Introduction to GAMS

(General Algebraic Modeling System)

Tevy Chawwa - 2013

Download the GAMS

- Detailed Information in:

  Tutorial GAMS :

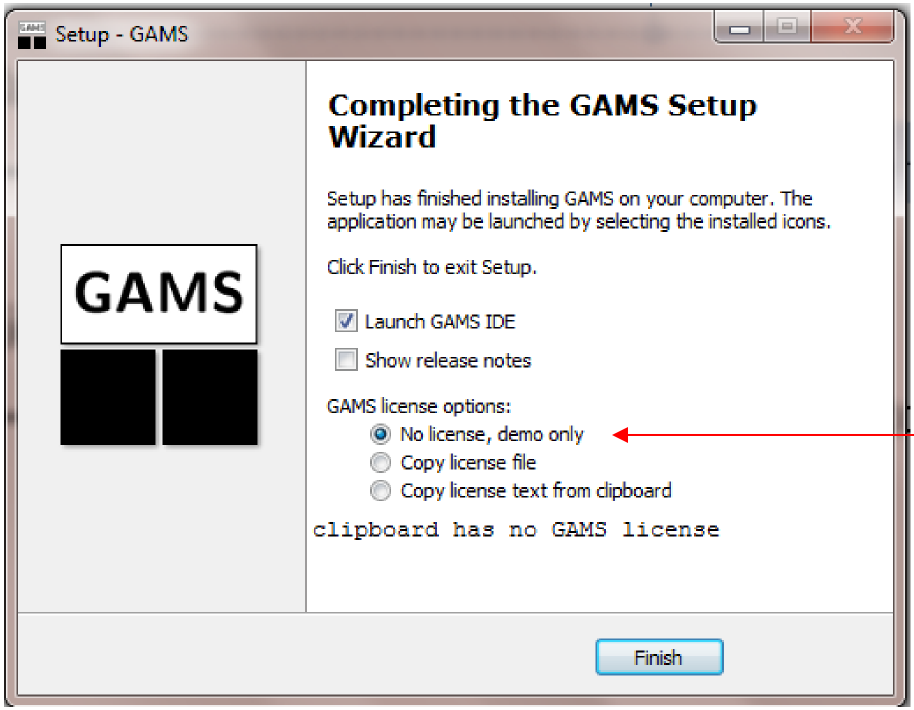
  – http://www.gams.com/dd/docs/gams/Tutorial.pdf
  – http://www.gams.com/dd/docs/bigdocs/GAMSUsersGuide.pdf

- Download the GAMS :

  - Go to http://www.gams.com/
  - You will find the latest version under : Download Current GAMS System'.
  - Choose the compatible version based on your computer system (Windows 32 bit/64 bit, Mac, etc)

Installation

- Run the downloaded file for setup: windows_x64_64(1)
- License file
  - Choose ‘No’ when asked if you wish to copy a license file



Limitation

- **Limitation of free demo version :**
  - Number of constraints and variables: 300
  - Number of nonzero elements: 2000 (of which 1000 nonlinear)
  - Number of discrete variables: 50 (including semi continuous, semi integer and member of SOS-Sets)
  - Additional Global solver limits: Number of constraints and variables: 10
  - The GAMS log will indicate that your system runs in demo mode:

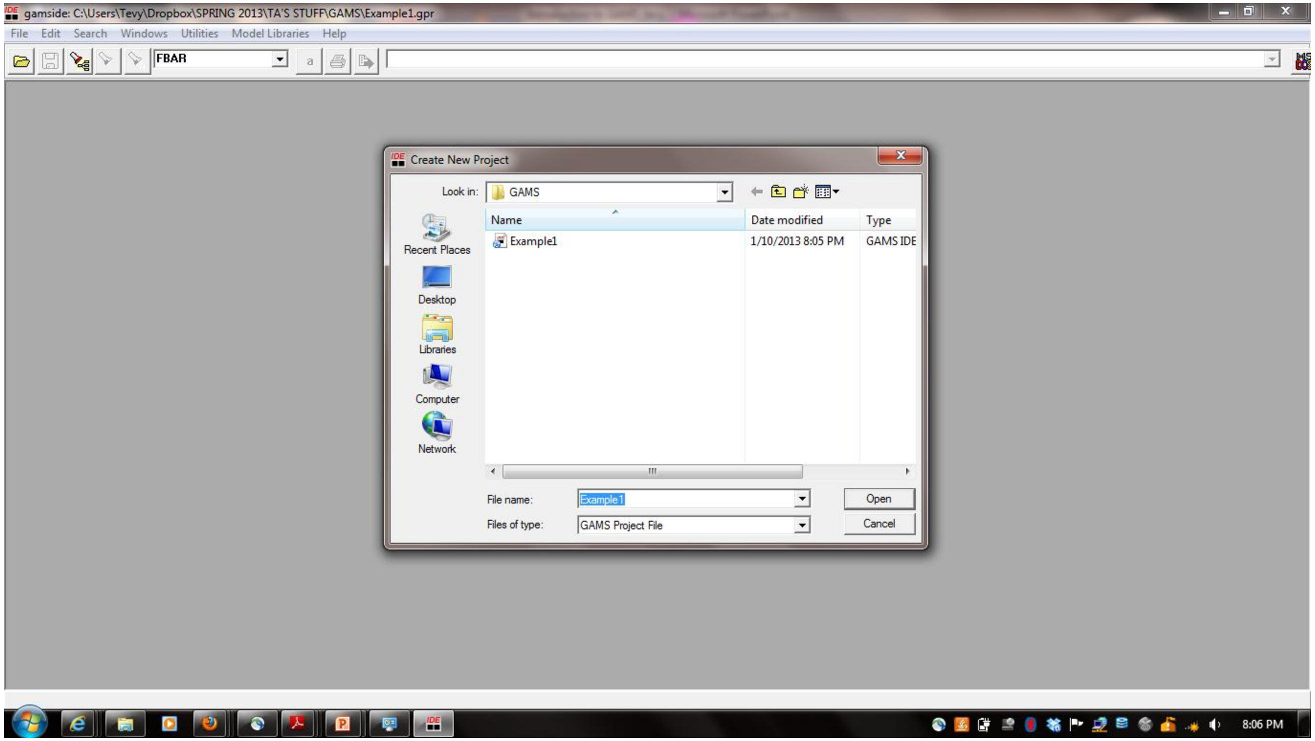
    GAMS Rev 240 Copyright (C) 1987-2012 GAMS Development. All rights reserved Licensee: GAMS Development Corporation, Washington, DC G871201/0000CA-ANY Free Demo, 202-342-0180, sales@gams.com, www.gams.com DC0000
  - GAMS will terminate with a licensing error if you hit one of the limits above:

    *** Status: Terminated due to a licensing error

    ***   Inspect listing file for more information

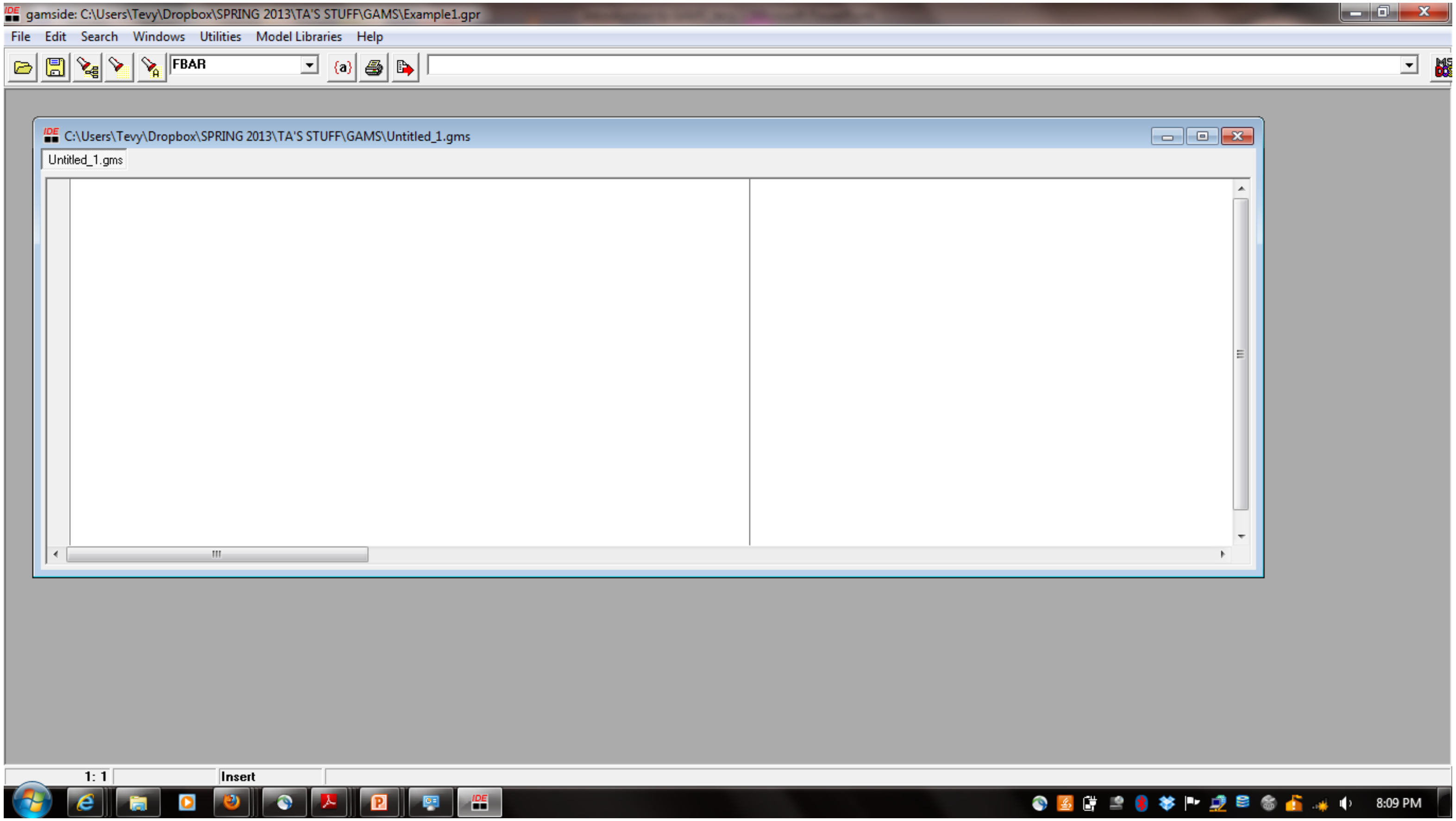
Start GAMS

- ***File → Project → New project***
- ***Specify the name of project and the folder : example1.gpr***
- The GAMS window should now show the example1.gpr project window



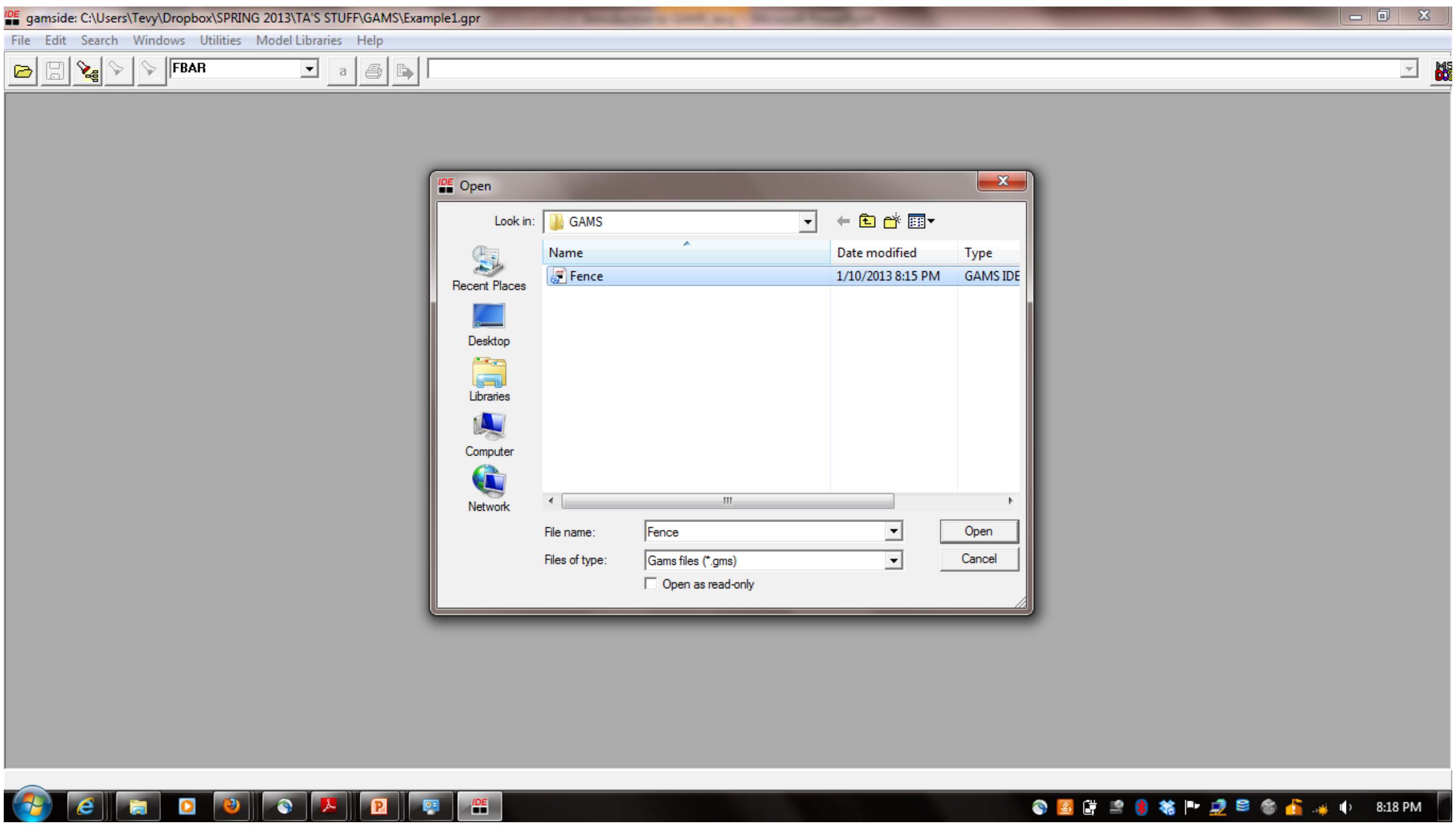
Create New GAMS Code File

- Select: ***File → New***
- You should see the new file ***"Untitled_1.gms"***



Open GAMS Code File

- Select: ***File → Open*** → choose the .gms file
- Example : fence. gms



Structure of GAMS Code

Inputs :

- Sets
  - Declaration
  - Assignment of members
- Data (Parameters, Tables, Scalars)
  - Declaration
  - Assignment of values
- Variables
  - Declaration
  - Assignment of type
- Assignment of bounds and/or initial values (optional)
- Equations
  - Declaration
  - Definition
- Model and Solve statements
- Display statement (optional)

Output :

- Echo Print
- Reference Maps
- Equation Listings
- Status Reports
- Results

Some basic rules

1. Ordering of statements : an entity of the model cannot be referenced before it is declared to exist.
2. GAMS statements may be laid out typographically in almost any style that is appealing to the user :
   - Multiple lines per statement, embedded blank lines, and multiple statements per line are allowed
3. You should terminate every statement with a semicolon
4. The GAMS compiler does not distinguish between upper- and lowercase letters, so you are free to use either.
5. Documentation is crucial to the usefulness of mathematical models. There are 2 ways to do this :
   - starts with an asterisk in column 1 is disregarded as a comment line
   - documentary text can be inserted within specific GAMS statements

Some basic rules

6. The creation of GAMS entities involves two steps: a declaration and an assignment or definition.
   – Declaration: declaring the existence of something and giving it a name.
   – Assignment or definition : giving something a specific value or form.

- In the case of equations, you must make the declaration and definition in separate GAMS statements.
- For all other GAMS entities, however, you have the option of making declarations and assignments in the same statement or separately.

7. The names given to the entities of the model must start with a letter and can be followed by up to thirty more letters or digits.

Sets

<u>Example 1 :</u>
There are two goods I ={1,2}
We can write it as :

```
SET I Goods /1,2/ ;
```

<u>Example 2:</u>

- i = {Seattle, San Diego}
- j = {New York, Chicago, Topeka}

We can write it as:

```
Sets
       i plants / seattle, san-diego /
       j markets / new-york, chicago, topeka / ;
```

Or

```
      set i plants / seattle, san-diego / ;
      set j markets / new-york, chicago, topeka / ;
```

Sets

Example 3:

- t = {1991,1992,1993, ..., 2000}
- m = {mach1, mach2,......, mach24}.

We can write as :

- `Set t time periods /1991*2000/;`
- `Set m machines /mach1*mach24/;`

→ Note that set elements are stored as character strings, so the elements of t are not numbers.

- Another convenient feature is the alias statement, which is used to give another name to a previously declared set. In the following example:

```
Alias (t,tp);
```

- the name tp is like a t' in mathematical notation. It is useful in models that are concerned with the interactions of elements within the same set.

Data

- There are three formats of data: Lists, Table and Direct Assignments

1. Lists

```
Parameters
    a(i) capacity of plant i in cases
    / seattle 350
    san-diego 600 /
    b(j) demand at market j in cases
    / new-york 325
    chicago 300
    topeka 275 / ;
```

Rules :

- the entire list must be enclosed in slashes
- the element-value pairs must be separated by commas or entered on separate lines.
- There is no semicolon separating the element-value list from the name, domain, and text that precede it.
- Zero is the default value for all parameters.
- A scalar is regarded as a parameter that has no domain. It can be declared and assigned with a Scalar
- statement containing a degenerate list of only one value, as in the following statement from the
- transportation model. Example :

```
Scalar f freight in dollars per case per thousand miles /90/;
```

Data

2. Table

```
table d(i,j) distance in thousands of miles
           new-york   chicago    topeka
seattle       2.5       1.7        1.8
san-diego     2.5       1.8        1.4 ;
```

- declare the parameter d and specify its domain as the set of ordered pairs in the Cartesian product of i and j. The values of d are also given in this statement under the appropriate heading.
- If there are blank entries in the table, they are interpreted as zeroes.

Data

3. Direct Assignments

```
parameter c(i,j) transport cost in thousands of dollars
per case ;
c(i,j) = f * d(i,j) / 1000 ;
```

Other example :

```
Y Income
P(I) Prices of goods;
P(I)=1;
Y=100;
```

- The same parameter can be assigned a value more than once. Each assignment statement takes effect immediately and overrides any previous values

Variables

- The decision variables (or endogenous variables) of a GAMS-expressed model must be declared with a Variables statement.
- Each variable is given a name, a domain if appropriate, and (optionally) text.
- Examples :

```
VARIABLES
U        Utility level
P(I)     Prices
C(I)     Consumption levels;
```

Or

```
Variables
x(i,j)  shipment quantities in cases
z total transportation costs in thousands of dollars ;
```

Equations

- Equations must be declared and defined in separate statements.
- <u>Declaration :</u>

```
Equations
  cost       define objective function
  supply(i)  observe supply limit at plant i
  demand(j)  satisfy demand at market j ;
```

<u>Definition</u>

Component of definition :

1. The name of the equation being defined
2. The domain
3. Domain restriction condition (optional)
4. The symbol '..'
5. Left-hand-side expression
6. Relational operator: =l=, =e=, or =g=
   (less than or equal to, equal to, greater than or equal to)
7. Right-hand-side expression

Equations

```
Example 1:
cost .. z =e= sum((i,j), c(i,j)*x(i,j)) ;
supply(i) .. sum(j, x(i,j)) =l= a(i) ;
demand(j) .. sum(i, x(i,j)) =g= b(j) ;


Example 2:
UTILITY..U=E=ALPHA*PROD(I, C(I)**BETA(I));
DEMAND(I)..C(I)=E=BETA(I)*Y/P(I);

Remember :
```

- The '=' symbol is used only in direct assignments, and the '=e=' symbol is used only in equation definitions.

Values for variables and equations

- GAMS was designed with a small database system in which records are maintained for the variables and equations.
- The most important fields in each record are:

  .lo lower bound

  .l level or primal value

  .up upper bound

  .m marginal or dual value

- Example : initial value of variables :

```
U.L=UO;
C.L(I)=CO(I);
C.LO(I)=0;
```

Model and Solve Statements

- Model means collection of equations
- If all previously dened equations are to be included you can enter /all/ in place of the explicit list.

```
model transport /all/ ;
```

- If we were to use some equations only :

```
model transport / cost, supply / ;
```

The domains are omitted from the list since they are not part of the equation name

- Once a model has been declared and assigned equations, we are ready to call the solver. This is done with a solve statement, which in our example is written as

```
solve transport using lp minimizing z ;
```

Model and Solve Statements

The format of the solve statement is as follows:

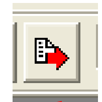
1. The key word solve
2. The name of the model to be solved
3. The key word using
4. An available solution procedure. The complete list is
   - lp for linear programming
   - qcp for quadratic constraint programming
   - nlp for nonlinear programming
   - dnlp for nonlinear programming with discontinuous derivatives
   - mip for mixed integer programming
   - rmip for relaxed mixed integer programming
   - miqcp for mixed integer quadratic constraint programming
   - minlp for mixed integer nonlinear programming
   - rmiqcp for relaxed mixed integer quadratic constraint programming
   - rminlp for relaxed mixed integer nonlinear programming
   - mcp for mixed complementarity problems
   - mpec for mathematical programs with equilibrium constraints
   - cns for constrained nonlinear systems
5. The keyword 'minimizing' or 'maximizing'
6. The name of the variable to be optimized

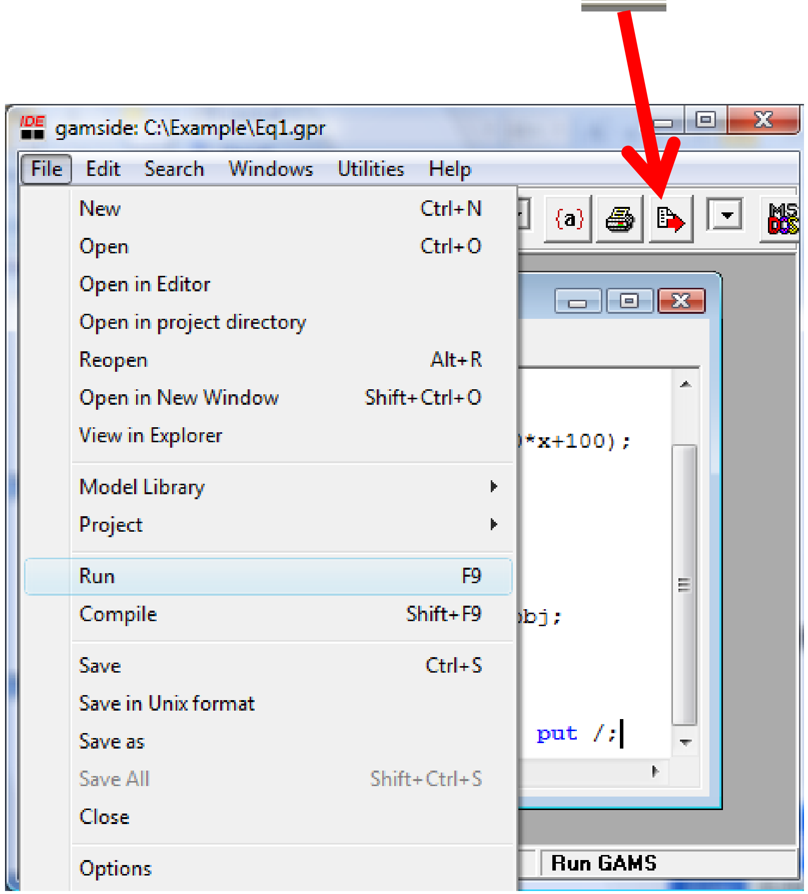
Display

- We can request a display of the results from GAMS.

```
Display U,P  ;
```

Run the Model

- Select: ***File → Run***, or Press the ***red arrow***  ***button***



Output

- Echo prints : copy, of your input file
- Error messages :

  These messages always start with **** and contain a '$' directly below the point at which the compiler thinks the error occurred. The $ is followed by a numerical error code, which is explained after the echo print.
- Reference Maps : pair of reference maps that contain summaries and analyses of the input le for the purposes of debugging and documentation.
- Equation Listings
- Model Statistics : number of equations, variables etc
- Status Reports

  The desired solver status is 1 NORMAL COMPLETION

  Model status can be : 1. OPTIMAL, 2. LOCALLY OPTIMAL (For NLP),

  3. UNBOUNDED, 4 INFEASIBLE
- Solution Reports

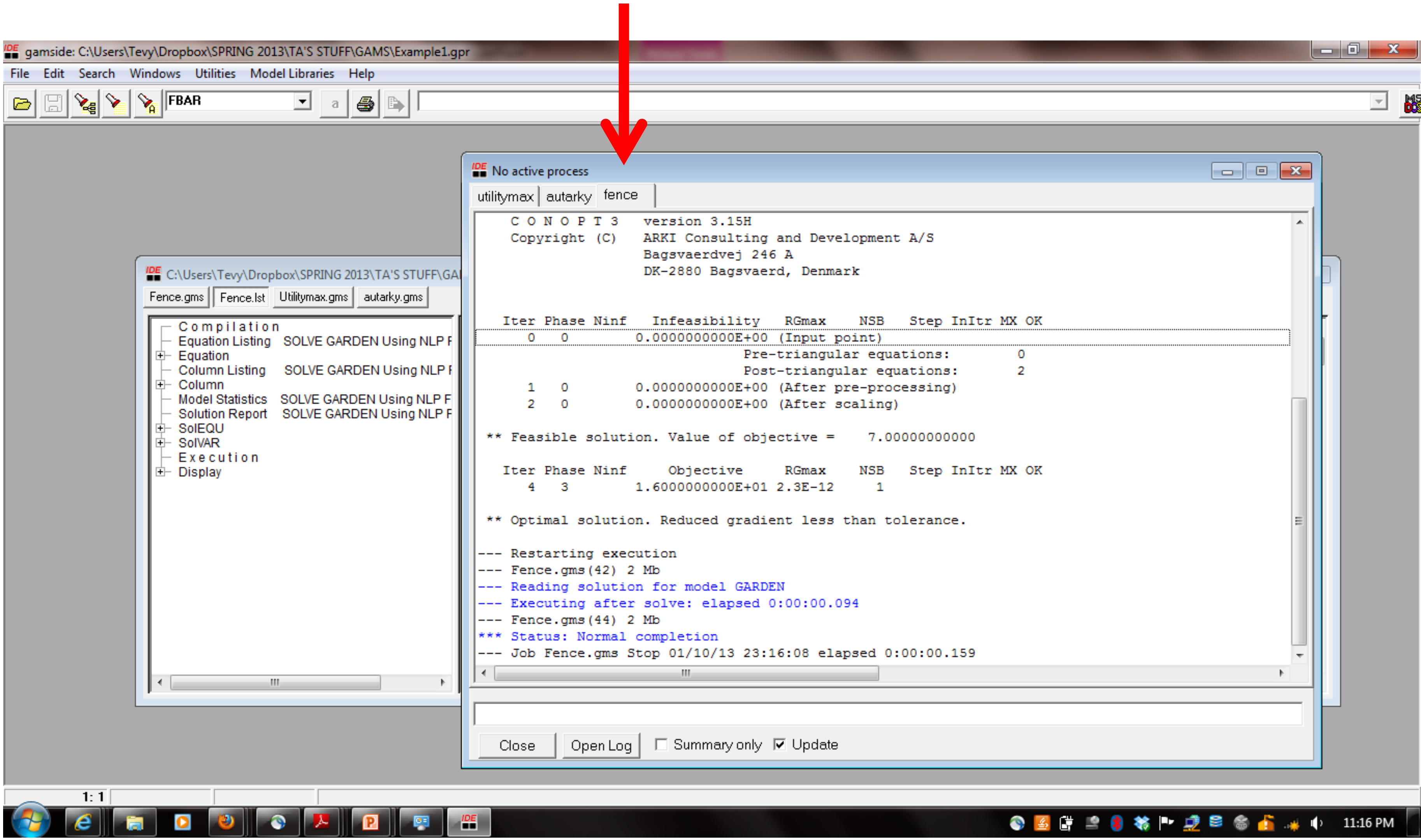
Simple Example of Problem : Fence

```
*This is a program called garden.
*We have 16 feet of fencing.
*Our garden is initially 1 foot by 7 feet.
*We want to find the dimensions that maximize area of the
garden.

*create names for parameters
PARAMETERS
A0 Initial area
L0 Initial length
W0 Initial width;

*Assign values to the parameters
W0=1;
L0=7;
A0=7;
```

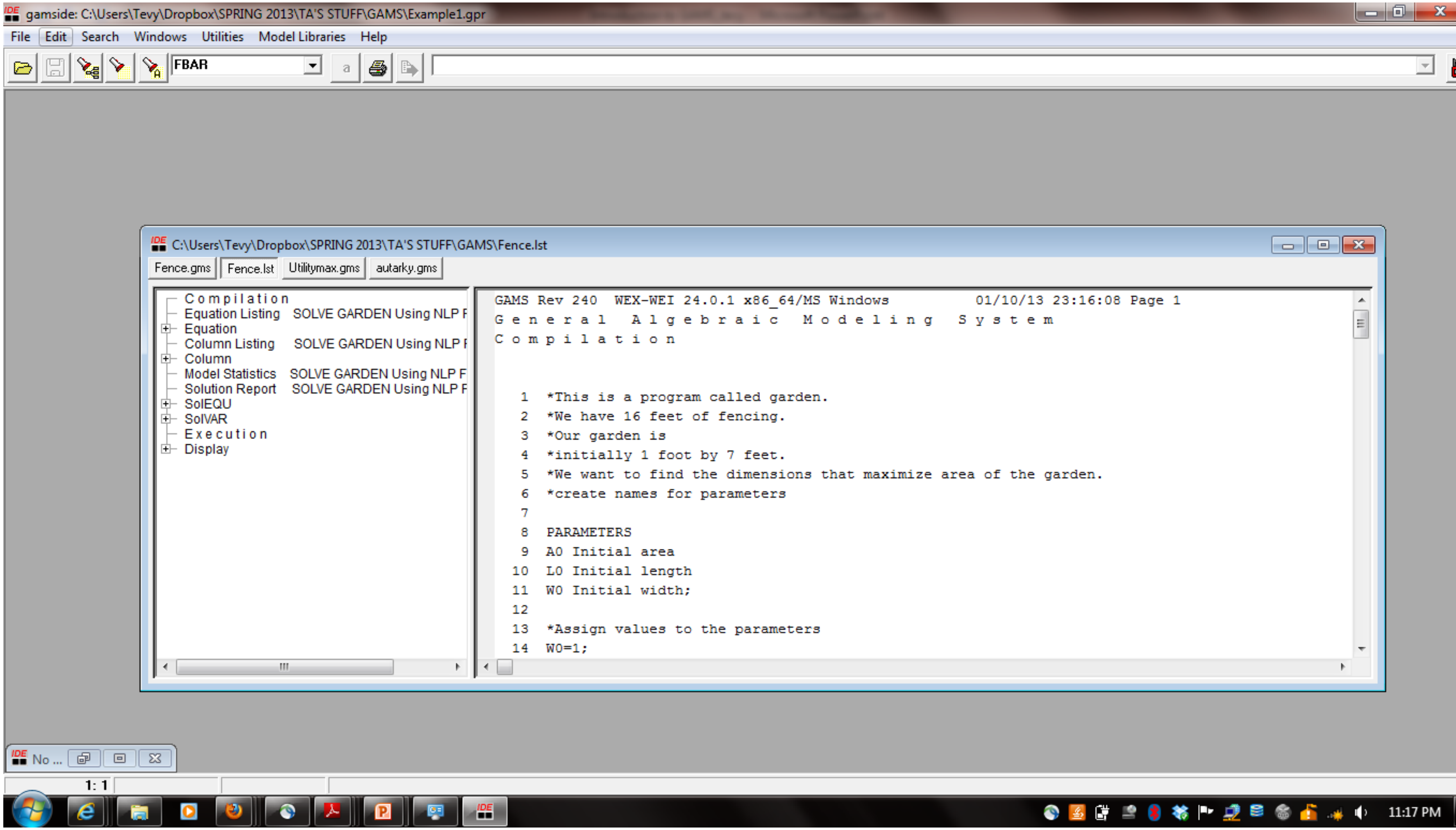
```
*Create names for variables

VARIABLES
A Area
L Length
W Width;

*Create names for equations
Equations

Area,
Fence;

* Assign the expressions to the equation names
Area..A=E=L*W;
Fence..2*L+2*W=E=16;

*Assign initial values to variables
L.L=L0;
W.L=W0;
A.L=A0;

Model GARDEN/ALL/;
SOLVE GARDEN USING NLP MAXIMIZING A;
display L.L, W.L, A.L;
```

GAMS Model Results

- "No active process" window



- “fence.lst” window



Solution

```
**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
                             0     ERRORS
GAMS Rev 240  WEX-WEI 24.0.1 x86_64/MS Windows          01/10/13 23:16:08
Page 6
G e n e r a l   A l g e b r a i c   M o d e l i n g   S y s t e m
E x e c u t i o n


----     44 VARIABLE L.L                  =        4.000  Length
            VARIABLE W.L                  =        4.000  Width
            VARIABLE A.L                  =       16.000  Area


EXECUTION TIME       =        0.063 SECONDS      2 Mb  WEX240-240 Dec 18,
2012
```