Structure and Performance of Six European Wholesale Electricity Markets in 2003, 2004 and 2005

Appendix II

DG Comp

Presented to DG Comp
26th February 2007

Global Energy Decisions

Scripts and Procedures to Compute and Report Marginal Average Cost

Version 1.0

January 2007

Table of Contents

Figures

1. OVERVIEW

1.1. Conceptual Overview

DG Competition's study of the structure and functioning of the EU's major countries' electricity markets uses as a benchmark the marginal average cost (MAC) of delivering electricity in each country during each hour of the study period, that would be observed during a reasonably efficient dispatch to meet loads, as produced by a PROSYM cost-based simulation in the MARKETSYM electricity-market simulation system. The MAC measure involves identifying the most-expensive unit, in terms of average generation cost, excluding those units whose generation is constrained; the MAC is the average generation cost, in Euros/MWh, of the most expensive unit. Under conditions of workable competition, electricity prices will exceed the MAC by enough to cover startup costs and the costs of operating stations at minimum load—the adjusted MAC. Prices substantially in excess of this adjusted MAC should, absent significant barriers to entry, encourage entry (in the short term through the commitment of additional generating stations, in the long term through investment in additional physical generation and/or transmission capacity); if prices do not cover the adjusted MAC, profit-maximizing generators under conditions of workable competition will tend to exit, in the short term by decommitting stations, in the long-term by retiring generating capacity from service.

Global Energy's MARKETSYM electricity-market simulation system simulates the efficient commitment and dispatch that would occur under conditions of perfect competition by finding a reasonable (although not, in general, perfect) solution to the problem of minimizing the cost of serving loads, subject to a variety of physical and chronological constraints, using the PROSYM simulation engine. In common with the general optimization solution, the PROSYM solution computes a shadow price of loads, often known as the System Lambda, which describes the additional cost to the system of serving an additional MW of load.

In industries with relatively high fixed costs, and relatively low barriers to entry and exit, an "industry marginal cost" cannot generally be identified with a market price. The reason is that the marginal unit cannot cover its fixed costs, and if it could have avoided participating in the money-losing transaction it would have. This is precisely the obstacle to using the simple System Lambda from a PROSYM run as a benchmark price measure. The "investment decision" to participate in an hour's market is simply the cost to commit (or the cost to decommit), and it is generally easy for a generator to leave (or enter) this market; as a result one does not expect to see prices equal to the short-run system marginal cost. This has not prevented the use of the PROSYM systems from being used to support price forecasts. The usual approach is to identify the markups to bids that are necessary to allow different types of units (in particular low-capacity-factor peaking stations, and the high-cost peaking segments of mid-merit and base-loaded generation) to cover all of the costs of commitment and dispatch, and make a reasonable contribution to fixed costs. In the usual practice, this approach may be understood as a monotonic transformation of the underlying industry incremental cost curve, with the low-cost elements of segments of the cost curve, corresponding to base-loaded generation, marked up little if at all, but with markups increasing as one moves up the curve, so that units or segments that run very rarely are able to recover their non-incremental costs in the small number of hours that they

actually run. Since the resulting bid-augmented stack is a monotonic transformation of the system incremental costs, and since electricity is modelled in the short run as having inelastic demand, the resulting dispatch is efficient. Indeed, a PROSYM dispatch that includes carefully-calibrated bid markups will be identical to a pure cost-based markup, albeit with simulated system lambdas that can be used to forecast prices.

A drawback of this approach, however, is that it requires that the analyst decide what a "reasonable contribution" is, and to otherwise introduce values into the model that are not directly derived from more-or-less objective data. This makes the conventional price-forecasting approach unsuitable for use in regulatory and investigative contexts, where the objective is to identify problems in industrial structure and conduct. The MAC approach avoids this drawback by eliminating the subjective bid-markup step, and instead simply identifying the costs that would need to be covered by market prices in an efficient dispatch, which may then be compared to the wholesale prices for electricity that were actually observed.

The MAC approach does set as a benchmark the average cost of the most expensive station serving loads in an hour's efficient dispatch. If in actual practice the most expensive unit operating would be excluded from the efficient dispatch, then a price that just covers the inefficient unit(s) cost would produce economic profits in the efficient dispatch. This is a feature, not a weakness, of the MAC approach: a high-price solution necessarily involves loss of consumer surplus; in an efficient dispatch some of this loss of consumer surplus is captured by producers, while in an inefficient dispatch there is pure deadweight loss. While the MAC approach does not distinguish between the outcomes of an efficient monopolist and an inefficient dispatch, it correctly captures the inefficient dispatch, which is in general a strong indicator of markets that are not functioning well—typically because of barriers to entry and other obstacles to competition that interfere with the application of market discipline.

1.2. Implementing MAC in MARKETSYM: Overview

To implement the Marginal Average Cost approach in MARKETSYM, the user simulates the cost-minimizing commitment and dispatch to serve loads. Hourly values for each station in the model are reported for generation and generation cost. In hours when a station starts up, start costs (if any) are included in the reported generation cost. These costs are computed, taking into consideration as appropriate start cost profiles which distinguish between hot, warm, and cold starts. Stations that are either excluded from the analysis by prior specification (primarily Combined Heat and Power stations and stations that are specified as Must Run to force out-of-merit operations), or that have simulated generating levels at or below their specified minimums are identified and reported as having a zero average cost. For the remaining stations, computed start costs if any are subtracted from the reported generation cost, with the result divided by non-zero generation to arrive at the average cost that the generator must cover in that hour. The station(s) with the highest average costs in an hour are identified as marginal; their average costs are the Marginal Average Costs. Finally, in hours in which multiple stations share the same marginal position, duplicates are eliminated to provide a single Marginal Average Cost for use in subsequent analyses.

2. The MAC approach in MARKETSYM: Detailed Implementation

2.1. *Hourly Output setup*

As noted above, the PROSYM simulation engine identifies a reasonably efficient commitment and dispatch. A number of outputs are available to describe the dispatch. These are in general hourly variables, since each hour's dispatch, while not independent of the surrounding hours' dispatches, is at least distinct from the dispatch in every other hour. PROSYM (and its MARKETSYM shell) offer the user considerable flexibility in the outputs that are reported. However, hourly reporting carries a substantial cost in terms of processing time; the best processing times are achieved by restricting the hourly reporting to the minimum necessary for the task at hand. To select hourly variables for output in MARKETSYM, select *Output Settings→Hourly Variables* from the Menu, double-click on the Hourly-Variables entity to be included in the run (for example, *DG Comp MCP, Gen, incl avg cost*), then use the interface to select (at a minimum) the *Sta.Generation* and *Sta.Gen.Cost* variables. See Figure 1 below.

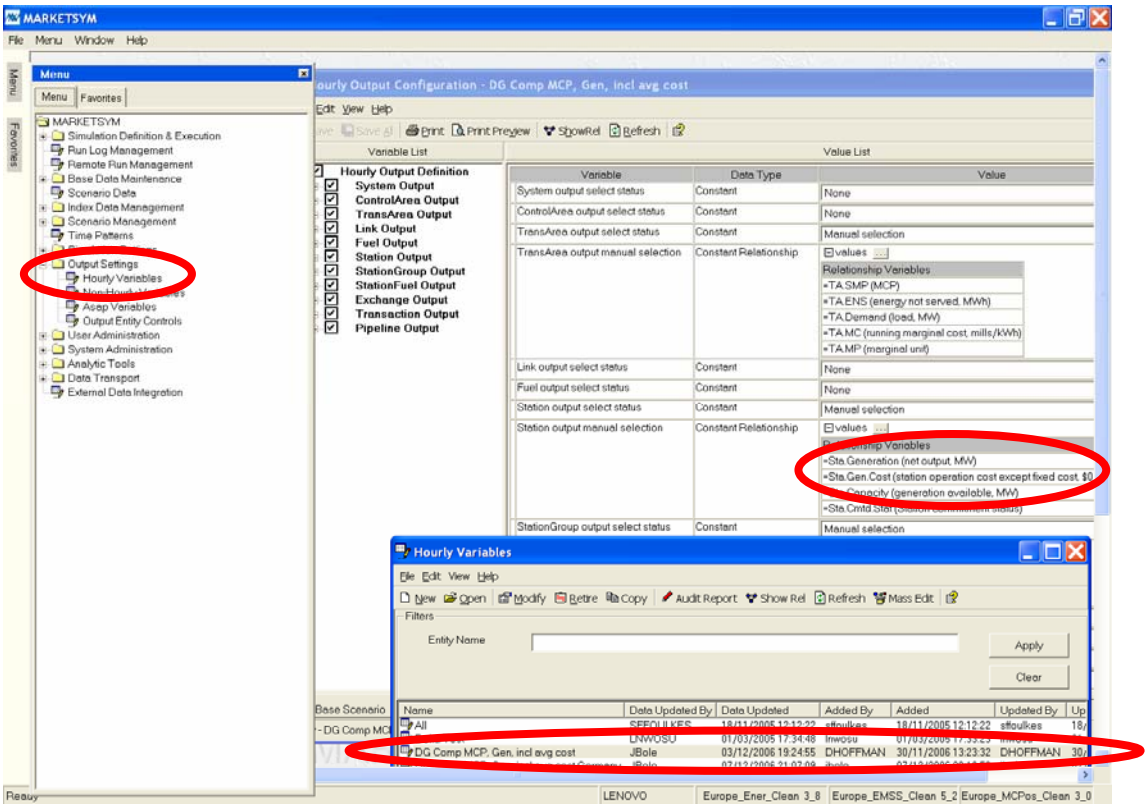

Figure 1. Hourly Output Variable Selection

Once the appropriate hourly output variables are specified, a Base Run may be run for a country. In the MARKETSYM system a file with the extension ***.dta*** is produced, which is then loaded into an Access database with the *Loader2K* program, delivered as part of the MARKETSYM installation. This database may then be linked to the *Reporter2K* database shell, also provided as part of the MARKETSYM installation. These steps, which are at the core of any MARKETSYM workflow, are not discussed further in this documentation of the *Marginal Average Cost* procedure.

2.2. Reporter2K Setup

To implement the *Marginal Average Cost* procedure in MARKETSYM, a specialized but very simple query must be added to the set of queries in the Reporter2K shell. In the copy of the Reporter2K delivered to DG Comp, this query is named *MAC_HourlyGenandCost*; it is shown in Figure 2 below in the Access design view.

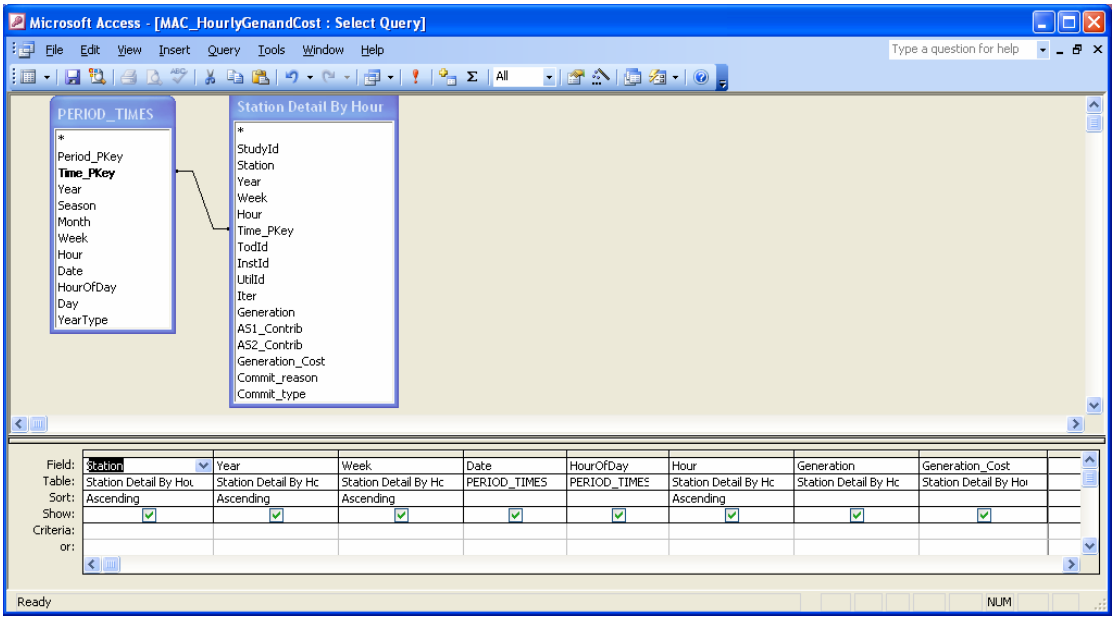

Figure 2. Hourly Generation Cost query from Reporter2K

While the design view shown above provides an intuitive interface to build and understand database queries, a straightforward approach to copying the queries from one setup to another is to simply copy the SQL code between the SQL View of the query; this also provides an unambiguous archive of the query design. The MAC_HourlyGenandCost query uses the SQL code

```
SELECT [Station Detail By Hour].Station, [Station Detail By Hour].Year,
[Station Detail By Hour].Week, PERIOD_TIMES.Date, PERIOD_TIMES.HourOfDay,
[Station Detail By Hour].Hour, [Station Detail By Hour].Generation,
[Station Detail By Hour].Generation_Cost
FROM PERIOD_TIMES INNER JOIN [Station Detail By Hour] ON
PERIOD_TIMES.Time_PKey = [Station Detail By Hour].Time_PKey
ORDER BY [Station Detail By Hour].Station, [Station Detail By Hour].Year, [Station Detail
By Hour].Week, [Station Detail By Hour].Hour;
```

2.3. Standing Data setup

The *MAC.pl* script requires, in addition to the results from completed simulation runs, some standing data to allow it to correctly process the results. First, it requires data on start costs, because it will subtract these costs from the generating costs in the first hour that a station runs after a period of non-operation. Second, it requires each station's Minimum Stable Generation (MSG), because when a station is running at MSG it is excluded from the computation of the marginal average cost. These two pieces of information are extracted from a file named *thermal.dat*. In addition, the *MAC.pl* script reads a file, named *chp.csv*; containing schedules of stations to be excluded from the computation of marginal average cost. Both of these files must be present in the run directory specified in the *MAC.pl* command (see section 2.5.3 below).

2.3.1. *thermal.dat*: Start Costs and Minimum Generation levels

The *MAC.pl* script is designed to take advantage of the same information on start costs and minimum stable generation that are used in the *PROSYM* simulation run. *MARKETSYM* extracts this information from the EMSS database (along with considerable additional information in the technical characteristics of power stations) and writes it into a *thermal.dat* file which is then read by the *PROSYM* simulation engine. The *thermal.dat* file contains one or more entries for each station, with the entry for each station initiated with a line with the station's *PROSYM* name starting in the first column; the remaining information in the station's entry are on subsequent lines which begin with one or more spaces.

In the default *MARKETSYM* configuration, each *PROSYM* variable for a station is written on a line that starts with the variable name (with optional continuation lines for variables with lengthy data entries). This arrangement is not, however, required by *PROSYM*, which does accept input files with multiple variables on one line. *MAC.pl* is written to use a default *MARKETSYM* configuration; modifications to a *thermal.dat*, performed either through manual editing of the file, or through the use of custom *PROSYM* script entered outside the interface for each variable, may not be interpreted correctly by the script; *thermal.dat* records that do not conform to *MARKETSYM* conventions should be deleted from the copy of *thermal.dat* that is read by *MAC.pl.*

The copy of *thermal.dat* that is provided to the *MAC.pl* script need not be complete. All that it must have is the names of the stations, along with their start costs and minimum generation levels. A script is available (*StripThermal.pl*) which filters just this information from a full *thermal.dat* file.

2.3.1.1. Start Costs

MARKETSYM supports two specifications of start costs: a simple single-point start cost, and a multi-point profile to describe hot, warm, and cold starts. This approach may be applied to both monetary start costs and to start fuel, which is then valued at the relevant fuel price. *MAC.pl* supports both single and multiple point start cost specifications, but only monetary costs—start-fuel purchases are not evaluated. In both specifications the *PROSYM* variable StartCost carries monetary start costs; in the multiple point specification this variable is a vector with at least two points, and a matching vector-valued StartHours variable identifies the number of off-line hours that matches each of the StartCost values. An example of a single-point start cost is as follows:

```
StartCost       20000.0
```

An example of a multiple-point specification is as follows:

```
StartHours      [V2]    8       48
StartCost       [V2]    12400.0  64010.0
```

In the multiple-point example, if the station has been off for 8 hours or less when it restarts, it will incur hot-start costs of €12,400. If the station has been off for 48 hour or more, it will incur cold-start costs of €64,010. If the station has been off for between 8 and 48 hours, it will incur warm-start costs found by interpolating linearly between the hot-start and warm-start costs. *MAC.pl* performs this interpolation when vector values of StartHours and StartCost are found, using information from the simulation run on the number of hours that the station has been off-line. While in the initial implementation of *MAC.pl* only two-point profiles are included, the script has been written to handle more complex startup profiles. However, this feature has not been tested.

2.3.1.2. Minimum Stable Generation

The *PROSYM* variable CapacityMin sets the generation level below which a station will not be dispatched (except after a startup when its ramp rate or run-up rate does not permit the station to reach that level immediately). This is typically set at a station's minimum stable generation. Stations often operate at poor efficiciencies at this level, but may operate at this level when their incremental costs at this level are higher than those of alternative suppliers of energy, and when high start costs and/or chronological constraints either prohibit a shut-down or would make the station unavailable when it would otherwise be needed in the future. In the Marginal Average Cost methodology supported by *MAC.pl*, the potential high cost of operating at this constrained level is excluded from the short-term average cost identification. *MAC.pl* compares each station's simulated generation level in each hour to its CapacityMin value, and excludes its average cost value from consideration as the marginal average cost.

In the general *MARKETSYM/PROSYM* environment, the CapacityMin variable can take a variety of data types, including in particular the Annual Pattern and Dated Constant types which include date specifications. *MAC.pl* does not at present recognize any dated data types; only a constant value may be used for this variable, at least in the

thermal.dat file. An example of an appropriate specification of CapacityMin in the *thermal.dat* file is:

```
Station.2
    ConvGroup       [V2]     1      5597
     …
   CapacityMax     130.4
    CapacityMin     55.0
```

If a dated value is required for a station, it can be specified in a separate file of *PROSYM* script.

2.3.2. *chp.csv*: schedule of excluded stations

Some generating stations are included in a dispatch for reason other than their ability to deliver economical electric energy. Examples of these are Combined Heat and Power or other cogeneration stations, for which the heat output contributes to the costs of operating the stations, or stations which meet non-energy grid requirements, such as requirements for voltage support, local spinning reserves, or other services. Stations which would produce an out-of-merit simulated average cost, and which often receive compensation in addition to the revenues from sales of energy, should be excluded from the set of stations that can set Marginal Average Cost. The *MAC.pl* script supports specification of such a set of stations in the *chp.csv* file, placed in the MAC run directory, as described in section 2.5.2 below.

MAC.pl reads a file which contains one row for each station which may be excluded from setting the Marginal Average Cost, followed by 12 comma-separated values, one for each month in the year, with a 1 for months in which the station is to be excluded from the MAC computation, and a 0 if it may be included. The file has no header row, and may be empty if no stations are to be excluded. The station names must be exactly the same as the station names in *thermal.dat*, including capitalization. An example file is:

```
CHP.Station.1,1,1,1,1,0,0,0,1,1,1,1,1,1
CHP.Station.2,1,1,1,1,1,1,1,1,1,1,1,1,1
Must.Run.Station.1,1,1,1,1,0,0,0,1,1,1,1,1,1
Must.Run.Station.2,1,1,1,1,1,1,1,1,1,1,1,1,1
```

2.4. Create the HourlyGenandCost.csv *file*

The first task in producing Marginal Average Cost values from a completed *MARKETSYM* run is to execute the *MAC_HourlyGenandCost* on the loaded output database, to produce the *HourlyGenandCost.csv* file that will be analyzed by the *MAC.pl* script. The preliminary step of using *Loader2K* to produce a *Reporter2K*-compatible database (the *Loaded Database*) is not covered in this documentation.

The creation of the *HourlyGenandCost.csv* file involves several steps:

1. Open *Reporter2K*;

2. Connect *Reporter2K* to the loaded database

3. Run the *MAC_HourlyGenandCost* Query as an Export to CSV to produce the *HourlyGenandCost.csv* file

2.4.1. *Open* Reporter2K

Figure 3 illustrates how to open *Reporter2K* from the Start Menu. *Reporter2K* is an Access database that is enhanced with VBA macros, which may be copied to other locations and opened by double-clicking; however its installed location in the *Program Files* directory is supported through the standard Windows Start Menu interface. Whichever copy of *Reporter2K* is used,; it must first have been modified with the *MAC_HourlyGenandCost* Query as described in section 2.2 above.

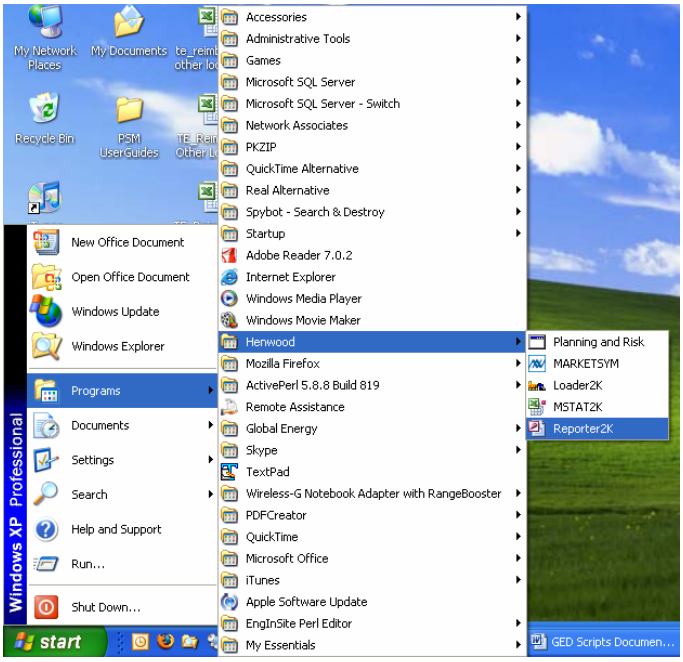

Figure 3. Launch *Reporter2K* from *Start* Menu

2.4.2. Connect to Loaded Database

When *Reporter2K* is opened, a top-level menu is offered to the user with seven buttons. Click the top button, “Select PROSYM Output Database”, and navigate to the location of the Loaded Database. Select the Loaded Database and click “Open”, as shown in Figure 4 below.

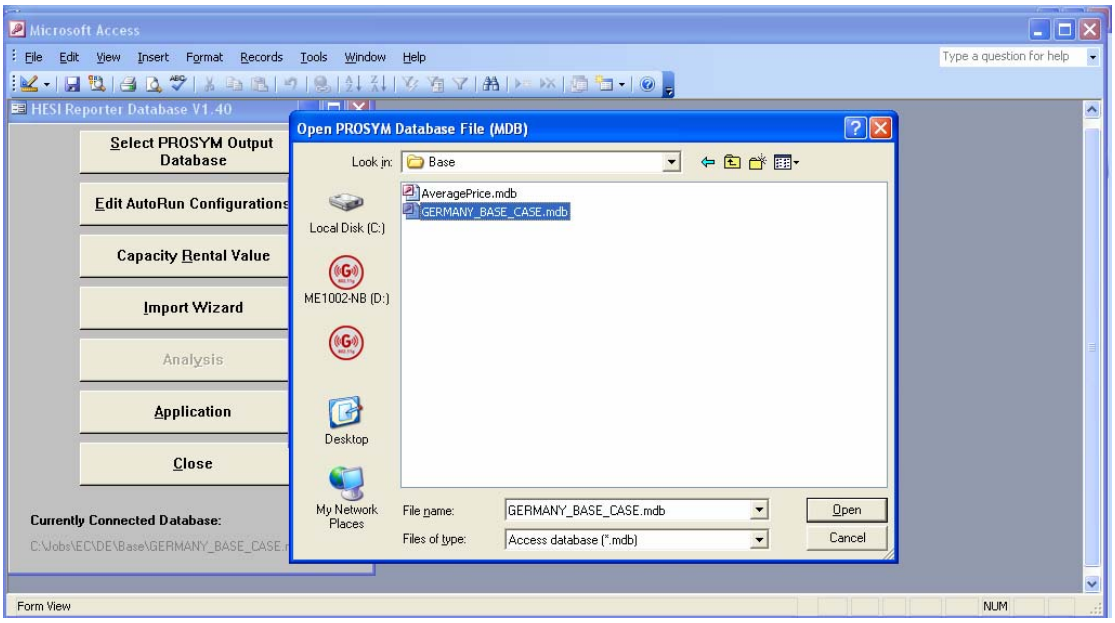

Figure 4. Connect to Loaded Database

2.4.3. Run the *MAC_HourlyGenandCost* Query

From the *Reporter* screen inside the *Access* interface opened by *Reporter2K,* find the *MAC_HourlyGenandCost* Query in the *Queries* section. Right-click on the Query and select *Export,* which will open up the *Export Query* window shown in Figure 5. Navigate to a sensible location (the default is the directory where the Loaded Database was found, which is by default the *MARKETSYM* run directory, which is as sensible a location as any), select *Save as type*: Text Files and name the file *HourlyGenandCost.csv,* and click the *Export* button. This is a fairly long step, producing a large file (hundreds of megabytes may be expected).

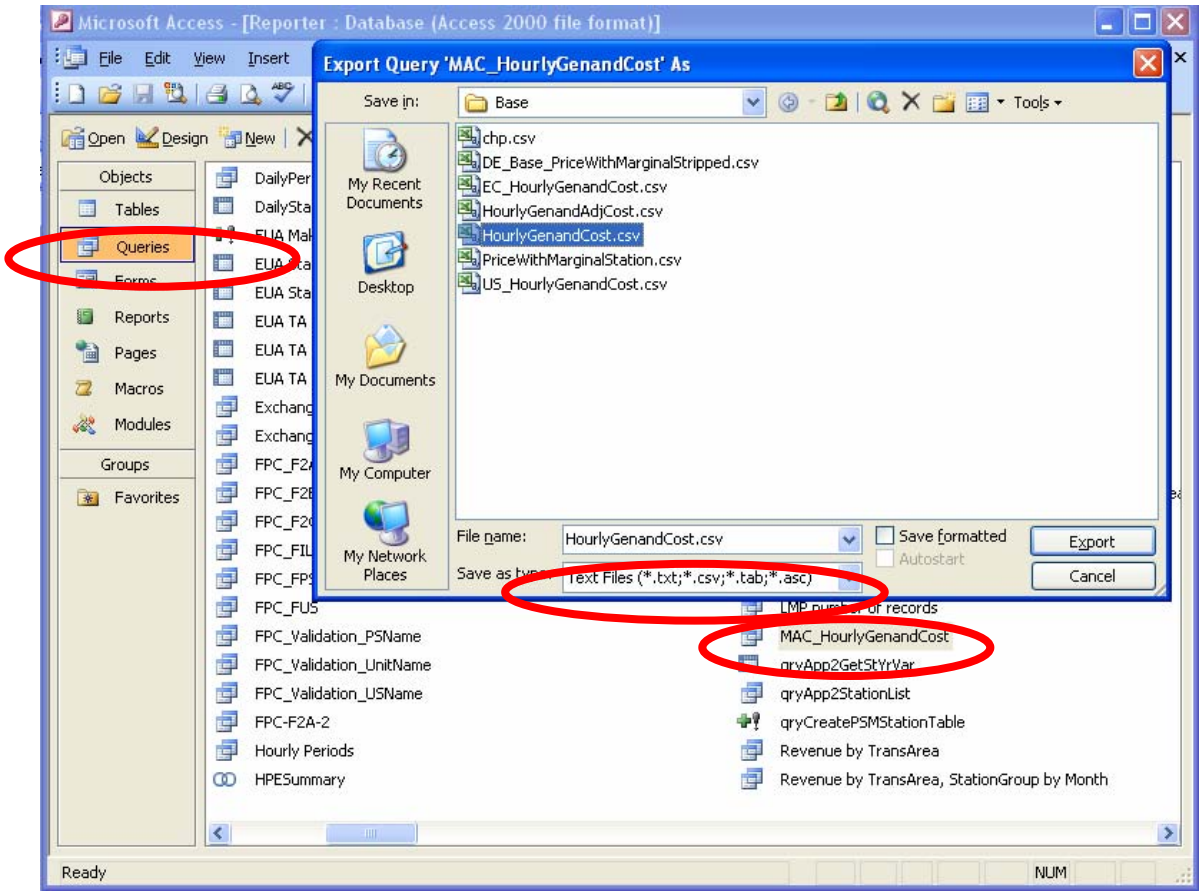

Figure 5. Export *MAC_HourlyGenandCost* Query as CSV

2.5. Running the MAC.pl script

The *MAC.pl* script processes the hourly output in the *HourlyGenandCost.csv* to produce an adjusted file which (1) removes start costs from the reported hourly generation cost and (2) reduces to zero the reported average costs for stations that are to be excluded from responsibility for Marginal Average Cost because of their inclusion (possibly month-specific) in *chp.csv* (see section 2.3.2) and/or because they are operating at or below Minimum Stable Generation (see section 2.3.1.2). Installation of the *Perl* interpreter is described in section 2.5.1. The files that must be found in the *MAC.pl* run directory are listed in section 2.5.2. The command line parameters for *MAC.pl*, and the precise steps to execute the script, are described in section. 2.5.3. Section 2.5.4 describes the default behaviour, while section 0 describes how to handle the complications arising on systems using non-standard (i.e., U.S.) date formats. Finally, section 2.5.6 contains recommendations for quick verification that the script is working properly.

2.5.1. The *Perl* Installation

MAC.pl is a Perl script, designed to be processed by Perl interpreter. Although the code is generic, it uses the file path specifications common to computers using Windows operating systems; it therefore requires the use of a Perl interpreter that will run under Windows. It has been tested on the Perl distribution published by ActiveState, which maintains a very high quality, well-documented Perl distribution, which the user may download at no charge from the ActiveState website, at http://www.activestate.com/store/activeperl/download. Registration is optional; after clicking the *Continue* button find the MSI package for the latest version (note that the script does very basic processing, unlikely to be version-dependent). Once the MSI is downloaded, install the software using the default configuration. Note that when the installation wizard completes, theWindows Start->Programs interface will contain links to Documentation, an OLE Browser , and the Perl Package Manager. The documentation is extensive; the other installed executables are not necessary for the use of *MAC.pl* (although the Perl Package Manager could be useful if *MAC.pl* were extended, for example to provide more sophisticated date processing). The installation wizard associates the *.pl* extension with the *perl.exe* executable, and includes the Perl executable's directory in the Windows search path, so that issuing name of a Perl script on the command line will pass the script, along with any command-line parameters, to the Perl interpreter.

Since the Perl installation wizard adds the directory containing the Perl interpreter in the Windows search path, it is convenient to place the *MAC.pl* script in this directory, which in a default ActiveState installation is *C:\Perl\bin*.

2.5.2. Required input files

MAC.pl requires the presence of three files in its run directory: a copy of a (possibly abbreviated), *thermal.dat* (see section 2.3.1), a copy (possibly empty) of *chp.csv* (see section 2.3.2), and the large HourlyGenandCost.csv file containing results from a *MARKETSYM* run (see section 2.4.3). These files might be in the run directory created in the *MARKETSYM* run, which would take advantage of the *MARKETSYM* facilities for run log management. However, they might also be placed in a results folder with descriptive path information; in the development of the system the latter strategy was followed, so that as an example the *MAC.pl* run directory for the Germany Base Case is *C:\Jobs\EC\DE\Base*, which contains only the required files.

2.5.3. Command-line parameters

The syntax to invoke *MAC.pl* is

CommandPrompt> <MACPath>MAC.pl RunDir <DateFormat>

where <MACPath> is the optional directory specification identifying the location of the *MAC.pl* script , RunDir is the mandatory specification of the directory where the required input files are located (and where the output file *HourlyGenandAdjCost.csv* will be created), and <DateFormat> is the optional specification of non-standard date formats for the input and/or output .csv files.

2.5.4. Default behaviour

The easiest setup and invocation involves placing the *MAC.pl* script in a directory in the Windows search path (such as *C:\Perl\bin*), opening up a DOS command window (Start->Programs->Accessories), navigating to the MAC.pl run directory (for example, cd \Jobs\EC\DE\Base), on a machine using the standard date format (DD/MM/YYYY). Then the script can be invoked simply as:

CommandPrompt>MAC.pl .

The period in the command line specifies that the script should operate in the current working directory. The absence of a date format indicates that the script will use as both input and output the standard date format DD/MM/YYYY.

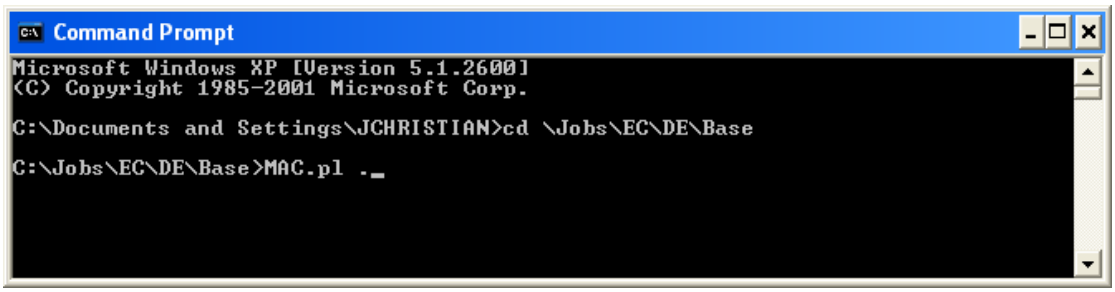

Figure 6. Recommended Default Command-Line Invocation of MAC.pl

2.5.5. Non-Standard Date Setups: Dealing with U.S. Date Configurations

A feature of the script is to handle a U.S. date format (MM/DD/YYYY) for the date fields in the input (*HourlyGenCost.csv*) and/or output (*HourlyGenAdjCost.csv*)files. The date formats supported are USUS (U.S. date formats in both files), USEC (U.S. dates in input file, standard in output), and ECUS (standard in input, U.S. in output).

2.5.6. Checking the output

Double-clicking the output file will load the first 65,536 rows into Excel. Check for correct date formats. Use the Auto-filter function to examine reasonable values in the *AvgGenGost EU/MWh* and*Reason* columns.

2.6. The MAC.mdb database (formerly AveragePrice.mdb)

The *MAC.mdb* database (renamed from the *AveragePrice.mdb* database distributed in early versions of the MAC system) is an *Access* database containing two queries that find hourly maximums from the *HourlyGenAdjCost* table. In the delivery copy of the database, this table is empty, so the database is quite small. It becomes large with the first use. The user should keep the empty version in a separate location, copying it

into run directories or other suitable location for use. It may be renamed at the user's discrection, and its location is not important.

The user imports the results of a *MAC.pl* run (as described in section 2.5), then can run either the *PriceWithMarginalStation* query, or the *MaxAverageCost* query. The *PriceWithMarginalStation* query in general returns more records than there are hours in the underlying run, because in some hours multiple stations may share the highest average cost in the dispatch. Analysis of the multiple stations in the high-cost position is possible from the *PriceWithMarginalStation* query, but is not supported by Global Energy; such analysis is up to the user. If a single record per hour is required, without information about the marginal station, then the user can open the *MaxAverageCost* query. Alternatively, the user can export the output of the *PriceWithMarginalStation* query, then use the *StripDup.pl* script on the resulting CSV file to eliminate multiple records for the same hour.

2.6.1. Importing *HourlyGenandAdjCost.csv*

The *MAC.pl* script produces a file in comma-separated-value (csv) format named *HourlyGenandAdjCost.csv*, containing a header row and a row of data for each station and each interval (usually but not necessarily an hour) in the study. The header row is:

Station,Date,HE,GenMW,Cost ThousandsEU,AvgGenCost EU/MWh,Reason

Station is the PROSYM name of each generating station in the run.

Date and HE are, respectively, the date and interval (usually but not necessarily the Hour Ending) specifications of the simulated interval.

GenMW is the simulated generation of the specified Station for the specified Date and HE.

Cost ThousandsEU is the total generation cost simulated in the hour, as reported by PROSYM, including fuel cost, start costs, and Variable O&M.

AvgGenCost EU/MWh is the average generation cost computed by the *MAC.pl* script. In the general case it is the total generation cost reported under Cost ThousandsEU, minus computed start costs, divided by simulated generation. It is zero if any of the following conditions holds: GenMW is zero, the station is in the excluded set for the simulation interval (from the *chp.csv* file; see section 2.3.2), and/or it is running at or below minimum stable generation. By setting AvgGenGost EU/MWh to zero in these cases, the station is prevented from setting the Marginal Average Cost

Reason explains the computation of AvgGenGost EU/MWh. The Reason string identifies the first case encountered that would cause a zero value; if there is a non-zero value the code will either be "*Running*" if there was no startup or, depending upon the start type, "*No-cost startup*," "*one-point startup: $StartCost*", "*hot start: $StartCost*", "*cold start: $StartCost*", or "*warm start cost TVC $GenTVC MW $GenMW StartCost $StartCost after $Off hours (between $LowerStartHours and $UpperStartHours)",* where *$StartCost* is the value of start costs used, and in the case

of the Reason message for warm starts the values of other variables are reported to allow validation of the computed value of *$StartCost*.

To import *HourlyGenandAdjCost.csv* into the *MAC.mdb* Access database, go to the*Tables* section of the Access interface, right-click, and select *Import…*, as shown in Figure 7below.

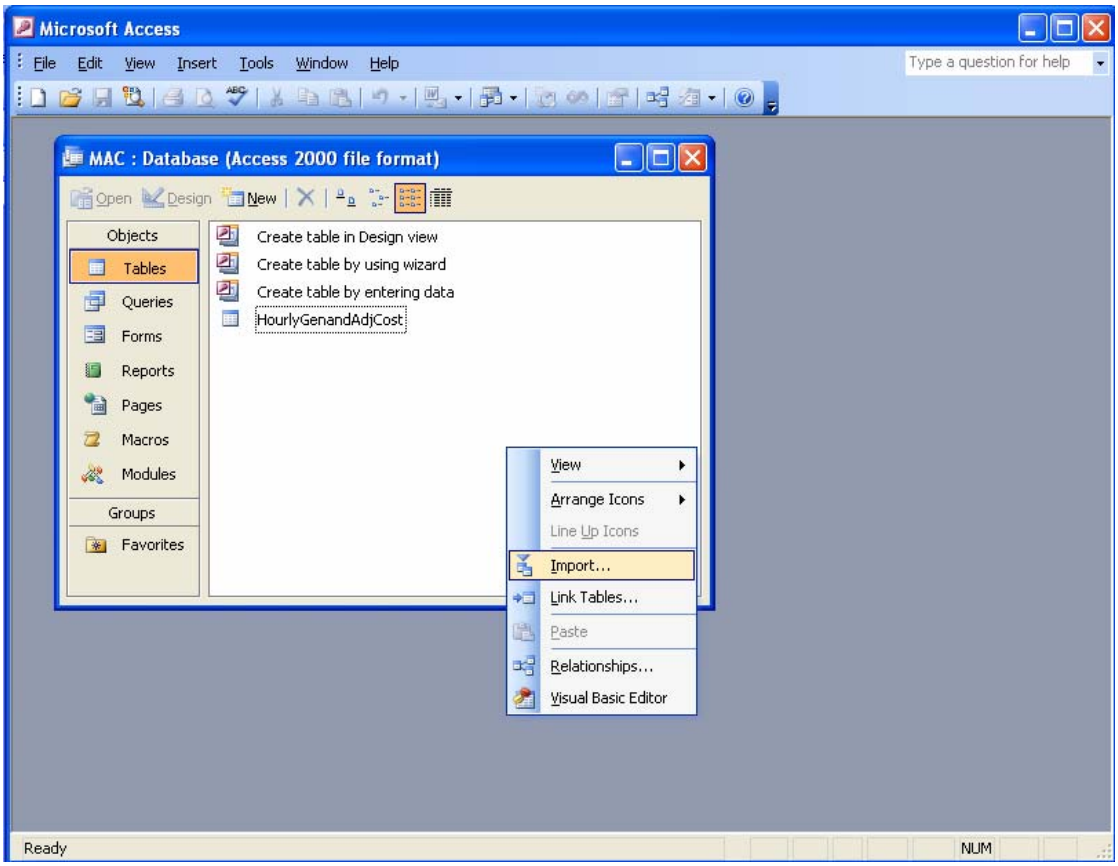

Figure 7. Starting Import Wizard to Import *HourlyGenandAdjCost.csv* into *MAC.mdb*

Next, select *Files of type:* Text Files (*.txt;*.csv;*.tab;*.asc) at the bottom of the *Import* file selection interface, then navigate to the location of the file, select it, and click the *Import* button. See Figure 8.

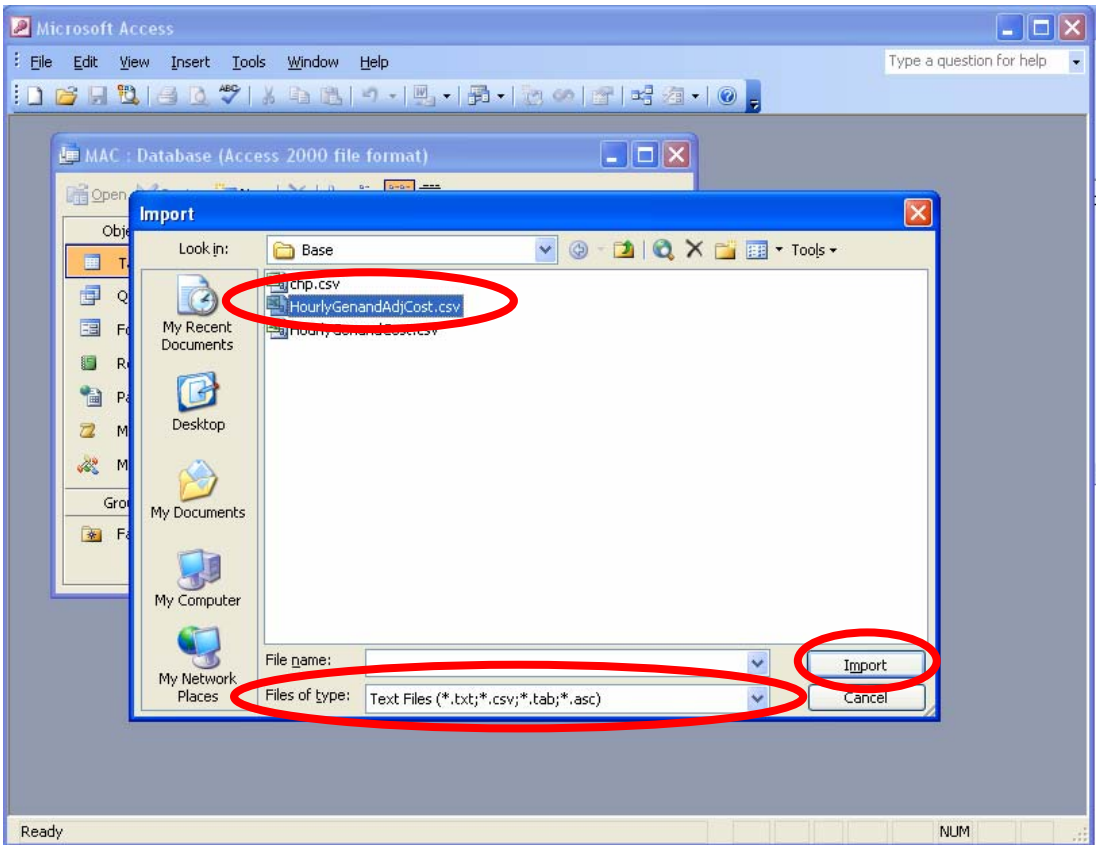

Figure 8. Select file for import

The wizard will then provide options for the import. The following must be set (see Figure 9 below):

- Set import to type *Delimited* using radio button on first screen
- Set delimiter to *Comma* using radio button on second screen
- Select *First Row Contains Field Names* by checking the box on the second screen

Then click the *Next>* button.

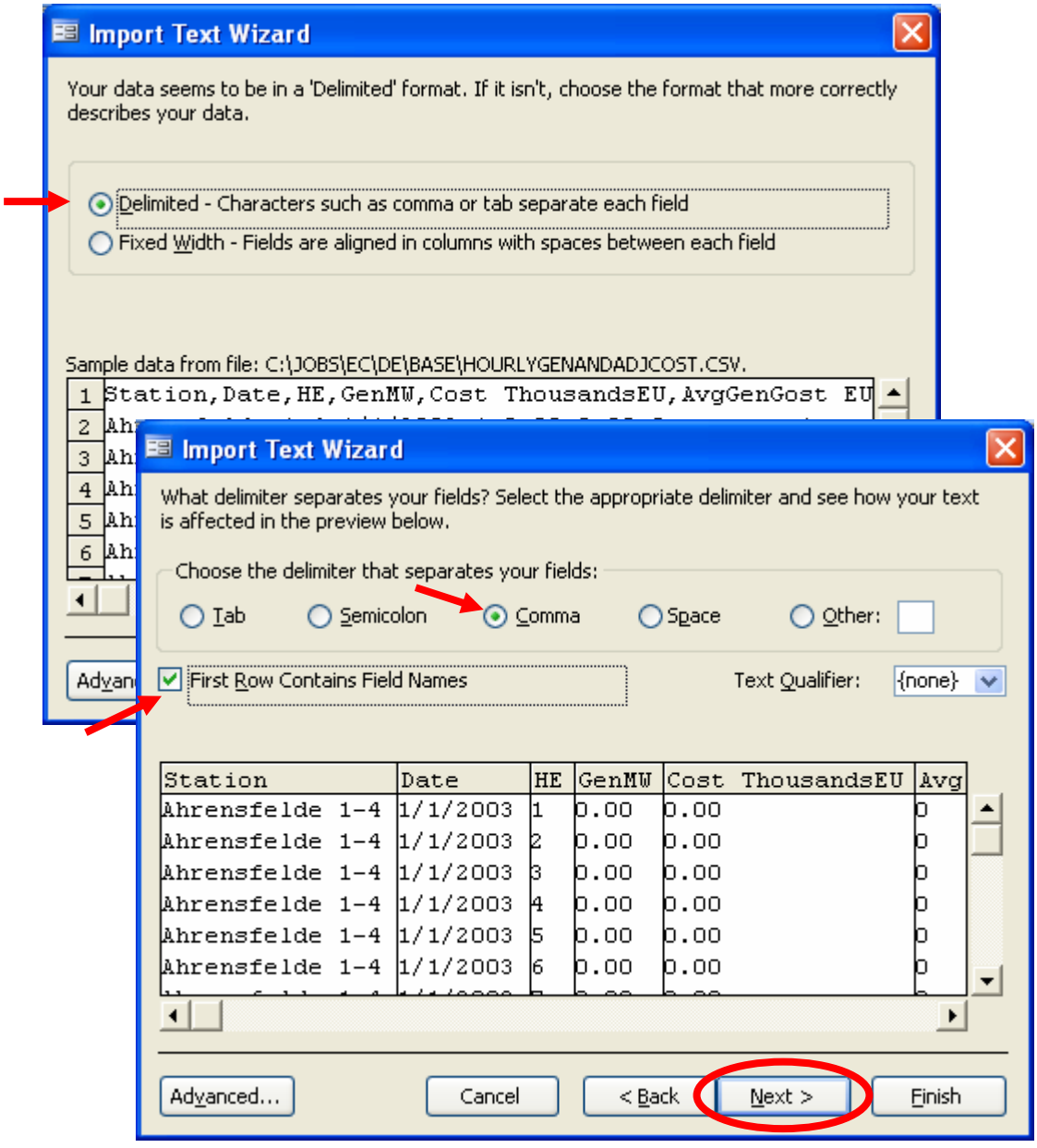

Figure 9. Configure CSV Import

2.6.1.1. Importing *HourlyGenandAdjCost.csv* as Existing Table (recommended)

The *Import Text Wizard* next offers to save the data in a new table or in an existing table. Either way can work, however it is simplest to use the existing table, as shown in Figure 10. This approach uses the data structures set up in the shell table provided in the blank *MAC.mdb* file, and ensures that the queries in the mdb will work. If the table is imported fresh, and if the first values in the file have zero in the AvgGenCost EU/MWh field, then the field will by default take integer values.

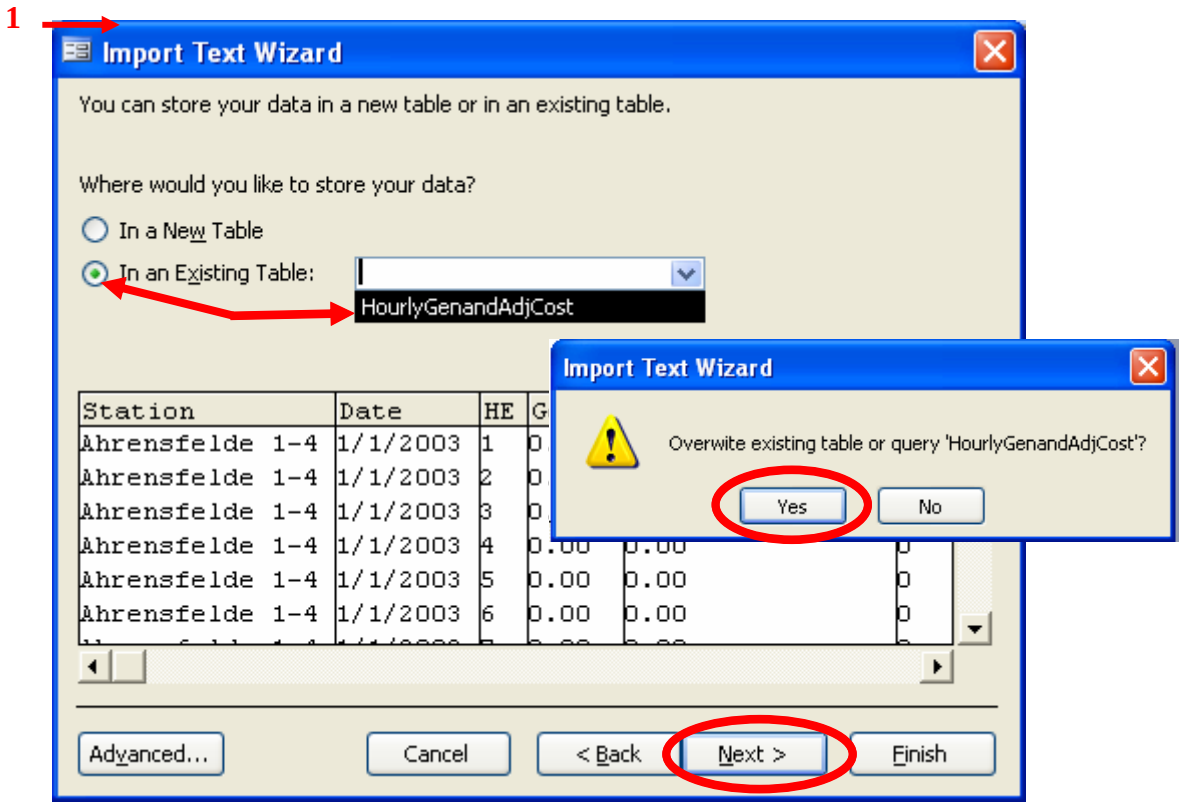

Figure 10. Import to Existing Table

2.6.1.2. Importing *HourlyGenandAdjCost.csv* as New Table

In the first uses of the *MAC.pl* system, users were importing the adjusted cost file as a *New* table, which is accomplished by selecting the radio button labelled "*In an New Table*" in the third screen of the Import Wizard (see **Error! Reference source not found.** above). If this option is selected, the Wizard will, by default, set data types on the new table based on the data in the first row of data. If the data in the Average Generation Cost field is "0", the Wizard will set the data type to *Long Integer,"* and truncate the decimal places for subsequent positive value. To correct this the user must override the default data type to specify type *Double.* From the second screen of the Wizard, click the *Advanced…* button to opent the Import Specification interface. Verify that the data type for the *AvgGenCost EU/MW* is set to *Double,* click the *OK* button to close the Import Specification, and click the *Finish* button to continue with the Import Wizard. See Figure 11 below.

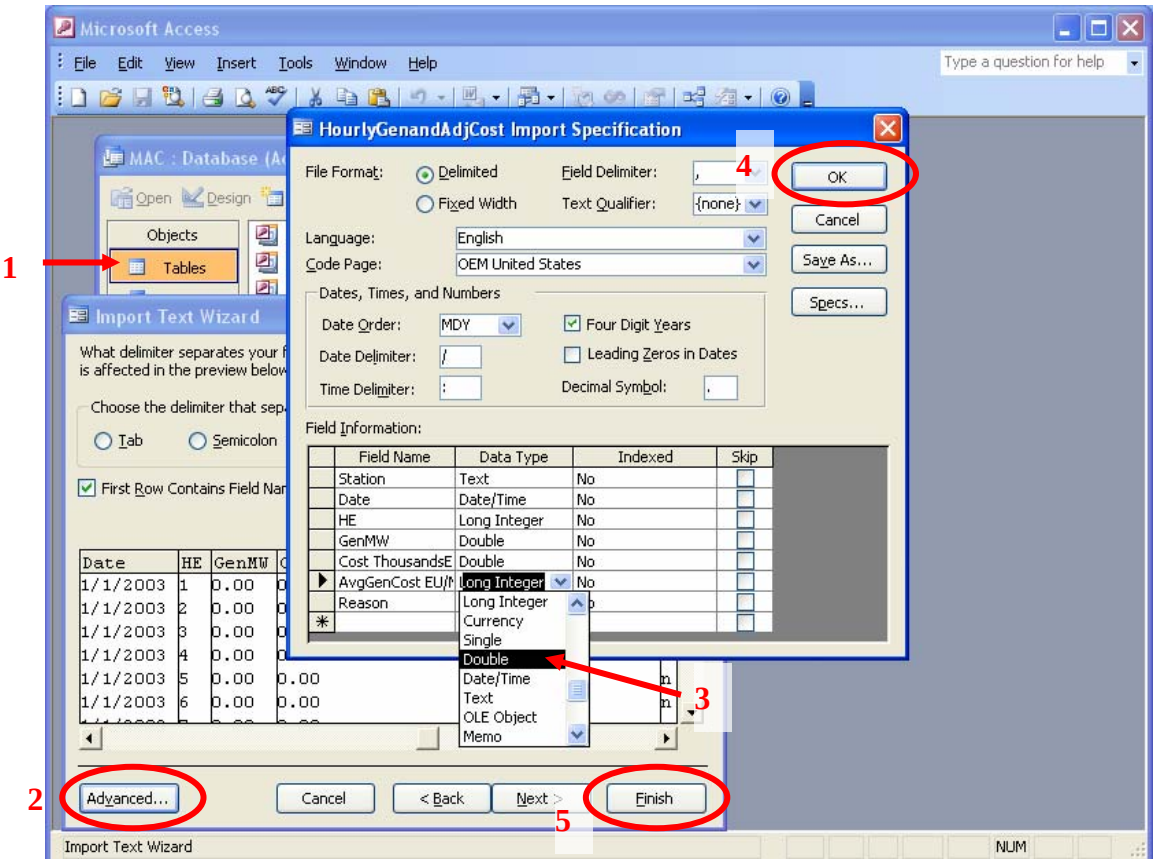

Figure 11. Check and Fix Data Type

By clicking the *Finish* button on the screen in Figure 11 above, the user instructs the Wizard to follow its default behaviour, which imports the file as a "new" table with a name for the table based on the name of the imported file; which will be

HourlyGenandAdjCost. This name is needed by the queries described in section 2.6.2. Since this name already exists, the Wizard will offer a confirmation dialogue to overwrite the existing table. Click *Yes*. See Figure 12 below.

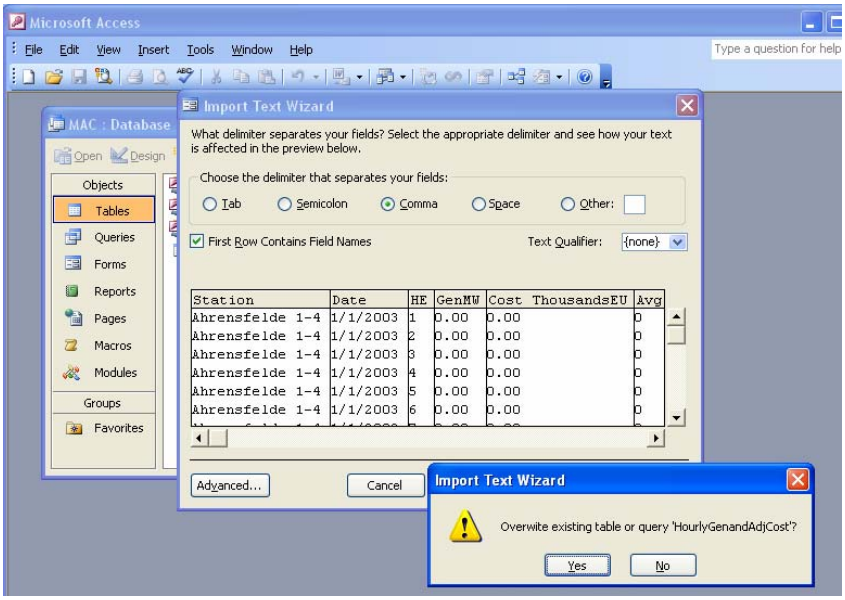

Figure 12. Confirm Overwrite of Existing Table

2.6.2. The Marginal Average Cost reports

The *MAC.mdb* Access database file contains two queries.

The *MaxAverageCost* query retrieves one record per hour (or other interval) and reports the highest adjusted average generation cost for the interval, but without any station-specific information.

The *PriceWithMarginalStation* query retrieves highest adjusted average generation cost in each hour (as computed by the *MAC.pl* script, see section 2.5), along with the station or stations with that cost and the Reason code (see section 2.6.1). If the station identification and/or Reason code are required this query should be run; if one record per hour is required it should be run as an export, with the results passed to the *StripDup.pl* script for further processing, as described in section 2.7 below.

2.6.2.1. Reporting Hourly Marginal Average Cost: the *MaxAverageCost* query

The *MaxAverageCost* query can be written in SQL as

```
SELECT HourlyGenandAdjCost.Date, HourlyGenandAdjCost.HE,
Max(HourlyGenandAdjCost.[AvgGenCost EU/MWh]) AS Price
FROM HourlyGenandAdjCost
GROUP BY HourlyGenandAdjCost.Date, HourlyGenandAdjCost.HE;
```

This is a simple query that identifies the maximum of the computed average costs as "price." This query may be run directly to produce a data grid which may be copied and pasted into other applications such as *Excel*, or it may be run as an export. In addition, the *PriceWithMarginalStation* query calls the *MaxAverageCost* query; if *MaxAverageCost* is not present the *PriceWithMarginalStation* query will fail.

2.6.2.2. Identifying Marginal Stations: the *PriceWithMarginalStation* query

The *PriceWithMarginalStation* query can be written in SQL as

```
SELECT HourlyGenandAdjCost.Station, HourlyGenandAdjCost.Date,
HourlyGenandAdjCost.HE, HourlyGenandAdjCost.[AvgGenCost EU/MWh] AS Price,
HourlyGenandAdjCost.Reason
FROM MaxAverageCost INNER JOIN HourlyGenandAdjCost ON (MaxAverageCost.Date =
HourlyGenandAdjCost.Date) AND (MaxAverageCost.HE = HourlyGenandAdjCost.HE)
WHERE (((HourlyGenandAdjCost.[AvgGenCost EU/MWh])=[MaxAverageCost]![Price]))
ORDER BY HourlyGenandAdjCost.Date, HourlyGenandAdjCost.HE;
```

The purpose of the *PriceWithMarginalStation* query is to retrieve information about the individual station or stations that sets the marginal average cost in each hour or other interval. It does this by selecting, in each hour, the stations whose average costs equals the marginal average cost that is found by the *MaxAverageCost* query. The query reports the station name, date, interval, Marginal Average Cost (labelled *Price*) and the *Reason* code.

If the user desires to postprocess these data so that there is only one record for each interval, while keeping some station information, then the query should be exported as an Excel workbook. Right-click on the query and select *Export…* to open the Access *Export Query* interface. Select Save as type: *Microsoft Excel 97-2003(*.xls)* and click the *Export* button. By default Access will create a file named *PriceWithMarginalStation.xls* in the working directory.

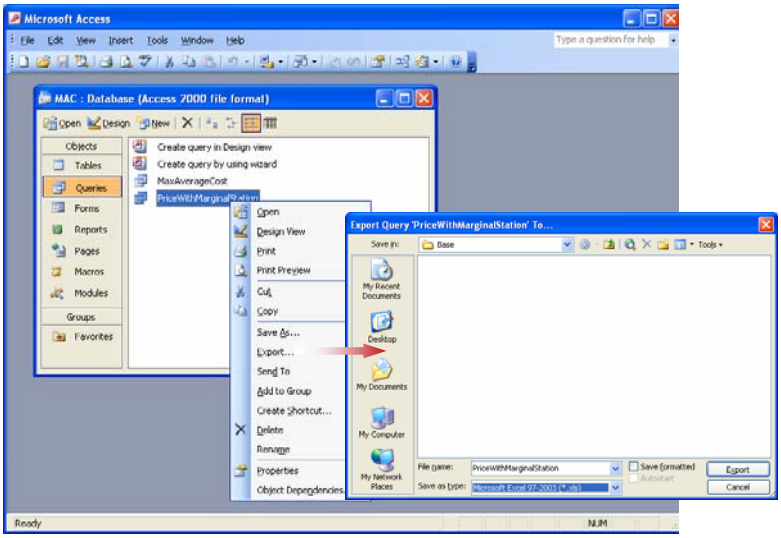

Figure 13. Export Query as Excel

2.7. Eliminating multiple hourly Marginal Stations: the StripDup.pl Script

The Perl script *StripDup.pl* reads a file of comma separated values and prints each line unless both the second and third fields are the same as in the previous line. The script file should be stored in a folder in the Windows search path . Since the Perl installation wizard adds the directory containing the Perl interpreter in the Windows search path, it is convenient to place the *StripDup.pl* script in this directory, which in a default ActiveState installation is *C:\Perl\bin*.

To produce a suitable CSV input file, the user can use Excel to open the *PriceWithMarginalStation.xls* file described in section 2.6.2.2, then use the Excel *File→Save as…*command. Save as type CSV. If the user accepts the defaults, the resulting file will be named *PriceWithMarginalStation.csv*. The same file could have been created directly by exporting the Access query as type CSV, however the Access Export to CSV is significantly slower than the Export to XLS.

Running *StripDup.pl* on *PriceWithMarginalStation.csv* produces a screen dump of the stripped records. To produce a new CSV file with one record per simulation interval, the script's output is directed to a new file. For example, the script could be directed to a file named *DE_Base_Stripped.csv* by issuing the command:

```
C:\Runs\DE\Base>StripDup.pl PriceWithMarginalStation.csv> DE_Base_Stripped.csv
```