SUPPLEMENT 2 TO CHAPTER 3

More about LINGO

Supplement 1 to Chapter 3 describes and illustrates how LINGO can be used to formulate and solve relatively small models. We now will show how LINGO can formulate a huge model like the one for the production planning problem for the Worldwide Corporation introduced in Sec. 3.6. To make this example more concrete, we will introduce actual data into a variation of the problem with a smaller number of products, months, plants (two each), and machines (three).

We begin by presenting the complete LINGO formulation for this example below, after which we will discuss the formulation and show the complete solution.

```
! LINGO model of the WORLDWIDE CORPORATION multi-
Plant/Machine/Product/Period planning problem of section 3.6;
! Notice the model says nothing about the number or
 the names of: Products, Months, Plants, Machines, or
 shipping combinations.  That information is determined
 completely by supplied data;

SETS:
!The simple/primitive entities and their attributes;
 Product: Price, InvtCost;
 Month: ProdDaysAvail;
 Plant: InvtCapacity;
 Machine;

!The derived or combination entities and their attributes;
 PrPlMn(Product, Plant, Month): Demand, Inventory, Sales;
 PrPlMa(Product, Plant, Machine): ProdCost, ProdRate;
 PrPlPl(Product, Plant, Plant): ShipCost;
 PrPlMaMn(Product, Plant, Machine, Month): Produce;
 PrPlPlMn(Product, Plant, Plant, Month): Ship;

ENDSETS
DATA:
! Get the data describing this Month's problem from
  the user's data source. Here we use inline input;

 Product, Price, InvtCost =
   a1        41       2
   a2        56       3;

 Month, ProdDaysAvail =
  Jan      22
  Feb      19;

 Plant, InvtCapacity =
  p1      2100
  p2      2000;

 Machine = m1 m2 m3;
 Demand = !(Product x Plant x Month);
  7100 12200  9800 0
  9000 10700 10100 0;

 ProdCost = !(Product  x Plant x Machine);
  30 29 39 32 33 38
  39 43 56 45 43 54;
```

```
  ProdRate = !(Product x Plant x Machine);
   200 280 190 260 220 200
   240 300 220 320 260 225;

  Shipcost=
   0  5     ! Product a1;
   5  0
   0  3     ! Product a2;
   4  0;
ENDDATA

! Total profit = sales revenue - Production cost - inventory cost - shipping cost;
 [Profit] MAX = @SUM(PrPlMn(a,p,t): Price(a) * Sales(a,p,t))
       - @SUM(PrPlMaMn(a,p,m,t): ProdCost(a,p,m)*Produce(a,p,m,t))
        - @SUM(PrPlMn(a,p,t): InvtCost(a)*Inventory(a,p,t))
       - @SUM(PrPlPlMn(a,fp,tp,t): ShipCost(a,fp,tp)* Ship(a,fp,tp,t));

! Production capacity at each Plant, Machine, each Month;
 @FOR(Plant(p):
  @FOR(Machine(m):
   @FOR(Month(t):
 [PCap] @SUM(Product(a): Produce(a,p,m,t)/ProdRate(a,p,m))
                      <= ProdDaysAvail(t););
      ););

!Inventory for Product a in Plant p in Month t:
  Beginning inventory of Product + Production + shipped into p
   = sales + ending inventory + shipped out of p;
 @FOR(PrPlMn(a,p,t)|t #GT# 1:
    [PBal] Inventory(a,p,t-1) + @SUM(Machine(m): Produce (a,p,m,t))
         + @SUM(PrPlPlMn(a,fp,p,t): Ship(a,fp,p,t))
         = Sales(a,p,t) + Inventory(a,p,t)
         + @SUM(PrPlPlMn(a,p,tp,t): Ship(a,p,tp,t));
      );

! Assuming initial inventory = 0, Production in period 1 + shipped into p
   = sales + ending inventory + shipped out of p;
 @FOR (PrPlMn(a,p,t)|t #EQ# 1:
  [PBal1] @SUM(Machine(m): Produce(a,p,m,t))
        + @SUM(PrPlPlMn(a,fp,p,t): Ship(a,fp,p,t))
        = Sales(a,p,t) + Inventory(a,p,t)
        + @SUM (PrPlPlMn(a,p,tp,t): Ship(a,p,tp,t));
      );

! Inventory <= inventory capacity in each Plant p, each Month t;
@FOR(Plant(p):
 @FOR(Month(t):
  [MxInv] @SUM(Product(a): Inventory(a,p,t)) <= InvtCapacity(p);
   ););

! Sales cannot exceed demand;
@FOR(PrPlMn: @BND(0, Sales, Demand););
```

Discussion of the Model, the SETS Section

The line (or declaration) in the SETS section

```
    Product: Price, InvtCost;
```

states that there will be a set of products and that each product will have a Price and an inventory cost that we will call InvtCost for short. There are two types of sets in LINGO: (a) simple or primitive sets and (b) derived sets. Each element of a simple set is a "fundamental" object that cannot be broken down any further. Derived sets are constructed from primitive sets.

The declaration for the derived set:

```
PrPlMn(Product, Plant, Month): Demand, Inventory, Sales:
```

says that we will need to look at combinations of a product, a plant, and a month, and associated with each such combination will be a demand, an inventory level, and a sales level. It happens to be true that Demand will be a prespecified or input datum, while Inventory and Sales will be calculated as part of the optimization. The SETS section, however, need give no advance warning about which attributes are input data and which are variables to be determined. The name of a derived set may be chosen arbitrarily, however, it is usually helpful to make the name suggestive of what it represents, yet short, e.g., PrPlMn is short for Product x Plant x Month.

The DATA Section

The segment

```
Product,  Price,  InvtCost
   a1       41       2
   a2       56       3;
```

says that this particular instance of the problem will have two products, named a1 and a2. The former has a Price of 41 and an InvtCost of 2, while the latter has a Price of 56 and an InvtCost of 3.

The segment
```
Demand = !(Product x Plant x Month);
   7100 12200  9800 0
   9000 10700 10100 0;
```

is a bit more complicated. Recall that Demand is an attribute of an element of a derived set of all combinations of products, plants, and months. At this point it is useful to point out that when LINGO reads in data sequentially into a derived set, the rightmost subscript or element (month) is varied most rapidly, while the leftmost subscript (product) is varied most slowly. So
```
DEMAND(a1,p1,Jan) = 7100; DEMAND(a1,p1,Feb) = 12200, etc.
```

Model Equations
The first segment gives the objective function

```
! Total profit = sales revenue - Production cost - inventory cost - shipping cost;
 [Profit] MAX = @SUM(PrPlMn(a,p,t): Price(a) * Sales(a,p,t))
      - @SUM(PrPlMaMn(a,p,m,t): ProdCost(a,p,m)*Produce(a,p,m,t))
      - @SUM(PrPlMn(a,p,t): InvtCost(a)*Inventory(a,p,t))
      - @SUM(PrPlPlMn(a,fp,tp,t): ShipCost(a,fp,tp)* Ship(a,fp,tp,t));
```

The "MAX =" string effectively says, Maximize the following expression. The first @SUM says that we want to sum over all members of the set PrPlMn (i.e., all combinations of Product, Plant, and Month). If *a* is

3

the product, p is the plant, and t is the month in a particular combination, we want to multiply the price of product a times the sales of product a at plant p in month t and sum them all up.

The segment

```
! Production capacity at each plant, machine, each month;
@FOR(Plant(p):
  @FOR(Machine(m):
    @FOR(Month(t):
      [PCap] @SUM(Product(a): Produce(a,p,m,t)/ProdRate(a,p,m))
              <= ProdDaysAvail(t););
```

can be read as: for each Plant, call it p: for each Machine, call it m, for each month, call it t: generate the following constraint.

You can solve the model by clicking on the red bullseye icon on the LINGO menu bar. You can get the solution displayed by clicking on the "$x=$" menu item, giving

```
Global optimal solution found.
Objective value:                          549837.2

                Variable        Value      Reduced Cost
              PRICE( A1)       41.00000       0.000000
              PRICE( A2)       56.00000       0.000000
           INVTCOST( A1)       2.000000       0.000000
           INVTCOST( A2)       3.000000       0.000000
    PRODDAYSAVAIL( JAN)       22.00000       0.000000
    PRODDAYSAVAIL( FEB)       19.00000       0.000000
      INVTCAPACITY( P1)       2100.000       0.000000
      INVTCAPACITY( P2)       2000.000       0.000000
    DEMAND( A1, P1, JAN)       7100.000       0.000000
    DEMAND( A1, P1, FEB)       12200.00       0.000000
    DEMAND( A1, P2, JAN)       9800.000       0.000000
    DEMAND( A1, P2, FEB)       0.000000       0.000000
    DEMAND( A2, P1, JAN)       9000.000       0.000000
    DEMAND( A2, P1, FEB)       10700.00       0.000000
    DEMAND( A2, P2, JAN)       10100.00       0.000000
    DEMAND( A2, P2, FEB)       0.000000       0.000000
 INVENTORY( A1, P1, JAN)       0.000000       4.000000
 INVENTORY( A1, P1, FEB)       0.000000       41.00000
 INVENTORY( A1, P2, JAN)       0.000000       9.000000
 INVENTORY( A1, P2, FEB)       0.000000       36.00000
 INVENTORY( A2, P1, JAN)       0.000000       6.575000
 INVENTORY( A2, P1, FEB)       0.000000       53.62500
 INVENTORY( A2, P2, JAN)       0.000000       8.687500
 INVENTORY( A2, P2, FEB)       0.000000       49.62500
     SALES( A1, P1, JAN)       6868.000       0.000000
     SALES( A1, P1, FEB)       12200.00      -2.000000
     SALES( A1, P2, JAN)       6561.250       0.000000
     SALES( A1, P2, FEB)       0.000000      -7.000000
     SALES( A2, P1, JAN)       9000.000      -1.800000
     SALES( A2, P1, FEB)       10700.00      -5.375000
     SALES( A2, P2, JAN)       10100.00      -3.687500
     SALES( A2, P2, FEB)       0.000000      -9.375000
   PRODCOST( A1, P1, M1)       30.00000       0.000000
   PRODCOST( A1, P1, M2)       29.00000       0.000000
   PRODCOST( A1, P1, M3)       39.00000       0.000000
   PRODCOST( A1, P2, M1)       32.00000       0.000000
   PRODCOST( A1, P2, M2)       33.00000       0.000000
   PRODCOST( A1, P2, M3)       38.00000       0.000000
```

```
      PRODCOST( A2, P1, M1)       39.00000        0.000000
      PRODCOST( A2, P1, M2)       43.00000        0.000000
      PRODCOST( A2, P1, M3)       56.00000        0.000000
      PRODCOST( A2, P2, M1)       45.00000        0.000000
      PRODCUST( A2, P2, M2)       43.00000        0.000000
      PRODCUST( A2, P2, M3)       54.00000        0.000000
      PRODRATE( A1, P1, M1)       200.0000        0.000000
      PRODRATE( A1, P1, M2)       280.0000        0.000000
      PRODRATE( A1, P1, M3)       190.0000        0.000000
      PRODRATE( A1, P2, M1)       260.0000        0.000000
      PRODRATE( A1, P2, M2)       220.0000        0.000000
      PRODRATE( A1, P2, M3)       200.0000        0.000000
      PRODRATE( A2, P1, M1)       240.0000        0.000000
      PRODRATE( A2, P1, M2)       300.0000        0.000000
      PRODRATE( A2, P1, M3)       220.0000        0.000000
      PRODRATE( A2, P2, M1)       320.0000        0.000000
      PRODRATE( A2, P2, M2)       260.0000        0.000000
      PRODRATE( A2, P2, M3)       225.0000        0.000000
      SHIPCOST( A1, P1, P1)       0.000000        0.000000
      SHIPCOST( A1, P1, P2)       5.000000        0.000000
      SHIPCOST( A1, P2, P1)       5.000000        0.000000
      SHIPCOST( A1, P2, P2)       0.000000        0.000000
      SHIPCOST( A2, P1, P1)       0.000000        0.000000
      SHIPCOST( A2, P1, P2)       3.000000        0.000000
      SHIPCOST( A2, P2, P1)       4.000000        0.000000
      SHIPCOST( A2, P2, P2)       0.000000        0.000000
   PRODUCE( A1, P1, M1, JAN)      0.000000        7.240000
   PRODUCE( A1, P1, M1, FEB)      0.000000        4.950000
   PRODUCE( A1, P1, M2, JAN)      2688.000        0.000000
   PRODUCE( A1, P1, M2, FEB)      5320.000        0.000000
   PRODUCE( A1, P1, M3, JAN)      4180.000        0.000000
   PRODUCE( A1, P1, M3, FEB)      2915.000        0.000000
   PRODUCE( A1, P2, M1, JAN)      2161.250        0.000000
   PRODUCE( A1, P2, M1, FEB)      3965.000        0.000000
   PRODUCE( A1, P2, M2, JAN)      0.000000        3.005682
   PRODUCE( A1, P2, M2, FEB)      0.000000        3.284091
   PRODUCE( A1, P2, M3, JAN)      4400.000        0.000000
   PRODUCE( A1, P2, M3, FEB)      0.000000        4.000000
   PRODUCE( A2, P1, M1, JAN)      5280.000        0.000000
   PRODUCE( A2, P1, M1, FEB)      4560.000        0.000000
   PRODUCE( A2, P1, M2, JAN)      3720.000        0.000000
   PRODUCE( A2, P1, M2, FEB)      0.000000        1.708333
   PRODUCE( A2, P1, M3, JAN)      0.000000        3.527273
   PRODUCE( A2, P1, M3, FEB)      0.000000        5.375000
   PRODUCE( A2, P2, M1, JAN)      4380.000        0.000000
   PRODUCE( A2, P2, M1, FEB)      1200.000        0.000000
   PRODUCE( A2, P2, M2, JAN)      5720.000        0.000000
   PRODUCE( A2, P2, M2, FEB)      4940.000        0.000000
   PRODUCE( A2, P2, M3, JAN)      0.000000        4.354167
   PRODUCE( A2, P2, M3, FEB)      0.000000        7.375000
      SHIP( A1, P1, P1, JAN)      0.000000        0.000000
      SHIP( A1, P1, P1, FEB)      0.000000        0.000000
      SHIP( A1, P1, P2, JAN)      0.000000        5.000000
      SHIP( A1, P1, P2, FEB)      0.000000        10.00000
      SHIP( A1, P2, P1, JAN)      0.000000        5.000000
      SHIP( A1, P2, P1, FEB)      3965.000        0.000000
      SHIP( A1, P2, P2, JAN)      0.000000        0.000000
      SHIP( A1, P2, P2, FEB)      0.000000        0.000000
      SHIP( A2, P1, P1, JAN)      0.000000        0.000000
      SHIP( A2, P1, P1, FEB)      0.000000        0.000000
      SHIP( A2, P1, P2, JAN)      0.000000        4.887500
      SHIP( A2, P1, P2, FEB)      0.000000        7.000000
```

```
SHIP( A2, P2, P1, JAN)        0.000000           2.112500
SHIP( A2, P2, P1, FEB)        6140.000           0.000000
SHIP( A2, P2, P2, JAN)        0.000000           0.000000
SHIP( A2, P2, P2, FEB)        0.000000           0.000000

                     Row    Slack or Surplus    Dual Price
                  PROFIT        549837.2           1.000000
    PCAP( P1, M1, JAN)        0.000000           3648.000
    PCAP( P1, M1, FEB)        0.000000           2790.000
    PCAP( P1, M2, JAN)        0.000000           3360.000
    PCAP( P1, M2, FEB)        0.000000           2800.000
    PCAP( P1, M3, JAN)        0.000000           380.0000
    PCAP( P1, M3, FEB)        3.657895           0.000000
    PCAP( P2, M1, JAN)        0.000000           2340.000
    PCAP( P2, M1, FEB)        0.000000           520.0000
    PCAP( P2, M2, JAN)        0.000000           2421.250
    PCAP( P2, M2, FEB)        0.000000           942.5000
    PCAP( P2, M3, JAN)        0.000000           600.0000
    PCAP( P2, M3, FEB)        19.00000           0.000000
    PBAL( A1, P1, FEB)        0.000000          -39.00000
    PBAL( A1, P2, FEB)        0.000000          -34.00000
    PBAL( A2, P1, FEB)        0.000000          -50.62500
    PBAL( A2, P2, FEB)        0.000000          -46.62500
   PBALL( A1, P1, JAN)        0.000000          -41.00000
   PBALL( A1, P2, JAN)        0.000000          -41.00000
   PBALL( A2, P1, JAN)        0.000000          -54.20000
   PBALL( A2, P2, JAN)        0.000000          -52.31250
       MXINV( P1, JAN)        2100.000           0.000000
       MXINV( P1, FEB)        2100.000           0.000000
       MXINV( P2, JAN)        2000.000           0.000000
       MXINV( P2, FEB)        2000.000           0.000000
```

You can control whether or not you get Reduced costs and Dual prices by clicking on: LINGO | Options | General Solver | Dual Computations.

Debugging and Verification of Large Models

Developing a nontrivial model is a lot like developing a nontrivial computer program. Our first attempt may have bugs, so we may need to debug the model. A general rule is to always develop a model incrementally. Try it on a very small data set first. Add one constraint type at a time, each time verifying the model has a feasible solution. One way of debugging a small model is to look at the explicit constraints that get generated. You do this by clicking on: LINGO | Generate | Display Model. The result is as follows.

```
MODEL:
  [PROFIT] MAX= - 5 * SHIP_A1_P1_P2_JAN - 5 * SHIP_A1_P1_P2_FEB
   - 5 * SHIP_A1_P2_P1_JAN - 5 * SHIP_A1_P2_P1_FEB - 3 * SHIP_A2_P1_P2_JAN
   - 3 * SHIP_A2_P1_P2_FEB - 4 * SHIP_A2_P2_P1_JAN - 4 * SHIP_A2_P2_P1_FEB
   - 30 * PRODUCE_A1_P1_M1_JAN - 30 * PRODUCE_A1_P1_M1_FEB
   - 29 * PRODUCE_A1_P1_M2_JAN - 29 * PRODUCE_A1_P1_M2_FEB
   - 39 * PRODUCE_A1_P1_M3_JAN - 39 * PRODUCE_A1_P1_M3_FEB
   - 32 * PRODUCE_A1_P2_M1_JAN - 32 * PRODUCE_A1_P2_M1_FEB
   - 33 * PRODUCE_A1_P2_M2_JAN - 33 * PRODUCE_A1_P2_M2_FEB
   - 38 * PRODUCE_A1_P2_M3_JAN - 38 * PRODUCE_A1_P2_M3_FEB
   - 39 * PRODUCE_A2_P1_M1_JAN - 39 * PRODUCE_A2_P1_M1_FEB
   - 43 * PRODUCE_A2_P1_M2_JAN - 43 * PRODUCE_A2_P1_M2_FEB
   - 56 * PRODUCE_A2_P1_M3_JAN - 56 * PRODUCE_A2_P1_M3_FEB
   - 45 * PRODUCE_A2_P2_M1_JAN - 45 * PRODUCE_A2_P2_M1_FEB
   - 43 * PRODUCE_A2_P2_M2_JAN - 43 * PRODUCE_A2_P2_M2_FEB
```

```
    - 54 * PRODUCE_A2_P2_M3_JAN - 54 * PRODUCE_A2_P2_M3_FEB
    - 2 * INVENTORY_A1_P1_JAN + 41 * SALES_A1_P1_JAN - 2 * INVENTORY_A1_P1_FEB
    + 41 * SALES_A1_P1_FEB - 2 * INVENTORY_A1_P2_JAN + 41 * SALES_A1_P2_JAN
    - 2 * INVENTORY_A1_P2_FEB + 41 * SALES_A1_P2_FEB - 3 * INVENTORY_A2_P1_JAN
   + 56 * SALES_A2_P1_JAN - 3 * INVENTORY_A2_P1_FEB + 56 * SALES_A2_P1_FEB
    - 3 * INVENTORY_A2_P2_JAN + 56 * SALES_A2_P2_JAN - 3 * INVENTORY_A2_P2_FEB
   + 56 * SALES_A2_P2_FEB ;
[PCAP_P1_M1_JAN] 0.005 * PRODUCE_A1_P1_M1_JAN
                 + 0.004166666666666667 * PRODUCE_A2_P1_M1_JAN <= 22 ;
[PCAP_P1_M1_FEB] 0.005 * PRODUCE_A1_P1_M1_FEB
               + 0.004166666666666667 * PRODUCE_A2_P1_M1_FEB <= 19 ;
[PCAP_P1_M2_JAN] 0.003571428571428571 * PRODUCE_A1_P1_M2_JAN +
       0.003333333333333334 * PRODUCE_A2_P1_M2_JAN <= 22 ;
[PCAP_P1_M2_FEB] 0.003571428571428571 * PRODUCE_A1_P1_M2_FEB +
       0.003333333333333334 * PRODUCE_A2_P1_M2_FEB <= 19 ;
[PCAP_P1_M3_JAN] 0.005263157894736842 * PRODUCE_A1_P1_M3_JAN +
       0.004545454545454545 * PRODUCE_A2_P1_M3_JAN <= 22 ;
[PCAP_P1_M3_FEB] 0.005263157894736842 * PRODUCE_A1_P1_M3_FEB +
       0.004545454545454545 * PRODUCE_A2_P1_M3_FEB <= 19 ;
[PCAP_P2_M1_JAN] 0.003846153846153846 * PRODUCE_A1_P2_M1_JAN
      + 0.003125 * PRODUCE_A2_P2_M1_JAN <= 22 ;
[PCAP_P2_M1_FEB] 0.003846153846153846 * PRODUCE_A1_P2_M1_FEB
      + 0.003125 * PRODUCE_A2_P2_M1_FEB <= 19 ;
[PCAP_P2_M2_JAN] 0.004545454545454545 * PRODUCE_A1_P2_M2_JAN +
       0.003846153846153846 * PRODUCE_A2_P2_M2_JAN <= 22 ;
[PCAP_P2_M2_FEB] 0.004545454545454545 * PRODUCE_A1_P2_M2_FEB +
       0.003846153846153846 * PRODUCE_A2_P2_M2_FEB <= 19 ;
[PCAP_P2_M3_JAN] 0.005 * PRODUCE_A1_P2_M3_JAN
      + 0.004444444444444444 * PRODUCE_A2_P2_M3_JAN <= 22 ;
[PCAP_P2_M3_FEB] 0.005 * PRODUCE_A1_P2_M3_FEB
      + 0.004444444444444444 * PRODUCE_A2_P2_M3_FEB <= 19 ;
[PBAL_A1_P1_FEB] - SHIP_A1_P1_P2_FEB + SHIP_A1_P2_P1_FEB +
    PRODUCE_A1_P1_M1_FEB + PRODUCE_A1_P1_M2_FEB + PRODUCE_A1_P1_M3_FEB +
    INVENTORY_A1_P1_JAN - INVENTORY_A1_P1_FEB - SALES_A1_P1_FEB = 0 ;
[PBAL_A1_P2_FEB] SHIP_A1_P1_P2_FEB - SHIP_A1_P2_P1_FEB +
    PRODUCE_A1_P2_M1_FEB + PRODUCE_A1_P2_M2_FEB + PRODUCE_A1_P2_M3_FEB +
    INVENTORY_A1_P2_JAN - INVENTORY_A1_P2_FEB - SALES_A1_P2_FEB = 0 ;
[PBAL_A2_P1_FEB] - SHIP_A2_P1_P2_FEB + SHIP_A2_P2_P1_FEB +
    PRODUCE_A2_P1_M1_FEB + PRODUCE_A2_P1_M2_FEB + PRODUCE_A2_P1_M3_FEB +
    INVENTORY_A2_P1_JAN - INVENTORY_A2_P1_FEB - SALES_A2_P1_FEB = 0 ;
[PBAL_A2_P2_FEB] SHIP_A2_P1_P2_FEB - SHIP_A2_P2_P1_FEB +
    PRODUCE_A2_P2_M1_FEB + PRODUCE_A2_P2_M2_FEB + PRODUCE_A2_P2_M3_FEB +
    INVENTORY_A2_P2_JAN - INVENTORY_A2_P2_FEB - SALES_A2_P2_FEB = 0 ;
[PBALL_A1_P1_JAN] - SHIP_A1_P1_P2_JAN + SHIP_A1_P2_P1_JAN +
    PRODUCE_A1_P1_M1_JAN + PRODUCE_A1_P1_M2_JAN + PRODUCE_A1_P1_M3_JAN -
    INVENTORY_A1_P1_JAN - SALES_A1_P1_JAN = 0 ;
[PBALL_A1_P2_JAN] SHIP_A1_P1_P2_JAN - SHIP_A1_P2_P1_JAN +
    PRODUCE_A1_P2_M1_JAN + PRODUCE_A1_P2_M2_JAN + PRODUCE_A1_P2_M3_JAN -
    INVENTORY_A1_P2_JAN - SALES_A1_P2_JAN = 0 ;
[PBALL_A2_P1_JAN] - SHIP_A2_P1_P2_JAN + SHIP_A2_P2_P1_JAN +
    PRODUCE_A2_P1_M1_JAN + PRODUCE_A2_P1_M2_JAN + PRODUCE_A2_P1_M3_JAN -
    INVENTORY_A2_P1_JAN - SALES_A2_P1_JAN = 0 ;
[PBALL_A2_P2_JAN] SHIP_A2_P1_P2_JAN - SHIP_A2_P2_P1_JAN +
    PRODUCE_A2_P2_M1_JAN + PRODUCE_A2_P2_M2_JAN + PRODUCE_A2_P2_M3_JAN -
    INVENTORY_A2_P2_JAN - SALES_A2_P2_JAN = 0 ;
[MXINV_P1_JAN] INVENTORY_A1_P1_JAN + INVENTORY_A2_P1_JAN <= 2100 ;
[MXINV_P1_FEB] INVENTORY_A1_P1_FEB + INVENTORY_A2_P1_FEB <= 2100 ;
[MXINV_P2_JAN] INVENTORY_A1_P2_JAN + INVENTORY_A2_P2_JAN <= 2000 ;
[MXINV_P2_FEB] INVENTORY_A1_P2_FEB + INVENTORY_A2_P2_FEB <= 2000 ;
@BND( 0, SALES_A1_P1_JAN, 7100); @BND( 0, SALES_A1_P1_FEB, 12200);
@BND( 0, SALES_A1_P2_JAN, 9800); @BND( 0, SALES_A1_P2_FEB, 0);
@BND( 0, SALES_A2_P1_JAN, 9000); @BND( 0, SALES_A2_P1_FEB, 10700);
```

```
  @BND( 0, SALES_A2_P2_JAN, 10100); @BND( 0, SALES_A2_P2_FEB, 0);
  END
```

You can verify that all the variables and constraints you intended were generated.

   Another way of checking a model is to try extreme cases. Suppose we set ProdRate = 0 for product a1 in plant p1 on machine m1; that is, we cannot produce product a1 in plant p1 on machine m1. We would expect that in the resulting solution all the Produce variables would be zero for that combination of product, plant, and machine. Instead what happens is that we get the error message: "Arithmetic error in constraint PCAP(p1, m1, Jan)." The culprit is the constraint

```
[PCap] @SUM (product(a): Produce(a,p,m,t)/ProdRate(a,p,m)) <= ProdDaysAvail(t););
```

Because one of the ProdRate(a,p,m)is zero, there is a divide by zero. There are two possible ways of fixing this problem. The first is to put in an explicit check to avoid the divide by zero. This can be done in LINGO by modifying the above to

```
  [PCap] @SUM (product(a)|ProdRate(a,p,m) #GT# 0:
         Produce(a,p,m,t)/ProdRate(a,p,m))<= ProdDaysAvail(t); );
```

Note that we must also modify the production balance constraints similarly, for example, replace in the Pbal constraints

```
  @SUM (machine(m): Produce(a,p,m,t))
```
by
```
  @SUM (machine(m)|ProdRate(a,p,m)#GT# 0: Produce(a,p,m,t)).
```

That is, you cannot get any production of product p on machine a if the production rate is 0.

   An alternative way of fixing this bug is to change the definition of the Produce(a,p,m) from "units of product a produced in plant p in month m" to "days of production of product a in plant p in month m." The division by ProdRate in the capacity constraint is then replaced by a multiplication by ProdRate in the production balance constraints. Multiplication by zero does not cause a problem.

Getting Input Data from and Moving Results to External Files

LINGO allows you to retrieve data from external files and insert results in existing files. Suppose that we have all the data for the problem stored in a spreadsheet. The spreadsheet looks as in Fig. 3S2.1.

Figure 3S2.1 Input data for WorldWide problem stored in a spreadsheet.

We have to do two things to use the spreadsheet as the data source for the LINGO model: a) we have to modify the LINGO model, and b) we have create some range names in the spreadsheet that tell LINGO where to look for the data   The only change that needs to be made to the LINGO model is to replace the original DATA section by the following:

```
DATA:
! Get the data describing this Month's problem from
  the user's data source. Here it is a spreadsheet.
  The (only) open spreadsheet must have
  Range names matching the input item;

 Product, Price, InvtCost= @OLE( );
 Month, ProdDaysAvail    = @OLE( );
 Plant, InvtCapacity     = @OLE( );
 Machine  = @OLE( );
 Demand   = @OLE( );
 ProdCost = @OLE( );
 ProdRate = @OLE( );
 Shipcost = @OLE( );

 ! Send (part of) the solution back to the open spreadsheet into
 an appropriately named Range;
  @OLE( ) = SALES;
 ENDDATA
```

The @OLE function provides the "plumbing" to hook up a spreadsheet to a LINGO model. OLE stands for "Object Linking and Embedding". The details of this hookup are as follows: Proper matching is achieved by having range names in the spreadsheet that match the names used in the LINGO. In the LINGO model, any attribute can be retrieved from a spreadsheet by inserting a line in the DATA section like:

```
Demand   = @OLE( );
```

So there must be a Range in the spreadsheet with the name "Demand", that contains the demand data.

Each attribute to be sent back to a spreadsheet after the model is solved must have a line in the data section like:

```
@OLE( ) = SALES;
```

After the model has been solved, the spreadsheet looks as in Fig. 3S2.2. Notice the difference in the column under "Sales".
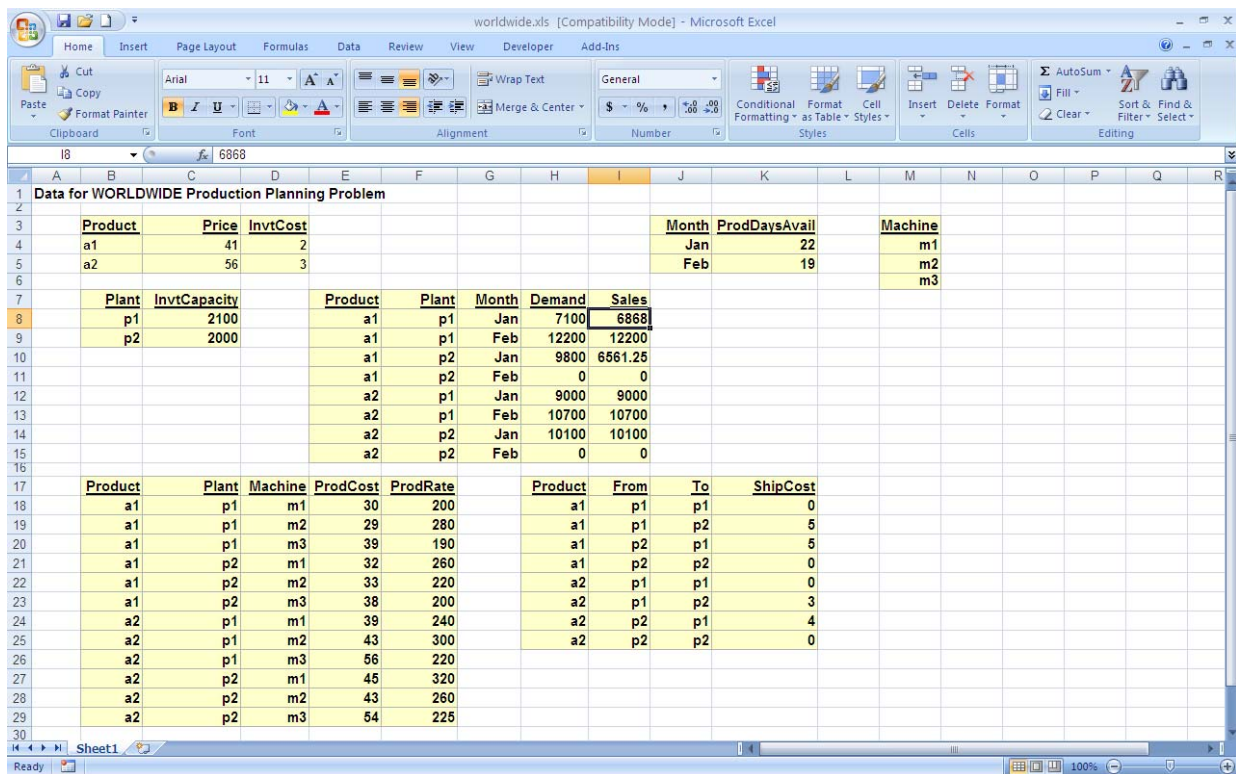

Figure 3S2.2 Data for WorldWide problem, including solution data, Sales.

If instead of a spreadsheet, you want to use a database, then you must replace each @OLE( ) by @ODBC(*database_name*), where *database_name* is the name of the database to be used. See Supplement 1 of chapter 3 for an example. ODBC(Open DataBase Connectivity) is a standard way of interfacing with databases that support SQL(Structured Query Language). Most popular database systems support ODBC/SQL.

10