text
- LINGO specified operators and functions appear in blue text
- All other text is shown in black
- Each LINGO statement must end in a semi-colon (;)
- Variable names are not case-sensitive and must begin with a letter (A-Z)

Solving a LINGO Model

- Once the model has been entered into the Model Window, it can be solved by:
  - clicking the *Solve* button
  - Selecting Solve from the LINGO menu
  - Using the ctrl+s keyboard shortcut
- Errors (if any) will be reported

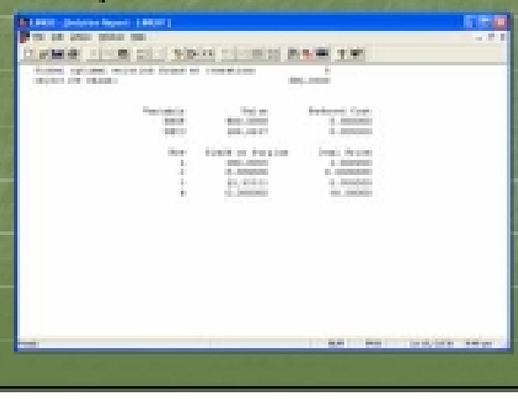
LINGO Solver Status Window

- If no errors are found, the LINGO Solver Status window appears



LINGO Solution Report Window

- Close the Solver Status window to see the Solution Report window



LINGO Solution Report Window

- Slack or Surplus
  - Zero if a constraint is completely satisfied as an equality
  - Positive shows how many more units of the variable could be added to the optimal solution before the constraint becomes an equality
  - Constraint has been violated if negative

LINGO Solution Report Window

- Reduced Cost
  - How much the objective function would degrade if one unit of a variable (not included in the current solution) were to be included
- Dual Price
  - How much the objective function would improve if the constraining value is increased by one unit

Using Sets in LINGO

- LINGO allows you to group many instances of the same variable into sets
  - Example: If a model involved 27 delivery trucks, then these 27 trucks could be described more simply as a single set
- Sets may also include attributes for each member, such as the hauling capacity for each delivery truck

Using Sets

- *SETS* section must be defined before any of the set members are used in the model's constraints
- Primitive set example:

```
SETS:
  Trucks/TR1..TR27/:Capacity;
ENDSETS
```

Using Sets (cont.)

- Derived set example:

```
SETS:
   Product/X Y/;
   Machine/L M/;
   Make(Product Machine)/X L, X M,
     Y M/;
ENDSETS
```

Set Looping Statement Examples

```
@FOR(Trucks(T): Capacity(T)<=3000);
```

- This @FOR statement sets the hauling capacity for all 27 delivery trucks in the Trucks set to at most 3000 pounds

```
TOTAL_HAUL=@SUM(Trucks(J): Capacity(J));
```

- This @SUM statement calculates the total hauling capacity from the individual trucks

LINGO Data Section

- Values can be defined for different variables
  - Set numbers
  - Attributes for sets
  - Scalar variable parameters

LINGO Data Example

```
SETS:
   SET1 /A, B, C/: X, Y;
ENDSETS

DATA:
   X = 1, 2, 3;
   Y = 4, 5, 6;
ENDDATA
```

Variable Types in LINGO

- All variables in a LINGO model are considered to be non-negative and continuous unless otherwise specified
- LINGO's four variable domain functions can be used to override the default domain for given variables

Variable Types in LINGO (cont.)

- @GIN – any positive integer value
- @BIN – a binary value (ie, 0 or 1)
- @FREE – any positive or negative real value
- @BND – any value within the specified bounds