

## Supplement to Appendix 3.1

### More About LINGO

Appendix 3.1 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. 3.7. To make this example more concrete, we will introduce actual data into a variation of the problem with a smaller number of products, months, and plants (two each) and of 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.

```
! Production Planning model in LINGO;
! Notice: model says nothing about number or names of:
  products, months, plants, machines, or
  shipping combinations. That information is
  determined completely by supplied data;
SETS:
!The simple sets;
  product: Price, InvtCost;
  month: ProdDaysAvail;
  plant: InvtCapacity;
  machine;;
!The derived sets;
  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;
  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
```

```
5 0
```

```
0 3
```

```
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: Beginning inventory + 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));
```

```
);
```

```

! Inventory for period 1, assumes start inventory = 0;
@FOR( PrPlMn( a, p, t) | t #EQ# 1:
    [PBall] @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;
@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 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:

```
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 given datum, while Inventory and Sales will be calculated as part of the optimization. The DATA section, however, need give no advance warning about which attributes are input data and which are variables to be determined.

### 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 have a Price and InvtCost of 41 and 2, whereas the latter has a price of 56 and 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.

### Model Equations

The first segment gives the objective function:

```

[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 in 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 the product, 'p' is the plant, and 't' is the month in a particular combination, we want 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 bullseye icon on the LINGO menu bar. You can get the solution displayed by clicking on the 'x=' menu item, giving:

Variable	Value	Reduced Cost
PRICE( A1)	41.00000	0.0000000
PRICE( A2)	56.00000	0.0000000
INVT COST( A1)	2.000000	0.0000000
INVT COST( A2)	3.000000	0.0000000
PRODDAYS AVAIL( JAN)	22.00000	0.0000000
PRODDAYS AVAIL( FEB)	19.00000	0.0000000
INVT CAPACITY( P1)	2100.000	0.0000000
INVT CAPACITY( P2)	2000.000	0.0000000
DEMAND( A1, P1, JAN)	7100.000	0.0000000
DEMAND( A1, P1, FEB)	12200.00	0.0000000
DEMAND( A1, P2, JAN)	9800.000	0.0000000
DEMAND( A1, P2, FEB)	0.0000000	0.0000000
DEMAND( A2, P1, JAN)	9000.000	0.0000000
DEMAND( A2, P1, FEB)	10700.00	0.0000000
DEMAND( A2, P2, JAN)	10100.00	0.0000000
DEMAND( A2, P2, FEB)	0.0000000	0.0000000
INVENTORY( A1, P1, JAN)	0.0000000	4.000000
INVENTORY( A1, P1, FEB)	0.0000000	41.00000
INVENTORY( A1, P2, JAN)	0.0000000	9.000000
INVENTORY( A1, P2, FEB)	0.0000000	36.00000
INVENTORY( A2, P1, JAN)	0.0000000	6.575001
INVENTORY( A2, P1, FEB)	0.0000000	53.62500
INVENTORY( A2, P2, JAN)	0.0000000	8.687500
INVENTORY( A2, P2, FEB)	0.0000000	49.62500
SALES( A1, P1, JAN)	6868.000	0.0000000
SALES( A1, P1, FEB)	12200.00	-2.000000
SALES( A1, P2, JAN)	6561.250	0.0000000
SALES( A1, P2, FEB)	0.0000000	-7.000000
SALES( A2, P1, JAN)	9000.000	-1.799999
SALES( A2, P1, FEB)	10700.00	-5.375000
SALES( A2, P2, JAN)	10100.00	-3.687500
SALES( A2, P2, FEB)	0.0000000	-9.375000
PRODCOST( A1, P1, M1)	30.00000	0.0000000

PRODCOST ( A1, P1, M2)	29.00000	0.0000000
PRODCOST ( A1, P1, M3)	39.00000	0.0000000
PRODCOST ( A1, P2, M1)	32.00000	0.0000000
PRODCOST ( A1, P2, M2)	33.00000	0.0000000
PRODCOST ( A1, P2, M3)	38.00000	0.0000000
PRODCOST ( A2, P1, M1)	39.00000	0.0000000
PRODCOST ( A2, P1, M2)	43.00000	0.0000000
PRODCOST ( A2, P1, M3)	56.00000	0.0000000
PRODCOST ( A2, P2, M1)	45.00000	0.0000000
PRODCOST ( A2, P2, M2)	43.00000	0.0000000
PRODCOST ( A2, P2, M3)	54.00000	0.0000000
PRODRATE ( A1, P1, M1)	200.0000	0.0000000
PRODRATE ( A1, P1, M2)	280.0000	0.0000000
PRODRATE ( A1, P1, M3)	190.0000	0.0000000
PRODRATE ( A1, P2, M1)	260.0000	0.0000000
PRODRATE ( A1, P2, M2)	220.0000	0.0000000
PRODRATE ( A1, P2, M3)	200.0000	0.0000000
PRODRATE ( A2, P1, M1)	240.0000	0.0000000
PRODRATE ( A2, P1, M2)	300.0000	0.0000000
PRODRATE ( A2, P1, M3)	220.0000	0.0000000
PRODRATE ( A2, P2, M1)	320.0000	0.0000000
PRODRATE ( A2, P2, M2)	260.0000	0.0000000
PRODRATE ( A2, P2, M3)	225.0000	0.0000000
SHIPCOST ( A1, P1, P1)	0.0000000	0.0000000
SHIPCOST ( A1, P1, P2)	5.000000	0.0000000
SHIPCOST ( A1, P2, P1)	5.000000	0.0000000
SHIPCOST ( A1, P2, P2)	0.0000000	0.0000000
SHIPCOST ( A2, P1, P1)	0.0000000	0.0000000
SHIPCOST ( A2, P1, P2)	3.000000	0.0000000
SHIPCOST ( A2, P2, P1)	4.000000	0.0000000
SHIPCOST ( A2, P2, P2)	0.0000000	0.0000000
PRODUCE ( A1, P1, M1, JAN)	0.0000000	7.240000
PRODUCE ( A1, P1, M1, FEB)	0.0000000	4.950000
PRODUCE ( A1, P1, M2, JAN)	2688.000	0.0000000
PRODUCE ( A1, P1, M2, FEB)	5320.000	0.0000000
PRODUCE ( A1, P1, M3, JAN)	4180.000	0.0000000
PRODUCE ( A1, P1, M3, FEB)	2915.000	0.0000000
PRODUCE ( A1, P2, M1, JAN)	2161.250	0.0000000
PRODUCE ( A1, P2, M1, FEB)	3965.000	0.0000000

PCAP( P1, M3, JAN)	0.0000000	380.0000
PCAP( P1, M3, FEB)	3.657895	0.0000000
PCAP( P2, M1, JAN)	0.0000000	2340.000
PCAP( P2, M1, FEB)	0.0000000	520.0000
PCAP( P2, M2, JAN)	0.0000000	2421.250
PCAP( P2, M2, FEB)	0.0000000	942.5000
PCAP( P2, M3, JAN)	0.0000000	600.0000
PCAP( P2, M3, FEB)	19.00000	0.0000000
PBAL( A1, P1, FEB)	0.0000000	-39.00000
PBAL( A1, P2, FEB)	0.0000000	-34.00000
PBAL( A2, P1, FEB)	0.0000000	-50.62500
PBAL( A2, P2, FEB)	0.0000000	-46.62500
PBAL1( A1, P1, JAN)	0.0000000	-41.00000
PBAL1( A1, P2, JAN)	0.0000000	-41.00000
PBAL1( A2, P1, JAN)	0.0000000	-54.20000
PBAL1( A2, P2, JAN)	0.0000000	-52.31250
MXINV( P1, JAN)	2100.000	0.0000000
MXINV( P1, FEB)	2100.000	0.0000000
MXINV( P2, JAN)	2000.000	0.0000000
MXINV( P2, FEB)	2000.000	0.0000000

## 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. The standard ways of checking or verifying a model are a)One way of debugging a small model is to look at the explicit constraints that get generated. You do this by clicking on the 'LINGO' menu item, and then on the 'Generate' option. The result is as follows.

```

MAX    56 SALES( A2, P2, FEB) - 3 INVENTORY( A2, P2, FEB)
+ 56 SALES( A2, P2, JAN) - 3 INVENTORY( A2, P2, JAN)
+ 56 SALES( A2, P1, FEB) - 3 INVENTORY( A2, P1, FEB)
- 56 SALES( A2, P1, JAN) - 3 INVENTORY( A2, P1, JAN)
- 41 SALES( A1, P2, FEB) - 2 INVENTORY( A1, P2, FEB)
+ 41 SALES( A1, P2, JAN) - 2 INVENTORY( A1, P2, JAN)
+ 41 SALES( A1, P1, FEB) - 2 INVENTORY( A1, P1, FEB)
+ 41 SALES( A1, P1, JAN) - 2 INVENTORY( A1, P1, JAN)
- 54 PRODUCE( A2, P2, M3, FEB) - 54 PRODUCE( A2, P2, M3, JAN)
- 43 PRODUCE( A2, P2, M2, FEB) - 43 PRODUCE( A2, P2, M2, JAN)
- 45 PRODUCE( A2, P2, M1, FEB) - 45 PRODUCE( A2, P2, M1, JAN)

```

- 56 PRODUCE( A2, P1, M3, FEB) - 56 PRODUCE( A2, P1, M3, JAN)  
 - 43 PRODUCE( A2, P1, M2, FEB) - 43 PRODUCE( A2, P1, M2, JAN)  
 - 39 PRODUCE( A2, P1, M1, FEB) - 39 PRODUCE( A2, P1, M1, JAN)  
 - 38 PRODUCE( A1, P2, M3, FEB) - 38 PRODUCE( A1, P2, M3, JAN)  
 - 33 PRODUCE( A1, P2, M2, FEB) - 33 PRODUCE( A1, P2, M2, JAN)  
 - 32 PRODUCE( A1, P2, M1, FEB) - 32 PRODUCE( A1, P2, M1, JAN)  
 - 39 PRODUCE( A1, P1, M3, FEB) - 39 PRODUCE( A1, P1, M3, JAN)  
 - 29 PRODUCE( A1, P1, M2, FEB) - 29 PRODUCE( A1, P1, M2, JAN)  
 - 30 PRODUCE( A1, P1, M1, FEB) - 30 PRODUCE( A1, P1, M1, JAN)  
 - 4 SHIP( A2, P2, P1, FEB) - 4 SHIP( A2, P2, P1, JAN)  
 - 3 SHIP( A2, P1, P2, FEB) - 3 SHIP( A2, P1, P2, JAN)  
 - 5 SHIP( A1, P2, P1, FEB) - 5 SHIP( A1, P2, P1, JAN)  
 - 5 SHIP( A1, P1, P2, FEB) - 5 SHIP( A1, P1, P2, JAN)

SUBJECT TO

PCAP( P1, M1, JAN)] .0041667 PRODUCE( A2, P1, M1, JAN)  
 + .005 PRODUCE( A1, P1, M1, JAN) <= 22  
 PCAP( P1, M1, FEB)] .0041667 PRODUCE( A2, P1, M1, FEB)  
 + .005 PRODUCE( A1, P1, M1, FEB) <= 19  
 PCAP( P1, M2, JAN)] .0033333 PRODUCE( A2, P1, M2, JAN)  
 + .0035714 PRODUCE( A1, P1, M2, JAN) <= 22  
 PCAP( P1, M2, FEB)] .0033333 PRODUCE( A2, P1, M2, FEB)  
 + .0035714 PRODUCE( A1, P1, M2, FEB) <= 19  
 PCAP( P1, M3, JAN)] .0045455 PRODUCE( A2, P1, M3, JAN)  
 + .0052632 PRODUCE( A1, P1, M3, JAN) <= 22  
 PCAP( P1, M3, FEB)] .0045455 PRODUCE( A2, P1, M3, FEB)  
 + .0052632 PRODUCE( A1, P1, M3, FEB) <= 19  
 PCAP( P2, M1, JAN)] .003125 PRODUCE( A2, P2, M1, JAN)  
 + .0038462 PRODUCE( A1, P2, M1, JAN) <= 22  
 PCAP( P2, M1, FEB)] .003125 PRODUCE( A2, P2, M1, FEB)  
 + .0038462 PRODUCE( A1, P2, M1, FEB) <= 19  
 PCAP( P2, M2, JAN)] .0038462 PRODUCE( A2, P2, M2, JAN)  
 + .0045455 PRODUCE( A1, P2, M2, JAN) <= 22  
 PCAP( P2, M2, FEB)] .0038462 PRODUCE( A2, P2, M2, FEB)  
 + .0045455 PRODUCE( A1, P2, M2, FEB) <= 19  
 PCAP( P2, M3, JAN)] .0044444 PRODUCE( A2, P2, M3, JAN)  
 + .005 PRODUCE( A1, P2, M3, JAN) <= 22  
 PCAP( P2, M3, FEB)] .0044444 PRODUCE( A2, P2, M3, FEB)  
 + .005 PRODUCE( A1, P2, M3, FEB) <= 19  
 PBAL( A1, P1, FEB)]- SALES( A1, P1, FEB) - INVENTORY( A1, P1, FEB)



```

END
SUB SALES( A2, P2, FEB)          0.000
SUB SALES( A2, P2, JAN)         10100.000
SUB SALES( A2, P1, FEB)         10700.000
SUB SALES( A2, P1, JAN)          9000.000
SUB SALES( A1, P2, FEB)          0.000
SUB SALES( A1, P2, JAN)          9800.000
SUB SALES( A1, P1, FEB)         12200.000
SUB SALES( A1, P1, JAN)          7100.000

```

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

Using method (b), check extreme cases, we can discover that our model has a "bug". 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, we must also modify the production balance constraints similarly, e.g., 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)).

```

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 called fredexms.xls. The spreadsheet looks as follows:

### Multi-product/plant/machine/month Production Planning Data

#### Product Data:

<i>product</i>	<i>Price</i>	<i>InvstCost</i>
a1	41	2
a2	56	3

#### Month Data:

<i>month</i>	<i>ProdDaysAvail</i>
Jan	22
Feb	19

#### Machine Data:

<i>machine</i>
m1
m2
m3

#### Plant Data:

<i>plant</i>	<i>InvstCapacity</i>
p1	2100
p2	2000

#### PrPIMa/Demand Data:

<i>product</i>	<i>plant</i>	<i>month</i>	<i>Demand</i>
a1	p1	Jan	7100
a1	p1	Feb	12200
a1	p2	Jan	9800
a1	p2	Feb	0
a2	p1	Jan	9000
a2	p1	Feb	10700
a2	p2	Jan	10100
a2	p2	Feb	0

#### PrPIMa Data:

<i>product</i>	<i>plant</i>	<i>machine</i>	<i>ProdCost</i>	<i>ProdRate</i>
a1	p1	m1	30	200
a1	p1	m2	29	280
a1	p1	m3	39	190
a1	p2	m1	32	260
a1	p2	m2	33	220
a1	p2	m3	38	200
a2	p1	m1	39	240
a2	p1	m2	43	300
a2	p1	m3	56	220
a2	p2	m1	45	320
a2	p2	m2	43	260

#### PrPIPI/Ship Cost Data:

<i>product</i>	<i>from</i>	<i>to</i>	<i>Shipcost</i>
a1	p1	p1	0
a1	p1	p2	5
a1	p2	p1	5
a1	p2	p2	0
a2	p1	p1	0
a2	p1	p2	3
a2	p2	p1	4
a2	p2	p2	0

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;
  product, Price, InvtCost = @OLE('d:\p123\fredexms.xls');
  month, ProcdDaysAvail = @OLE('d:\p123\fredexms.xls');
  plant, InvtCapacity = @OLE('d:\p123\fredexms.xls');
  machine = @OLE('d:\p123\fredexms.xls');
  Demand = @OLE('d:\p123\fredexms.xls');
  ProdCost = @OLE('d:\p123\fredexms.xls');
  ProdRate = @OLE('d:\p123\fredexms.xls');
  Shipcost = @OLE('d:\p123\fredexms.xls');
! Send results back to user's file(s);
@OLE('d:\p123\fredexms.xls') = PrPlMn;
@OLE('d:\p123\fredexms.xls') = Inventory;
@OLE('d:\p123\fredexms.xls') = Sales;
@OLE('d:\p123\fredexms.xls') = PrPlMaMn, Produce;
@OLE('d:\p123\fredexms.xls') = PrPlPlMn, Ship;
ENDDATA

```

The @OLE function provides the "plumbing" to hookup a spreadsheet to a LINGO model. OLE stands for "Object Linking and Embedding". The details of this hookup are as follows:

a) One must create range names in the spreadsheet for each data area that is to be either a source of data to or a recipient of data from the LINGO model. For simplicity, these range names should be the same as the attribute names in the LINGO model. Range names can be created in Excel by 1) highlight the range of interest with the mouse, and then 2) give it the desired name by using the Insert/Name/Define command.

b) In the LINGO model, any attribute to be retrieved from a spreadsheet must appear in the DATA section in a line like:

```
Demand = @OLE('d:\p123\fredexms.xls');
```

Each attribute to be sent back to a spreadsheet must appear on the other side of the equality sign, e.g., as in:

```
@OLE('d:\p123\fredexms.xls') = Sales;
```

The spreadsheet, after the model has been solved, looks as follows:

Results:

PrPIMa:

product	plant	month	Inventory	Sales
A1	P1	JAN	0	6868
A1	P1	FEB	0	12200
A1	P2	JAN	0	6561.25
A1	P2	FEB	0	0
A2	P1	JAN	0	9000
A2	P1	FEB	0	10700
A2	P2	JAN	0	10100
A2	P2	FEB	0	0

PrPIPIMn

product	from	to	month	Ship
A1	P1	P1	JAN	0
A1	P1	P1	FEB	0
A1	P1	P2	JAN	0
A1	P1	P2	FEB	0
A1	P2	P1	JAN	0
A1	P2	P1	FEB	3965
A1	P2	P2	JAN	0
A1	P2	P2	FEB	0
A2	P1	P1	JAN	0
A2	P1	P1	FEB	0
A2	P1	P2	JAN	0
A2	P1	P2	FEB	0

A2	P2	P1	JAN	0
A2	P2	P1	FEB	6140
A2	P2	P2	JAN	0
A2	P2	P2	FEB	0

PrPIMaMn

product		plant	machine	month	Produce
A1	P1		M1	JAN	0
A1	P1		M1	FEB	0
A1	P1		M2	JAN	2688
A1	P1		M2	FEB	5320
A1	P1		M3	JAN	4180
A1	P1		M3	FEB	2915
A1	P2		M1	JAN	2161.25
A1	P2		M1	FEB	3965
A1	P2		M2	JAN	0
A1	P2		M2	FEB	0
A1	P2		M3	JAN	4400
A1	P2		M3	FEB	0
A2	P1		M1	JAN	5280
A2	P1		M1	FEB	4560
A2	P1		M2	JAN	3720
A2	P1		M2	FEB	0
A2	P1		M3	JAN	0
A2	P1		M3	FEB	0
A2	P2		M1	JAN	4380
A2	P2		M1	FEB	1200
A2	P2		M2	JAN	5720
	P2		M2	FEB	4940
A2	P2		M3	JAN	0
A2	P2		M3	FEB	0

If instead of using a spreadsheet, you want to use a standard database system, such as Microsoft Access, then you must replace the @OLE above by @ODBC, where ODBC stands for Open DataBase Connectivity.