Tutorial 5: Satellite Programs

Tutorial 5 Satellite programs	**Learning goals** 1. Using MS Excel as a satellite program. 2. Using the UNION# function. 3. Using GET$ and PUT$ to run a satellite program.	**Prior knowledge** 1. Interface 2. Tutorial 1: Getting Started 3. Tutorial 2: Objects and TeLiTabs 4. Tutorial 3: Interpolation and Integration 5. Tutorial 4: Solution Managementt	

1 Objective

In this tutorial, we will obtain the economical speed according to Townsin for each of our ships in the dataset. For this speed, both the resistance and power will be calculated in MS Excel and imported in Quaestor. Additionally, the Froude number of a ship will be calculated using a satellite program.

Start

For this tutorial, the knowledgebase from tutorial 4 is used. You can either use your own (verified) knowledgebase, or download it here:[Tutorial 5 Start]

Also an Excel sheet (Resistance prediction.xls) and an executable satellite program (froude_calc.exe) should be used. Use the hyperlinks to download the components and place them in your default Applic directory, usually [My Documents]\My knowledge\Applic\ (see also Directory structure).

2 Obtain a prediction of the economical speed, resistance and power

First of all, we'll clean up the knowledgebase.

blocked URL Delete all solutions (erase the workbase). Move all parameters and relations involving stability from the class *Top Goals* to *Stability*. Then create a new class: *Resistance*.

We'll export data from Quaestor to Excel, using the telitab *Stability_check#* as input. An inputscrip will define where to put which data.

blocked URL In the class *Resistance*, add the following relation:

Resistance# = EXCEL#(1, "NullString", "NullString", Stability_check# , TEXTITEM$(1), TEXTITEM$(2))

Do not yet save the relation, as we will enter some data in the dataslot.

See the EXCEL#() function for extensive information on this syntax. The arguments represent the following:

- *1* is the mode of the function, which assures the sheet will be visible
- The first *"NullString"* should be the excel sheet that is used. Because of the value NullString, the excel sheet should be embedded in the relation. Note that entering "NullString" results in the same as just entering ""
- The second *"NullString"* should be the sheetname under which the edited excelsheet should be saved. Because of the value NullString, our sheet will not be saved at all. Instead, data from the calculations in the sheet will be stored in a telitab (see below).
- *Stability_check#* is the telitab containing data for the input. Remember that our *stability_check#* telitab contains all variations of ships including GM and stability/comfort rating.
- *TEXTITEM$(1)* is the input script (in Telitab format) in which the positions in the spreadsheet are stated. Note that this syntax refers to the first text item (telitab) in the data slot of the expression.
- *TEXTITEM$(2)* is the output script (in Telitab format) in which the information is stated which values of the edited Excel sheet should be present in the telitab Resistance$.

Including the sheet in the frame will be done later, let's first define our input and output scripts. We will use method 2a from the examples of the EXCEL# function.

blocked URL In the dataslot of the relation, enter the following:

TEXTITEM1=
|7
"Rho" "Resis.Rho"
"Ship_no" "Resis.Ship(2)"
"Lpp" "Resis.Lpp(3)"
"B" "Resis.B(4)"
"T" "Resis.T(5)"
"Cb" "Resis.Cb(6)"
"DISP" "Resis.DISP(7)"|

This syntax results in the following. There are *6* parameters to be placed in the sheet. Parameter *Rho*, which is not case dependant, will be placed in the worksheet *Resis*, in the cell with **name** *Rho*. The parameter *Lpp* for example, is case dependant, the syntax above means:Which is the inputscript for our excelsheet.

1. Quaestor will go to the column number corresponding to the number behind the parameter name, in the case of Lpp this is the third column;
2. Quaestor will start placing values below the point where it has found a cell with the value (not the name) corresponding to the name provided in the input script in front of the column number, in this case "Lpp";
3. The same follows for the other multi-case parameters, which are placed in their corresponding columns below the cell with value "B", "T", etc.

After Quaestor has placed the values in the worksheet, other columns will calculate the economical speed according to Townsin, the predicted resistance at economical speed (a very rough indication) and the effective power (Rt * V_eco). We'll read these values and put them in *Resistance#* by means of the outputscript:

blocked URL In the dataslot of the relation, also enter:

TEXTITEM2=
|4
"Ship_no" "Resis.Ship(2)"
"V_eco" "Resis.V_eco kn(10)"
"Rt" "Resis.Rt(17)"
"PE" "Resis.PE(18)"|

Which uses the same syntax as the inputscript. For example, resistance *Rt* is column *17* in the sheet. All values below the cell with **value** *Rt* are stored in the telitab *Resistance#* as parameter *Rt*. Still, don´t save the relation yet.

We´ll use a new Attribute to direct the behaviour of Quaestor. The sheet should be included in the relation, but normally when a satellite is included Quaestor will immediately calculate the relation to check the satellite. We don´t want that this time, so the attribute @NOCALC is used.

blocked URL In the dataslot of the relation, also add the following attribute:

@NOCALC

See figure 1. Save the relation.

blocked URL

Figure 1 Adding the NOCALC attribute to the relation

Let's take a look at the excelsheet.

The worksheet where the calculations will take place is called *Resis*, as is referred to in the input and output script. The cell with the value for the density of the water is named *Rho*, because the input script for a single value uses the ***name*** of a cell. More names aren´t neccesary, the multiple case values are placed below the cells with ***values*** *Ship*, *Lpp*, *B* etc. in the columns defined in the scripts.

Now, for convenience we´ll include the excelsheet in the relation as binary. This is not necesarry, you could use an external file, as long as you stick to the directory structure of Quaestor. THis means that you should place the external files (.exe etc.) eihter in the Applic directory of the knowledge base or the Applic directory in the Kbs directory. So, we will include the binary:

blocked URL Right click the relation (not the parameter) for *Resistance#*, and select *Include binary in frame*. Browse for the excelsheet and double click it or select open. The sheet is now included in the relation.

Now you want to check the relation.

blocked URL Run a solution for *Resistance#* using the ship data. *Resistance#* is not present in Top Goals, in fact nothing is available in the Top Goals class, so nothing is shown. To use the data anyway, open the process manager, select Data and press the Select Data button. In case you use the classic buttons you'll see that the S button is changed into a U button (to unselect...). Now double click the parameter you want to make it a top goal, and finally run a solution using the play button.

During the calculation, Excel is started. As we used mode 1 for the EXCEL#() function, the sheet is shown (figure 2), and you are able to check the values Quaestor wrote to the sheet. Note that this dialogue is not shown if mode 0 is used.

blocked URL Click *OK* to continue with the solution

blocked URL

Figure 2 The sheet is shown during the solution

The solution, the telitab Resistance#, is shown in figure 3.

blocked URL

Figure 3 The telitab Resistance#, containing values read from the excelsheet

NOTE: The parameters V_eco, Rt and PE are not defined in the knowledgebase. Therefore, the Telitab (which is basically a text file) will contain these parameters, but they will not be available for further calculations or shown when you double click on the "Text/Telitab" next to Resistance# in the solution. If they should, the parameters have to be defined in the knowledgebase.

2 Combining telitabs using UNION#

We´d like to combine the results of the resistance prediction with the stability data we have gathered before, and present them together in one TeLiTab. Therefore, we´ll use the UNION# function.

blocked URL Add the following relation to your top goals:

Ship_calc# = UNION#(Stability_check#,"",Resistance#,"",1)

To avoid a possible manual input of *Resistance#*, make sure *Resistance#* is determined by *SYL: System/Function*. In the relation for *Stability_check#*, remove the *subgoals* attribute: the best ship evaluation does not include the resistance and speed.

Make sure you read the information about the UNION#() function. You may run a solution for *Ship_data#* together with the Data in the Dataset to see the results.

3 Use GET$ and PUT$ to run an executable satellite program

The economical speed of each ship V_eco was derived from the excelsheet above. Based on this speed and the ship's length the dimensionless froude number can be calculated. A simple executable satellite program is used to do this calculation, to show you how to use a simple PUT$() and GET$() action.

External programs able to use scripted input usually have their own syntax. Typically, you´d like to use some sort of template script, with certain values replaced by values calculated in Quaestor. This is possible using the TEMPLATE$() function. In this case, we are dealing with a very simple program, called *Froude_calc.exe*. This program needs the following kind of input file:

```
2
"V_knots" 16.00
"Lpp_m" 60.00
```

In which, obviously, 16.00 is the value of the ship speed in knots, and 60.00 is the length of the ship in meters.

Remark: this simple program works with TeLiTab input and output. Please note that this is not necesarry at all. You can use the TEMPLATE$() and PARSE#() function to create and read any ASCII in- and output (see also Use of external or satellite programs).

The following relation constructs this input file with the actual values of the speed and length of the vessel.

blocked URL Add the following relation as system in your class *Resistance*:

Froude_input$ = TEMPLATE$(TEXTITEM$(1),0,V_eco,Lpp)

The dataslot containing:

```
TEXTITEM1 =
|2
"V_knots" ~V_eco
"Lpp_m" ~Lpp
|
```

You might want to check out the possibilities of the TEMPLATE$() function. Furthermore, note that V_eco is added to the knowledge base. Give the dimension kts for knots, you will be informed that this is based on the SI units m/s.

The output of the Froude_calc program is the following:

```
2
"IER" 0
"Fn" 0.34
```

In which *IER* is an error check by the simple program:

ier = 0 No error
ier = 1 Error in the function FINDVAL of the simple program
ier = 2 File FROUDE.epi not found
ier = 3 Read errror on file FROUDE.epi
ier = 4 End of file reached on 'FROUDE.epi'

We´d like to put the Froude_input$ string in the program, and process the output of the program. This is realised by the following syntax.

blocked URL As it is TeLiTab output, we add the following relation to the class Resistance:

Fn# = GET$("Froude.EPO", "Froude_calc.exe", PUT$("Froude.EPI", Froude_input$))

This syntax places the content of the string parameter *Froude_input$* in the input file: *Froude.EPI*. **EPI** stands for: External Process Input. It runs the process: *Froude_calc.EXE* which generates the output file *Froude.EPO* (External Process Output) and gets the output back in a TeLiTab named *Fn#*. When it is not TeLiTab you receive as output, you can make TeLiTab by adding the PARSE#() function around the GET$() function to process the results. You might want to check out the GET$() and PUT$() functions.

As mentioned earlier, we don't have to include the program in the relation. An external program should be placed in the application directory of Quaestor (My documents\My Knowledge\Applic by default) or the specific Knowledgebase application directory (_KBname\Applic). Otherwise, Quaestor will ask for the program location.

To be able to select the Froude number as Top Goal for calculations, it has to be a parameter with a relation. The following relation retrieves the value for Fn from the TeLiTab *Fn#*.

blocked URL Add the following relation to the class Resistance:

Fn = SELECT(Fn#, 1, "Fn", 1)

Make sure that both Fn# and Fn are determined by *System/Equation*. Check out the description of the SELECT() function.

As an alternative you could also write:

Fn = Fn#.Fn

Please note that this is the general way to refer to data (Fn) from a TeLiTab (Fn#) and can be used for any TeLiTab content.

We´ll add the froude number to our total list of data in the next tutorial.

blocked URL For now, test the froude number functions by running a solution for Fn. Enter, for example, Lpp=100 and V_eco=15.

4 Check

You can verify your results by comparing it to [Tutorial 5 finish]

<< Back to tutorial 4 - Continue with tutorial 6 >>