Abstract

Energy is a precious resource, and yet it is often wasted due to processes using it not being optimal.

Process integration addresses this problem by trying to use potentially wasted resources, such as mass or energy, which would be sent outside the system. By using as an energy source internal elements of the process that otherwise would be wasted, we can reduce energy dependency on external sources, thus reducing energy costs.

The problem is that modifications to achieve this goal come at a price, and then it is hard to decide if it is worth reducing costs on external utilities at the expense of investing in new equipment for process integration.

The objective of this project is to develop a software tool capable of doing Pinch Analysis on an industrial process.

The tool is meant to assist the engineer, making easier to analyze a process involving hot and cold streams, displaying graph data to visualize possible energy recovery, helping to make decisions by estimating costs on utilities and on new equipment in order to find an optimal solution for process integration.

Furthermore, the tool is not simply programmed in classic functional way, but it is made using class based OOP (object oriented programming) and heavily documented so it can be easily modified and further upgraded to add more and more capabilities to it.
Master Thesis - Development of a software dedicated to pinch analysis

Daniel Guitart Pagola
Contents

Abstract 1

Preface 9

Introduction 11

1 The pinch analysis method 13
   1.1 Interest of the pinch analysis method .......................... 13
   1.2 Steps of the pinch analysis method .............................. 14
      1.2.1 Data extraction ........................................ 14
      1.2.2 Selection of the $\Delta T_{\text{min}}$ temperature ............. 15
      1.2.3 Search for the minimal energy requirements and construction
           of the composite curves .................................. 15
      1.2.4 Targeting the cost and area of the exchanger network .... 17

2 Example of the pinch analysis method 21
   2.1 Extracting the data from a real process ........................ 21
   2.2 The heat cascade, finding the pinch point ....................... 21
   2.3 The composite curves, finding the minimum energy requirements .. 22
   2.4 Targeting the area and the costs in the network ................ 22

3 Heat exchanger network design 27

4 Development of a software tool to help in pinch analysis 29
   4.1 Why choosing C++? ........................................... 29
   4.2 Why using object oriented programming? ........................ 30
5 Universal Modeling Language 31
  5.1 The Use Case diagram 31
  5.2 The Sequence diagrams 32
  5.3 The Logic diagram 32
  5.4 The documentation 32

6 The program: PinchDlg 35

7 Retaking the example using the program 37
  7.1 Previous set up 37
  7.2 Extracting the data from a real process 37
  7.3 Inputting the data in the software 38
  7.4 Calculation of MER and Pinch point 38
  7.5 Plotting and Visualization, a tool for decision making 38
  7.6 Choosing a model for costs 39
  7.7 Calculating target area and cost 40
  7.8 Saving the data for searching the optimal solution 40

8 Possible program upgrades 43

Conclusions 45

Acknowledgements and personal commentary 47

Bibliography 49

Appendix 49

A Use case diagrams 51

B Class diagram 53

C Sequence diagrams 55
D Source code

D.1 Headers
- Curve.h ........................................ 65
- Interval.h ....................................... 69
- IntervalH.h ..................................... 73
- PinchDlg.h .................................... 80
- PinchDlgDlg.h .................................. 81
- Point.h ......................................... 84
- Problem.h ....................................... 86
- resource.h ...................................... 102
- StdAfx.h ......................................... 103
- Stream.h ....................................... 104
- STREAM_TYPE.h ................................ 108
- Utility.h ....................................... 109

D.2 Sources
- Curve.cpp ....................................... 111
- Interval.cpp .................................... 114
- IntervalH.cpp .................................. 115
- PinchDlg.cpp ................................... 118
- PinchDlgDlg.cpp ................................ 120
- Point.cpp ....................................... 133
- Problem.cpp .................................... 134
- StdAfx.cpp ...................................... 158
- Stream.cpp ...................................... 159
- STREAM_TYPE.cpp ............................... 160
- Utility.cpp ...................................... 161
# List of Figures

1.1 Example process ........................................... 14  
1.2 Data extraction from the process .......................... 15  
1.3 $\Delta T_{\text{min}}$ based on the type of contact ........... 16  
1.4 $\Delta T_{\text{min}}$ based on the industrial sector ............ 16  
1.5 Costs associated with the HEN ............................. 17  
1.6 Example $\Delta T_{\text{min}}$ based on the type of stream and the film coefficient 19  
1.7 Coefficients for the cost formula, based on heat exchanger material 19  
1.8 Coefficients for the cost formula, based on pressure ratings ........ 20  
2.1 Data needed for the Pinch analysis ......................... 21  
2.2 Sorted temperatures in the exemple, values used to calculate ...... 22  
2.3 The heat cascade for the example ........................... 23  
2.4 Points of the composite curves calculated in the example .......... 23  
2.5 Hot and cold composite curves from the exemple drawn .......... 24  
2.6 Data needed with the utilities ............................... 24  
2.7 Enthalpy intervals with an example interval with the data displayed 25  
5.1 Sample of an Use Case diagram ............................. 31  
5.2 Example of a Sequence diagram ............................. 32  
5.3 Example of a Logic diagram or Class diagram ............... 33  
5.4 Example of the documentation inside the code ................ 33  
6.1 The program: PinchDlg ..................................... 36  
6.2 Monthly evolution of the program ........................... 36  
7.1 The streams and utilities input text files as they open after clicking the buttons ........................................... 38
7.2 The pinch temperature and the MER shown after the calculation . . 39
7.3 Text file output of the calculated GCC points . . . . . . . . . . . . . 39
7.4 GCC and CC as plotted by the program through GNUplot . . . . . 40
7.5 Menu for calculating the target cost . . . . . . . . . . . . . . . . . . 40
7.6 Results for the cost as shown in the program . . . . . . . . . . . . . 40
7.7 Add, clear and edit buttons and the text file with the results added . . 41
7.8 Different tries to find an optimum . . . . . . . . . . . . . . . . . . . . 41

C.1 CalculateEnthalpyIntervals() . . . . . . . . . . . . . . . . . . . . . . 56
C.2 CalculateMER() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
C.3 CalculateTargetArea() . . . . . . . . . . . . . . . . . . . . . . . . . . 58
C.4 OnButtonCostFileadd() . . . . . . . . . . . . . . . . . . . . . . . . . . 58
C.5 OnButtonCostFileClean() . . . . . . . . . . . . . . . . . . . . . . . . . 59
C.6 OnButtonCostFileOpen() . . . . . . . . . . . . . . . . . . . . . . . . . 59
C.7 OnButtonLoadData() . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
C.8 OnButtonPlot() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
C.9 OnButtonPlotbcc() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
C.10 OnButtonPlotcc() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
C.11 OnButtonPointsgraph() . . . . . . . . . . . . . . . . . . . . . . . . . 62
C.12 OnButtonStreams() . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
C.13 OnButtonUtilities() . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
C.14 OnButtonTcost() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
C.15 OnExit() . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
Preface

Project origins

This project is the result of my ERASMUS experience at the French city of Toulouse, after five years of engineering studies at UPC - ETSEIB in Barcelona, Spain. The project has been done at the LGC (Laboratoire de Génie Chimique) inside INP - ENSIACET University at Toulouse. The original project was offered to me by Raphaële Thery back on 2010 and the contents and the reach have been changed since then, but the essence remained the same. My tutors have been Prof. Raphaële Thery, on the process integration and programming field, and Prof. Gilles Hetreux, on the programming and optimization field.

Project motivation

I had great interest not only in trying to the Thesis abroad, with the complications that it brings, but to do it in a foreign language to get experience with it. The project was started in French but I was asked to switch to English as it is more international and thus more suitable for a thesis. I also wanted to do a project slightly outside my field of specialization to prove that I have the potential to learn about any subject fast, and then use the gathered knowledge in a practical problem. Furthermore, the this project is of great interest to me, as It implies learning programming and I always liked programming as a hobby, but never tried to go deeper and learn the real deal (Object oriented programming, modelling before programming, getting to know languages like C++).

Project requirements

The goal of the project is much better understood if the reader has notions of applied Thermodynamics, to understand matters like heat transfer, enthalpy of a stream, energy savings. Chemistry is not required, even if the examples on which the shown method is applied are chemical processes, this method is general and can be used in any field. Knowledge in C++ programming is essential to review the code, but the UML diagrams should be understandable to anyone that knows very basic programming.

Daniel Guitart Pagola
Introduction

The energetic problem and the need of process integration

A process could be defined as any operation that given some inputs produces some outputs. An industrial process is a process where raw materials are inputted and as a result a product is made. This requires energy to be done, as well as raw materials. The problem is that processes have an efficiency associated. This efficiency determines which amount of the materials and the energy are really used in the final product and which ones are released to the environment as wasted resources.

The problem is that in the actual industry in France, and in other countries, industrial processes are not as effective as they could and still have a lot margin for improving. In fact it is estimated that 11Mtep in final energy are used in utility treatment (production of steam, heating systems for streams) and with simple modifications and no need of new technology, 2Mtep could be easily saved.

Energy integration is then, needed for achieving this savings. But, what is energy integration? It could be defined as the series of improvements that can be done in a process so it can use more efficiently the energy, the raw materials or the water.

There are several examples of process integration, for instance:

- Recycling of materials and energy: to try to recover the wasted resources directly to energy, or to reuse wasted materials as raw materials.
- Heat recovery: to use the heat in streams that need to be cooled to heat streams that need to be heated.
- Co-generation: to use the wasted heat generated in a power plant (for example a turbine) to feed the needs of the process or to generate even more electricity.
- Reutilization of wasted heat: using the wasted heat from the process to feed the energy needs of machines that need mechanical or electrical energy by transforming it.

The focus of this project is the second one, heat recovery. The problem is that heat recovery analysis can be hard if the process isn’t viewed in energetic terms, which is often hard in complex processes.

Daniel Guitart Pagola
Project goals

The goal of this project is to create a software tool capable of helping the engineer to analyze the process to plan or improve process integration.

Project reach

This project only focuses in the heat recovery aspect of energy integration. It also only deals with the first phase of the energy integration process: Pinch analysis to find the optimum temperature so all the parameters are found in order to do the second phase, the design of the heat exchanger network.
Chapter 1

The pinch analysis method

1.1 Interest of the pinch analysis method

The problem of the study of heat recovery within a series of streams inside a process is a complex one. There is a huge amount of variables and targets to be achieved, such as temperatures on the streams, and technical constraints, but also restricted by investment in the heat exchangers, or the most complex, the design of the heat exchanger network (the placement of the heat exchangers in the process). There is a need then, for a simple analysis that can clearly see the energy needs and availability in any process no matter if simple or complex, existing or a newly designed one.

The pinch analysis was developed by Linnhoff (UMIST Manchester) at start of the eighties. It is a method that achieves this by taking into account the goal temperatures of the streams and assuming a $\Delta T_{\text{min}}$ which dictates the minimum temperature jump in any heat exchanger. This hypothesis in the temperature jump may lead to non optimal results, but multiple analyses can be done to find an optimal value to $\Delta T_{\text{min}}$. The analysis searches all the heat available in the hot streams and tries to give it to the cold streams, thus reducing to a minimum the external needs of heat and cold utilities, and finding the value known as MER (minimal energy requirements).

The pinch analysis is also capable of inter process study, because even if it focuses on a closed system, a required utility will be found for this system at a certain temperature, and this can be used to take advantage of the energy wasted in other system in order to feed the original one.

The main advantage thought, is that the analysis can study the whole process just knowing the temperatures and the flux of liquid circulating trough the streams, no matter what the process is or if there are exchangers actually in the process.
1.2 Steps of the pinch analysis method

1.2.1 Data extraction

It is necessary to retrieve the data in the real process figure 1.1 (or in the process that is being designed) to use it in the analysis.

The most important data is the temperatures $T$ on the streams. No matter what the streams do, all we have to know is the targeted temperatures before and after the heat exchange.

Next it is necessary to find the mass flux that goes through the stream, multiplied for the specific heat known of the liquid that fills the stream. This parameter is known as $mC_p$. This can be done if that value is known beforehand or if the actual heat exchange required from the utilities to achieve the temperature differences is known.

$$Q = m \cdot C_p \cdot \Delta T = m \cdot C_p \cdot (T_{out} - T_{in})$$  \hspace{1cm} (1.1)

With the formula 1.1 the exchanged heat $Q$ can be found and then it is possible to find the $m \cdot C_p$ term.

Once all this data is known it is possible to proceed to do a basic pinch analysis, but if finding the optimum $\Delta T_{min}$ for achieving the minimum cost is intended, it is necessary to provide the film heat transfer coefficients $h$ of the streams.

In the Figure 1.2 it is possible to see a summary of the data needed to proceed with the analysis.
1.2.2 **Selection of the $\Delta T_{\text{min}}$ temperature**

The $\Delta T_{\text{min}}$ temperature dictates the minimum jump of temperature that will be found in any counter-current heat exchanger inside the network.

It is a value that is extremely important, as it determines the results directly, yet is a human inputted value, based on experience.

This is a decision based in the experience of the engineer, the type of fluid in the process, the type of exchanger, and more importantly, previous results $\Delta T_{\text{min}}$.

1.2.3 **Search for the minimal energy requirements and construction of the composite curves**

First it is necessary to find the minimal energy requirements, as it is needed to calculate how much deviated is the cold from the hot composite curve.

For doing so, the Problem table method (Linhoff) will be used.

First it is necessary to define a hot and a cold stream: a hot stream is one whose initial temperature is hot and needs to be lowered by heat exchange. Similarly a cold stream is one whose initial temperature is cold and needs to be increased by heat exchange.

Then it is necessary to pick all the in and out temperatures from all streams and correct them to obtain the $T^*$ temperatures. To make this correction each temperature will be substracted $\Delta T_{\text{min}}/2$ if belongs to a hot stream or added this same amount if belongs to a cold stream.

After this, series of linear intervals have been defined, delimited by the $T^*$ temperatures. These are assigned a list of cold and hot streams that belong to the interval (by being inside the temperature range of the interval). The temperature difference for each interval is also calculated.

Next the heat exchange for each interval is calculated using formula 1.2

$$H = m \cdot C_p \cdot (T_{j+1} - T_j) \quad (1.2)$$

<table>
<thead>
<tr>
<th>Stream</th>
<th>Type</th>
<th>Supply Temperature $T_S^{(\circ C)}$</th>
<th>Target Temperature $T_T^{(\circ C)}$</th>
<th>$\Delta H$</th>
<th>Heat Capacity $CP$ (MW $\cdot$ K$^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hot</td>
<td>121</td>
<td>48</td>
<td>36.5</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>Hot</td>
<td>94</td>
<td>38</td>
<td>112</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Cold</td>
<td>32</td>
<td>66</td>
<td>-51</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>Cold</td>
<td>54</td>
<td>88</td>
<td>-102</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 1.2: Data extraction from the process
Now it is possible to calculate the sum of heat: If the required hot utility is assumed as 0, and the heat exchange for each interval, which was previously calculated, is added successively, then the transition between intervals that yields the lowest result is the minimum hot utility and the temperature associated is the pinch temperature. The last calculated value of the sum is the minimum cold utility. So with this algorithm it is possible to find one or multiple pinch points, and the associated minimal required utilities (or minimum energy requirements). It is important to mention that previously existing utilities are excluded from this calculus.

The next step is to construct the composite curves. Composite curves represent the energy in the process and the quality (temperature) of this energy. The hot curve represents the energy available in the streams of the process, also known as energy sources. The cold curve represents the energy required in the streams of the process, also known as energy sinks.

The hot curve is made getting the temperatures of all the hot streams and then creating temperature intervals, delimited by the temperatures. After the intervals are created it is checked which of the streams fit which interval, according to in and out temperatures. After the intervals have the data of which streams cross them, enthalpy increments are calculated from each interval using formula 1.2

Starting from an total enthalpy of 0, each point of the curve is calculated by picking the next temperature and adding the enthalpy calculated in the next interval, until all the intervals are used.
The cold curve is calculated in a similar way, but starting with a total enthalpy equivalent of the cold minimum energy requirement.

The grand composite curve (GCC), which will help the engineer to decide which utilities use to feed the process, can be constructed in two ways: Either by retrieving the data already calculated for the heat cascade, as essentially the points from the GCC are the same as those found while calculating the thermal cascade (each point in the cascade has a $T^*$ temperature and an added enthalpy) or it can be calculated by subtracting the cold composite curve extrapolated $+\Delta T_{min}/2$ to the hot composite curve extrapolated $-\Delta T_{min}/2$ (which also results in the GCC as in the pinch point this difference is 0).

### 1.2.4 Targeting the cost and area of the exchanger network

Now valuable data has been retrieved: An approximation on the required utilities is known. The problem lies in the cost of using these utilities.

There are two costs associated to the energy integration project:

- The cost of the utilities, that varies depending of the type of utility, its temperature, the pressure and other factors

- The investment cost of the new exchanger network that is needed in order to save energy so the process uses the minimum energy requirement as utilities

The first one is approximately linear and positive increasing. The second one is more complex, as the cost depends on the area of the heat exchanger network, which is still unknown and depends on $\Delta T_{min}$.

There is then, an optimal point where it is possible to have a cheap enough heat exchanger network (capital investment) and good savings in the utilities (costs of use).

Daniel Guitart Pagola
So the first goal is to find a good approximation of this area before even thinking the way to put the exchangers in the network in order to achieve the MER target.

To do so it is necessary to the engineer to decide which utility to use, with the help of the grand composite curve. It is necessary to select a utility that can cover the energy needs below and above the pinch point. In order for a utility to cover the needs it has to be above the curve if above the pinch point, or below the curve if below the pinch point. Also it is wise to choose the utilities well, as the cold utilities the colder they are more expensive. Similarly, the hot utilities the hotter they are more expensive.

The next step is to decide the number of exchangers. The best possible scenario number of exchangers can be calculated with the formula 1.3, which assumes that there will be no loops in the network and there is no independent problems. This is an approximation as it is not possible to know yet the distribution of the heat exchangers, yet it is often accurate.

\[
N_{ex} = (N_{streams,abovepinch} - 1) + (N_{streams,belowpinch} - 1)
\]  

After this it is necessary to construct the balanced composite curves, that are the composite curves but taking into account the utilities. They are constructed in an identical fashion to the composite curves, treating the utilities as additional streams.

Once the curves are calculated, it is possible to divide the balanced composite curves in several enthalpy (vertical) intervals. Each of these intervals has various values associated, which would belong to a heat exchanger transferring energy from the hot to the cold curve. These values are the in and out temperatures to the exchanger on the cold and hot side, the logarithmic mean of the temperature associated, calculated with the formula 1.4, the area of the exchanger, the film coefficient of each of the streams and the heat associated to them.

\[
\Delta T_{lm} = \frac{(T_{hot,in} - T_{cold,out}) - (T_{hot,out} - T_{cold,in})}{\ln(T_{hot,in} - T_{cold,out})}
\]  

All of these values are known, and the film coefficients can be estimated in several ways, such as considering the type of fluid inside the stream. There is no need of getting a precise value for the film coefficients, as the biggest part of the possible error induced depends on the relative order of magnitude of the film coefficients from the hot versus the cold streams. It is estimated that if the relative order of magnitude stays within a ten times range, the error induced is less than 10%. The approximate area of the exchanger can then be calculated using the formula 1.5

\[
A_{Total} = \sum \frac{1}{\Delta T_{lm}} \left[ \sum \frac{Q}{h} \right]
\]  

Doing this calculation for each enthalpy interval and adding the result, returns an estimation of the total area of the heat exchanger network.
<table>
<thead>
<tr>
<th>Type of stream</th>
<th>Heat transfer coefficient</th>
<th>$\Delta T_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas stream</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>Liquid stream</td>
<td>560</td>
<td>10</td>
</tr>
<tr>
<td>Condensing stream</td>
<td>1600</td>
<td>6</td>
</tr>
<tr>
<td>Vaporizing stream</td>
<td>3600</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 1.6: Example $\Delta T_{\text{min}}$ based on the type of stream and the film coefficient

<table>
<thead>
<tr>
<th>Materials</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbon Steel (CS)</td>
<td>30800</td>
<td>750</td>
<td>0.81</td>
</tr>
<tr>
<td>Stainless Steel (SS)</td>
<td>30800</td>
<td>1644</td>
<td>0.81</td>
</tr>
<tr>
<td>CS/SS or SS/CS</td>
<td>30800</td>
<td>1339</td>
<td>0.81</td>
</tr>
<tr>
<td>Titanium (Ti)</td>
<td>30800</td>
<td>4407</td>
<td>0.81</td>
</tr>
<tr>
<td>CS/Ti or Ti/CS</td>
<td>30800</td>
<td>3349</td>
<td>0.81</td>
</tr>
<tr>
<td>SS/Ti or Ti/SS</td>
<td>30800</td>
<td>3749</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Figure 1.7: Coefficients for the cost formula, based on heat exchanger material

The cost of investment of the heat exchanger network can be calculated assuming that the cost will increase following some exponent, and that an initial cost might be unavoidable as there are parts on the heat exchanger that still are needed even if the area of the exchanger is small. This can be summarized in the formula  1.6.

$$Cost = N_{\text{min}} \cdot \left[ a + b \cdot \frac{A}{N_{\text{min}}} \right]$$  \hspace{1cm} (1.6)

The coefficients for this formula can be found by approximating known prices of real exchangers in the market to the curve of the formula, and are different with the material of the heat exchanger, the pressure inside. As an added note, an approximation for spiral type exchangers is to multiply the 'b' coefficient by ten and to multiply the 'c' coefficient by 0.7 which approximately yields double the cost comparing to a normal exchanger for the most common network areas.

The cost found must be compared to the utilities, and to do so, it must be annualized, as the utilities are a yearly cost, and the heat exchanger network is an one time (investment) cost. To annualize the cost one can think of the investment as a loan, and consider yearly payments to pay in 'n' years with a rate of interest 'i', following the formula  1.7

$$\text{Annualized Cost} = \text{Cost} \cdot \text{Depreciation coef} = \text{Cost} \ast \frac{i \cdot (1 + i)^n}{(1 + i)^n - 1}$$  \hspace{1cm} (1.7)

The two results can then be summed to find the total cost. As the only global
<table>
<thead>
<tr>
<th>Pressure (bar)</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>shell/tube</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/10</td>
<td>30800</td>
<td>750</td>
<td>0.81</td>
</tr>
<tr>
<td>10/35</td>
<td>30800</td>
<td>890</td>
<td>0.81</td>
</tr>
<tr>
<td>35/35</td>
<td>30800</td>
<td>1089</td>
<td>0.81</td>
</tr>
<tr>
<td>10/60</td>
<td>30800</td>
<td>983</td>
<td>0.81</td>
</tr>
<tr>
<td>60/60</td>
<td>30800</td>
<td>1438</td>
<td>0.81</td>
</tr>
<tr>
<td>35/60</td>
<td>30800</td>
<td>1201</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Figure 1.8: Coefficients for the cost formula, based on pressure ratings

variable to this whole process is the $\Delta T_{\text{min}}$ (and the utilities), it is possible to plot this results to find the $\Delta T_{\text{min}}$ associated to the optimal cost.
Chapter 2

Example of the pinch analysis method

2.1 Extracting the data from a real process

To back up the procedure explained in the previous chapter, and to help understand how the pinch analysis works, it is interesting to see it used in an example. It is also interesting because the algorithms that work inside the program work almost following this procedure.

It is possible to ignore completely what the process does, it is just necessary to look for the in and out temperatures of the streams, classify the streams as hot or cold, and calculate the m*\(C_p\) (as only information about actual H is given). In this case, we already have the data summed in the table 2.1.

2.2 The heat cascade, finding the pinch point

Now it is necessary to find the different temperature intervals, either by putting the intervals for both types of stream side by side and creating the missing points manually (if the problem is simple), or by reducing the temperatures in both problems to

<table>
<thead>
<tr>
<th>Stream</th>
<th>Type</th>
<th>Supply Temperature (T_S(\degree C))</th>
<th>Target Temperature (T_T(\degree C))</th>
<th>(\Delta H)</th>
<th>Heat Capacity (CP) (MW (-K^{-1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactor 1 feed</td>
<td>Cold</td>
<td>20</td>
<td>180</td>
<td>32.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Reactor 1 product</td>
<td>Hot</td>
<td>250</td>
<td>40</td>
<td>-31.5</td>
<td>0.15</td>
</tr>
<tr>
<td>Reactor 2 feed</td>
<td>Cold</td>
<td>140</td>
<td>230</td>
<td>27.0</td>
<td>0.3</td>
</tr>
<tr>
<td>Reactor 2 product</td>
<td>Hot</td>
<td>200</td>
<td>80</td>
<td>-30.0</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Figure 2.1: Data needed for the Pinch analysis
Figure 2.2: Sorted temperatures in the exemple, values used to calculate $T^*$ and sorting the list. Then the flux associated to each interval will be calculated. This is shown at figure 2.2.

After this the Pinch temperature is found (150°C for the hot streams and 140°C for the cold streams) as are the minimum energy requirements (7.5kW for the hot utility and 10kW for the cold utility) using the heat cascade method, this can be seen in the figure 2.3.

2.3 The composite curves, finding the minimum energy requirements

The next step is to construct the composite curves as they are necessary to the target area calculations. To do so, each point temperature and enthalpy is found by using the formula in the previous chapter with the list of sorted hot and cold temperatures. This is shown in figures 2.4 and 2.5.

2.4 Targeting the area and the costs in the network

Now that the composite curves have been found, it is possible to include the utilities to find the balanced composite curves. It is also possible to do area calculations without the utilities, taking only into account the area in the composite curves outside the range of the utilities. This is useful for taking decisions before selecting the utilities, but cost calculations for an optimum can’t be done as they rely on picking utilities that fit the MER each time. Figure 2.6 shows the new data including utilities that cover exactly the MER. Figure 2.7 and equation 2.1 show the areas in the balanced curve and an example calculation of one of the areas.
Figure 2.3: The heat cascade for the example

<table>
<thead>
<tr>
<th>Enthalpy (kW)</th>
<th>Temperature (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>20.00</td>
</tr>
<tr>
<td>34.00</td>
<td>140.00</td>
</tr>
<tr>
<td>54.00</td>
<td>180.00</td>
</tr>
<tr>
<td>69.00</td>
<td>230.00</td>
</tr>
</tbody>
</table>

(a) Cold composite curve

<table>
<thead>
<tr>
<th>Enthalpy (kW)</th>
<th>Temperature (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>40.00</td>
</tr>
<tr>
<td>6.00</td>
<td>80.00</td>
</tr>
<tr>
<td>54.00</td>
<td>200.00</td>
</tr>
<tr>
<td>61.50</td>
<td>250.00</td>
</tr>
</tbody>
</table>

(b) Hot composite curve

Figure 2.4: Points of the composite curves calculated in the example
24 Master Thesis - Development of a software dedicated to pinch analysis

Figure 2.5: Hot and cold composite curves from the example drawn

<table>
<thead>
<tr>
<th>Stream</th>
<th>Type</th>
<th>Supply Temperature $T_s (^{\circ}C)$</th>
<th>Target Temperature $T_T (^{\circ}C)$</th>
<th>$\Delta H$</th>
<th>Heat Capacity Flowrate CP (MW $\cdot K^{-1}$)</th>
<th>Film heat transfer coef $h$, (MW $\cdot m^{-2} \cdot K^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reactor 1 feed</td>
<td>Cold</td>
<td>20</td>
<td>180</td>
<td>32.0</td>
<td>0.2</td>
<td>0.0006</td>
</tr>
<tr>
<td>Reactor 1 product</td>
<td>Hot</td>
<td>250</td>
<td>40</td>
<td>-31.5</td>
<td>0.15</td>
<td>0.0010</td>
</tr>
<tr>
<td>Reactor 2 feed</td>
<td>Cold</td>
<td>140</td>
<td>230</td>
<td>27.0</td>
<td>0.3</td>
<td>0.0008</td>
</tr>
<tr>
<td>Reactor 2 product</td>
<td>Hot</td>
<td>200</td>
<td>80</td>
<td>-30.0</td>
<td>0.25</td>
<td>0.0008</td>
</tr>
<tr>
<td>Steam</td>
<td>Hot</td>
<td>240</td>
<td>239</td>
<td>7.5MER</td>
<td>7.5</td>
<td>0.0030</td>
</tr>
<tr>
<td>Cooling water</td>
<td>Cold</td>
<td>20</td>
<td>30</td>
<td>10MER</td>
<td>1.0</td>
<td>0.0010</td>
</tr>
</tbody>
</table>

Figure 2.6: Data needed with the utilities
Figure 2.7: Enthalpy intervals with an example interval with the data displayed

\[ \text{Area} = \frac{1}{\Delta T_{\text{min}}} \left[ \sum \frac{Q_{\text{hot}}}{h} + \sum \frac{Q_{\text{cold}}}{h} \right] = \frac{1}{29.38} \left[ \frac{(150-95) \cdot 0.15}{0.001} + \frac{(150-95) \cdot 0.25}{0.0008} + \frac{(140-30) \cdot 0.2}{0.0006} \right] \]

The total resulting area is 7409.6 $m^2$. With all the areas being known, and with a number of exchangers of 7, using the formula of the previous chapter and if the exchanger is for example, made of carbon steel but with the custom coefficients $a=40000$ $b=500$ $c=1$, a total cost of 2.051M$ is obtained.

After some tries with different $\Delta T_{\text{min}}$ values it is possible to find an optimum near 10$^\circ$C value. How to reach this value will be explained more deeply at chapter 7.
Chapter 3

Heat exchanger network design

Creating the network is the next logical step to complete the analysis and get a useful and real result, but is not the focus of this project. Still, the basic notions of how this is done must be explained.

An important thing to notice is that this problem is combinational with a huge number of possible results, as there are many ways to interconnect the network. There are usually two approaches to solve this:

- The pinch design method, which uses heuristic and feasibility rules and checks sequentially if it is possible to put the exchangers in the network according to this rules.
- Optimization method, using mixed integer nonlinear programming, method that uses classic algorithms in optimization meant to solve combinational problems (such as branch and bound algorithm)

Continuing the program to solve this step would require skills in programming and optimization, but is clearly feasible. For more information check the additional bibliography section [1]
Chapter 4

Development of a software tool to help in pinch analysis

4.1 Why choosing C++?

The language chosen to develop the application is C++. There are other languages which were a possibility, Java being the main one. But C++ was the final decision. It’s better to start talking about the disadvantages of using C++:

- It is a strict and complex language, so it is harder and slower to code with it. For instance it requires a careful memory management.
- It does not have default GUI libraries, so using Visual C++ is required
- Programs made with Visual C++, require a visual C++ redistribution package to work properly in another computer, and can only work on windows platforms

But it has important advantages:

- Programs made with C++ usually have better performance than programs made with other languages
- While it has not as many free libraries as Java, most professional applications have their libraries developed in C++
- Great interoperability with other programs due to the amount of APIs existing for C++, as interoperability with professional hardware. For instance, compatibility with math processing programs like MATLAB or with optimization programs.

So basically it was a language that has the better compatibility among the applications which are intended to be connected to the program in future updates, and still it is a powerful and fast language.
4.2 Why using object oriented programming?

There are two approaches to programming. Functional programming, whose base is to break the program into smaller subprograms called functions. And the other approach is object oriented programming, where a program is composed of classes, who have variables and methods, which are used to interact with other classes or to do operations.

Object oriented programming (OOP) is the standard nowadays; the reason is that at the end, it is more logical to think of a problem first as a bunch of objects and then see the possible ways of interaction, than to start thinking of the solution beforehand. As an example, in the problem of this project, it is easier to think of the problem as a bunch of streams and utilities, which need to be analyzed, than to start applying an algorithm right away.
Chapter 5

Universal Modeling Language

Known as UML, this language can be considered as a prerequisite for good OOP. Its strongest point is the ability to take an arbitrary problem and analyze it part by part, so the problem can be represented using classes and methods. Then these methods are thought in a generic way, without using any specific programming language using diagrams.

5.1 The Use Case diagram

This diagram shows the interaction between the user, the program (the methods of the program), which in this case are the possible clickable buttons on the interface, and the interaction with the files that the program uses. Figure 5.1 shows an example Use Case diagram, the diagram for the program, as other diagrams, can be found in the annex. In the example, the user can use three buttons, but the third only if it has used the second first. The second button will interact with a file.

![Sample of an Use Case diagram](image)

Figure 5.1: Sample of an Use Case diagram
5.2 The Sequence diagrams

These diagrams show basically how each method in the program works, describing operations, interactions with other methods, the classes that are used by the method, loops done, etc. An example diagram is shown in figure C.13 but the diagrams of all clickable buttons in the program are in the Annex. In the example, a function calls methods from another classes till the objective is reached and a message (information) is retrieved.

5.3 The Logic diagram

This diagram (Figure 5.3) shows every class which the program is made of. It also shows the methods which belong to the classes, the variables of each class, but most importantly, it shows the relation between classes: which classes are connected and with which rules. In the example a university is an aggregation of people, and there are two specializations of the class people, teachers and professors, who inherit all the characteristics of the class people.

5.4 The documentation

It is extremely important to heavily document each piece of work inside the program, so it can be understood to be easily modifiable, upgradeable by oneself or by other programmer. For this reason every class has a detailed description, as does every function, method, variable in the program. This documentation is easily accessible if the project of the program is open using Rational Rose, one of the tools used to create the program, but a copy of the documentation is copied into the code before each function to make it available to everyone.
### Other Operations (specified)

```cpp
/// Operation: readData%4D907A410344
// Method where the data is read from the streams.txt and
// utilities.txt file and stored in the Stream class
void readData();
```

```cpp
/// Operation: calculateProblemArray%4D907ABC0002
// Method where the data is sorted by temperature to
// create
// the intervals for the calculation of the MER (and the
// GCC). The data is stored in the Interval Class.
void calculateProblemArray();
```

Figure 5.3: Example of a Logic diagram or Class diagram

Figure 5.4: Example of the documentation inside the code
Chapter 6

The program: PinchDlg

The program itself is the purpose of the Thesis. It is a tool to help engineers to perform analysis on the design phase of processes that involve heat exchange, or to refine existing processes performing pinch analysis to help to understand better the process energy needs.

It also provides graphical data as an output, but for doing so it requires the installation of GNUplot, a free graphing utility tool under the GNU license. GNUplot is included inside the distribution of Octave, a command line based mathematical software, so it is easy to make an installation of GNUplot on installing Octave.

The program, which is called PinchDlg, uses MFC libraries so it is based on Microsoft Windows and only runs on Windows OS, but the build is static so the necessary libraries are provided inside the executable. Being static means that the program should run in any Windows machine without problems, still in case that the program does not work well, Microsoft Visual C++ 6 Redistributable is included also, to provide the user an installer to get the possible missing libraries. Figure 6.1 shows the program and figure 6.2 shows the monthly evolution of the program.

The program works at a fixed resolution, but the windows size is small enough to fit most screens. In case that one user has a resolution problem, a different resolution build can be made swiftly on demand.
Figure 6.1: The program: PinchDlg

(a) April  
(b) May  
(c) June

Figure 6.2: Monthly evolution of the program
Chapter 7

Retaking the example using the program

This chapter is dedicated to explain how to use the program to solve a problem. The problem used will be the previously shown example.

7.1 Previous set up

If the user wants to use the graphical tools provided, it is necessary to previously install Octave to install GNUPlot, as the program uses this program as the plotting engine. To install Octave just click on the executable provided and follow the steps. It is recommended to install Octave in the default directory.

Now move the program folder to your directory of choice and open the “pathsettings.txt” file with a text editor and follow the instructions to set the program path so it finds octave and it finds the working directory.

The next step is to also add the directory path to the plot macro files “gnuplotSettingsCC.txt”, “gnuplotSettingsBCC.txt” and “gnuplotSettingsGCC.txt”.

Now the graphical tool is ready to work and all the functionalities in the program should work fine.

7.2 Extracting the data from a real process

This step is identical to the one shown in chapter 2, but as we are performing a Target cost analysis also, it will be necessary to add a column with the film coefficient values. So the data used will be the one summarised in figure 2.6 in chapter 2.4.
7.3 Inputting the data in the software

Transferring this data to the program is simple: just click the streams button and the text file that must be edited will be opened with the default text editor. Input the data as shown in the example. If the process has defined utilities, input them the same way clicking in the utility button. If utilities are unknown or yet to be defined it is possible to uncheck the “include utilities” box to perform an analysis without them. Results made with this kind of analysis are all valid but consider that only the internal transfers in the process are taken into account. This is seen in figure 7.1.

When all the data is inputted, click the “Load Data” button. It is important not to forget this step because data won’t be read by the program unless the “Load Data” button has been clicked. Initially all the calculation buttons are disabled until the user loads the data to prevent this error, but not for a second calculation.

7.4 Calculation of MER and Pinch point

To find the pinch temperature and the Minimal Energy Requirements is as easy as click the “Calculate T Pinch and MER” button. $T^*$ Pinch will be displayed (to find $T^*$ Pinch hot, add $\Delta T_{\text{min}}/2$ to this temperature, and to find $T^*$ Pinch cold, subtract $\Delta T_{\text{min}}/2$ to this temperature) and the minimum requirements in hot and cold utility will be also displayed (Figure 7.2).

7.5 Plotting and Visualization, a tool for decision making

Now it is possible to view the process information in a graphical way by clicking on the Plot buttons. This way the user can check the overall process and the utility requirements at the GCC curve or the cold and hot process in the CC or the BCC (which includes the utilities). By reviewing the curves the engineer can decide which utilities to use in the process, include them by clicking the “utilities” button, and reload the process clicking “Load Data” and the “Calculate” buttons (Figure 7.4).
7.6 Choosing a model for costs

The next step is to choose a model for the heat exchangers. The user can select between preexisting models by clicking on a list, and all the coefficients will be filled automatically. This list contains heat exchangers sorted by material or by pressure.

The exchangers must be paid annually plus some interest for the loan required, this extra money will be taken into account automatically by inputting the interest and the number of years (Figure 7.5).
7.7 Calculating target area and cost

To find the target area just click the button “Calculate target area”. The results will be shown in the boxes right to the button. The user must input the estimated cost for the cold and hot utility in the boxes before clicking the button, as having chosen the coefficients for the heat exchangers and the loan data (Figure 7.6).

7.8 Saving the data for searching the optimal solution

Now it is possible to save the results by using the last set of buttons “add” “edit” and “clear”. The add button adds the last calculated results to a text file. The clear button clears all the results from the file but conserves the headers. The edit button opens the file in a text editor so the user can manually edit the file or use the results with another program (Figure 7.7).

By calculating for different $\Delta T_{\text{min}}$ values and adding them to a file it is possible to estimate an optimum value for the $\Delta T_{\text{min}}$ value. This can’t be done automatically.
Figure 7.7: Add, clear and edit buttons and the text file with the results added

because the user must choose a different set of utilities each time for the results to make sense (and the balance curves to be balanced, which means covering the exact energy required) (Figure 7.8).
Chapter 8

Possible program upgrades

The program is meant to be upgraded with new functionalities in the future. The most important ones would be:

- The addition of boxes for the user to input different types of hot utility, as the use of different pressure utilities is a standard, and can lead to clear savings. Moreover the amount of each kind of utility can be inferred from the GCC Plot
- The addition of an automatic solution for choosing the utility type, based on the data of the GCC
- The possibility of using different material types for each heat exchanger.
- The creation of a graphical motor so the program doesn’t rely on external engines.
- The addition of connectivity with MATLAB like programs, using the provided APIs
- Finally the most ambitious one would be for the program to solve the next step and find the distribution of the heat exchanger network, but that could be a whole new program by itself.
Conclusions

On the technical side, it is possible to save energy in a process by studying well the points which are a source of wasted energy and the points that need energy. Process integration with the adequate tools of analysis can lead to saves in resources and money.

On the objectives side, the goals have been achieved: The program was done, and it functions as intended. The program has been tested and the calculations it makes give correct results, based on previous handmade calculations and existing data. I also learned a lot in the field of programming and process integration.
Acknowledgements and personal commentary

First, about the project, I feel I have learned a lot from doing this project, as I always enjoy trying subjects outside my specialization. I also love finally having gone deeper into programming, as I never dared to.

Thanks to my parents and my family, who have supported me, not just now when living abroad but every single day of my life.

Thanks to all my friends, both those old from my country, who have wished me luck every day, but also the new ones in France, who I will never forget.

Thanks to my tutor here, Raphaëlle and also to Gilles, who have helped me since I put a foot in A7. And also thanks to all the colleagues in the office.

This project closes an episode of my life, so it was really important to me to write these words. Thanks to everyone.

Daniel Guitart Pagola
Bibliography


Appendix A

Use case diagrams

This chapter contains the use case diagram that shows all the actions that the user can make when using the program and the functions they trigger.
Appendix B

Class diagram

This chapter contains the class diagram of the program. In the diagram it is possible to see all the objects that compose the program and the relations between them.
Appendix C

Sequence diagrams

This chapter contains the sequence diagrams for the program functions. These are normally associated with the buttons inside the program dialog box.
Figure C.1: CalculateEnthalpyIntervals()
Figure C.2: CalculateMER()
Figure C.3: CalculateTargetArea()

Figure C.4: OnButtonCostFileadd()
Figure C.5: OnButtonCostFileClean()

Figure C.6: OnButtonCostFileOpen()
Figure C.7: OnButtonLoadData()

Figure C.8: OnButtonPlot()
Master Thesis - Development of a software dedicated to pinch analysis

Figure C.9: OnButtonPlotbcc()

Figure C.10: OnButtonPlotcc()
Figure C.11: OnButtonPointsGraph()

Figure C.12: OnButtonStreams()
Figure C.13: OnButtonUtilities()

Figure C.14: OnButtonTcost()
Figure C.15: OnExit()
Appendix D

Source code

This chapter contains all the source code written for the program, which is where the biggest part of the work time was invested. The code itself contains descriptions that explain what the functions do at the beginning of each function.

D.1 Headers

D.1.1 Curve.h

```cpp
/// # begin module%1.4%.codegen_version preserve=yes
/// Read the documentation to learn more about C++ code
generator
/// # end module%1.4%.codegen_version

/// # begin module%4D11CC70100.cm preserve=no
/// %X% %Q% %Z% %W%
/// # end module%4D11CC70100.cm

/// # begin module%4D11CC70100.cp preserve=no
/// # end module%4D11CC70100.cp

/// Module: Curve%4D11CC70100; Pseudo Package specification
/// Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\Curve.h

#ifndef Curve_h
#define Curve_h 1

/// # begin module%4D11CC70100.additionalIncludes preserve=no
/// # end module%4D11CC70100.additionalIncludes

/// # begin module%4D11CC70100.includes preserve=yes
#include <vector>

Daniel Guitart Pagola
```
// ## end module %4DD11CC70100 . includes

// Point
#include "Point.h"
// ## begin module %4DD11CC70100 . additionalDeclarations preserve = yes
// ## end module %4DD11CC70100 . additionalDeclarations

// ## begin Curve %4DD11CC70100 . preface preserve = yes
// ## end Curve %4DD11CC70100 . preface

/// Class: Curve %4DD11CC70100
/// Category: <Top Level>
/// Persistence: Transient
/// Cardinality/Multiplicity: n

class Curve
{
    // ## begin Curve %4DD11CC70100 . initialDeclarations preserve = yes
    // ## end Curve %4DD11CC70100 . initialDeclarations

    public:
        // ## Constructors (specified)
        // ## Operation: Curve %4DD2432400CD
        Curve ( std::string name );

        // ## Destructor (generated)
        ~Curve ( );

        // ## Other Operations (specified)
        // ## Operation: Extrapolate %4DD11D8903B4
        void Extrapolate ( double H , double T );

        // ## Operation: PlotToFile %4DD11DB0013D
        void PlotToFile ( bool addToExisting , std::string filename );

        // ## Get and Set Operations for Class Attributes (generated )

        // ## Attribute: name %4DD11E260370
        const std::string getName ( ) const;
        void setName ( std::string value );

        // ## Attribute: index %4DD28D200261
        const int getIndex ( ) const;
        void setIndex ( int value );
```
/// Get and Set Operations for Associations (generated)

/// Association: <unnamed>%4DD11CF2039D
/// Role: Curve::Points%4DD11CF3016B
const std::vector<Point> get_Points () const;
void set_Points (std::vector<Point> value);

// Additional Public Declarations
/// begin Curve%4DD11CC70100.public preserve=yes
/// end Curve%4DD11CC70100.public

protected:
// Additional Protected Declarations
/// begin Curve%4DD11CC70100.protected preserve=yes
/// end Curve%4DD11CC70100.protected

private:
// Additional Private Declarations
/// begin Curve%4DD11CC70100.private preserve=yes
/// end Curve%4DD11CC70100.private

public: //## implementation
// Data Members for Class Attributes
/// begin Curve::name%4DD11E260370.attr preserve=no
public: std::string {U}
std::string _Name;
/// end Curve::name%4DD11E260370.attr

/// begin Curve::index%4DD28D200261.attr preserve=no
public: int {U}
int _Index;
/// end Curve::index%4DD28D200261.attr

// Data Members for Associations

/// Association: <unnamed>%4DD11CF2039D
/// begin Curve::Points%4DD11CF3016B.role preserve=no
public: Point { -> 1..*VHgN}
std::vector<Point> Points;
/// end Curve::Points%4DD11CF3016B.role

// Additional Implementation Declarations
/// begin Curve%4DD11CC70100.implementation preserve=yes
/// end Curve%4DD11CC70100.implementation

};

/// begin Curve%4DD11CC70100.postscript preserve=yes
/// end Curve%4DD11CC70100.postscript

// Class Curve

Daniel Guitart Pagola
```cpp
//## Get and Set Operations for Class Attributes (inline)

inline const std::string Curve::getName () const
{
    //## begin Curve::getName%4DD11E260370.get preserve=no
    return _Name;
    //## end Curve::getName%4DD11E260370.get
}

inline void Curve::setName (std::string value)
{
    //## begin Curve::setName%4DD11E260370.set preserve=no
    _Name = value;
    //## end Curve::setName%4DD11E260370.set
}

inline const int Curve::getIndex () const
{
    //## begin Curve::getIndex%4DD28D200261.get preserve=no
    return _Index;
    //## end Curve::getIndex%4DD28D200261.get
}

inline void Curve::setIndex (int value)
{
    //## begin Curve::setIndex%4DD28D200261.set preserve=no
    _Index = value;
    //## end Curve::setIndex%4DD28D200261.set
}

//## Get and Set Operations for Associations (inline)

inline const std::vector<Point> Curve::get_Points () const
{
    //## begin Curve::get_Points%4DD11CF3016B.get preserve=no
    return Points;
    //## end Curve::get_Points%4DD11CF3016B.get
}

inline void Curve::set_Points (std::vector<Point> value)
{
    //## begin Curve::set_Points%4DD11CF3016B.set preserve=no
    Points = value;
    //## end Curve::set_Points%4DD11CF3016B.set
}

#ifdef
#endif
```
D.1.2 Interval.h

```cpp
/// ## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code
// generator
// versioning.
/// ## end module%1.4%.codegen_version

/// begin module%4D8C721B002A.cm preserve=no
/// %X% %Q% %Z% %W%
/// end module%4D8C721B002A.cm

/// begin module%4D8C721B002A.cp preserve=no
/// end module%4D8C721B002A.cp

/// Module: Interval%4D8C721B002A; Pseudo Package
/// specification
/// Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg\Interval.h

#ifndef Interval_h
#define Interval_h 1

/// begin module%4D8C721B002A.additionalIncludes preserve=no
/// end module%4D8C721B002A.additionalIncludes

/// begin module%4D8C721B002A.includes preserve=yes
/// end module%4D8C721B002A.includes

/// begin module%4D8C721B002A.additionalDeclarations preserve=yes
/// end module%4D8C721B002A.additionalDeclarations

/// begin Interval%4D8C721B002A.preface preserve=yes
/// end Interval%4D8C721B002A.preface

/// Class: Interval%4D8C721B002A
/// Class in charge to store the data specific to a
temperature interval. It is widely used by the methods
of the class Problem to do calculations.
/// Category: <Top Level>
/// Persistence: Transient
/// Cardinality/Multiplicity: n

class Interval
{
  /// begin Interval%4D8C721B002A.initialDeclarations preserve=yes
  /// end Interval%4D8C721B002A.initialDeclarations
```

Daniel Guitart Pagola
public:
    //## Constructors (specified)
    //## Operation: Interval%4D8C923800E2
    // Class constructor. DT is calculated using Te
    and Ts
    Interval (double Ts, double Te, double FCp);

    //## Destructor (generated)
    ~Interval();

    //## Other Operations (specified)
    //## Operation: calculateHFlux%4D944B510260
    // Calculates the enthalpy flux in an interval (DT * FCp)
    double calculateHFlux();

    //## Get and Set Operations for Class Attributes (generated)

    //## Attribute: DT%4D8C91E3022C
    // Stores the temperature difference between the
    // start and
    // the end of an interval
    const double getdT () const;
    void setdT (double value);

    //## Attribute: Fcp%4D8C91FB0247
    // Stores the FCp of an interval (FCp stands for F*
    // Cp, that
    // means massic flux * specific heat).
    // Defaults at 0, and his real value for a
    // specific problem
    // is found with the method calculateFCpSum()
    // using all the
    // streams that belong to the interval.
    const double getfcp () const;
    void setfcp (double value);

    //## Attribute: Ts%4D9340090050
    // Stores the starting temperature of the interval
    // (the
    // lowest)
    const double getts () const;
    void setts (double value);

    //## Attribute: Te%4D934011029F
    // Stores the final temperature of the interval (the
    // highest)
    const double gette () const;
```cpp
void sette (double value);

// Additional Public Declarations
// ### begin Interval%4D8C721B002A.public preserve=yes
// ### end Interval%4D8C721B002A.public

protected:

// Additional Protected Declarations
// ### begin Interval%4D8C721B002A.protected preserve=yes
// ### end Interval%4D8C721B002A.protected

private:

// Additional Private Declarations
// ### begin Interval%4D8C721B002A.private preserve=yes
// ### end Interval%4D8C721B002A.private

private: //### implementation

// Data Members for Class Attributes

    //### begin Interval::DT%4D8C91E3022C.attr preserve=no
    public: double {U}
    double _dT;
    //### end Interval::DT%4D8C91E3022C.attr

    //### begin Interval::Fcp%4D8C91FB0247.attr preserve=no
    public: double {U} 0
    double _fcp;
    //### end Interval::Fcp%4D8C91FB0247.attr

    //### begin Interval::Ts%4D9340090050.attr preserve=no
    public: double {U}
    double _ts;
    //### end Interval::Ts%4D9340090050.attr

    //### begin Interval::Te%4D934011029F.attr preserve=no
    public: double {U}
    double _te;
    //### end Interval::Te%4D934011029F.attr

    // Additional Implementation Declarations
    //### begin Interval%4D8C721B002A.implementation preserve=yes
    //### end Interval%4D8C721B002A.implementation

};

//### begin Interval%4D8C721B002A.postscript preserve=yes
//### end Interval%4D8C721B002A.postscript

// Class Interval

//### Get and Set Operations for Class Attributes (inline)
```
inline const double Interval::getdT () const
{
    // ## begin Interval::getdT %4 D8C91E3022C . get preserve=no
    return _dT;
    // ## end Interval::getdT %4 D8C91E3022C . get
}

inline void Interval::setdT (double value)
{
    // ## begin Interval::setdT %4 D8C91E3022C . set preserve=no
    _dT = value;
    // ## end Interval::setdT %4 D8C91E3022C . set
}

inline const double Interval::getfcp () const
{
    // ## begin Interval::getfcp %4 D8C91FB0247 . get preserve=no
    return _fcp;
    // ## end Interval::getfcp %4 D8C91FB0247 . get
}

inline void Interval::setfcp (double value)
{
    // ## begin Interval::setfcp %4 D8C91FB0247 . set preserve=no
    _fcp = value;
    // ## end Interval::setfcp %4 D8C91FB0247 . set
}

inline const double Interval::getts () const
{
    // ## begin Interval::getts %4 D9340090050 . get preserve=no
    return _ts;
    // ## end Interval::getts %4 D9340090050 . get
}

inline void Interval::setts (double value)
{
    // ## begin Interval::setts %4 D9340090050 . set preserve=no
    _ts = value;
    // ## end Interval::setts %4 D9340090050 . set
}

inline const double Interval::gette () const
{
    // ## begin Interval::gette %4 D934011029F . get preserve=no
    return _te;
    // ## end Interval::gette %4 D934011029F . get
}

inline void Interval::sette (double value)
{
D.1.3 IntervalH.h

```cpp
//## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
// versioning.
//## end module%1.4%.codegen_version

//## begin module%4DCA4DB700A8.cm preserve=no
// %X% %Q% %Z% %W%
//## end module%4DCA4DB700A8.cm

//## begin module%4DCA4DB700A8.cp preserve=no
//## end module%4DCA4DB700A8.cp

//## Module: IntervalH%4DCA4DB700A8; Pseudo Package specification
//## Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\IntervalH.h

#ifndef IntervalH_h
#define IntervalH_h

//## begin module%4DCA4DB700A8.additionalIncludes preserve=no
//## end module%4DCA4DB700A8.additionalIncludes

//## begin module%4DCA4DB700A8.includes preserve=yes
#include <vector>
#include "problem.h"
//## end module%4DCA4DB700A8.includes

class Problem;

//## begin module%4DCA4DB700A8.additionalDeclarations preserve=yes
//## end module%4DCA4DB700A8.additionalDeclarations
```

Daniel Guitart Pagola
class IntervalH
{
    //## begin IntervalH%4DCA4DB700A8.initialDeclarations
    //## end IntervalH%4DCA4DB700A8.initialDeclarations

    public:
    //## Constructors (specified)
    //## Operation: IntervalH%4DCA559F0375
    IntervalH (Problem* pb, double TColdIn, double TColdOut, double THotOut, double THotIn, double hLeft, double hRight);

    //## Destructor (generated)
    ~IntervalH();

    //## Other Operations (specified)
    //## Operation: FindDTlm%4DCA573A024E
    void FindDTlm ();

    //## Operation: FindSumQh%4DD5021303E1
    void FindSumQh ();

    //## Operation: FindArea%4DD4FFAC0056
    void FindArea ();

    //## Get and Set Operations for Class Attributes (generated)

    //## Attribute: TColdIn%4DCA52F50390
    const double getTColdIn () const;
    void settColdIn (double value);

    //## Attribute: TColdOut%4DCA53170101
    const double getTColdOut () const;
    void settColdOut (double value);

    //## Attribute: THotOut%4DCA531D0211
    const double getTHotOut () const;
    void settHotOut (double value);
/// Attribute: THotIn %4DCA532302B3
const double getHotIn () const;
void setHotIn (double value);

/// Attribute: DTlm %4DCA532E009A
const double getdTlm () const;
void setdTlm (double value);

/// Attribute: hLeft %4DCA57AE01E1
const double getHLeft () const;
void setHLeft (double value);

/// Attribute: hRight %4DCA57B701A8
const double getHRight () const;
void setHRight (double value);

/// Attribute: Area %4DD4FF5C0038
const double getarea () const;
void setarea (double value);

/// Attribute: QhHotStreams %4DD5023903D7
const double getqhHotStreams () const;
void setqhHotStreams (double value);

/// Attribute: QhColdStreams %4DD5024D034F
const double getqhColdStreams () const;
void setqhColdStreams (double value);

/// Get and Set Operations for Associations (generated)

/// Association: <unnamed> %4DCA5566036E
/// Role: IntervalH::pb %4DCA55670362
const Problem * get_pb () const;
void set_pb (Problem * value);

// Additional Public Declarations
/// begin IntervalH %4DCA4DB700A8 . public preserve = yes
/// end IntervalH %4DCA4DB700A8 . public

protected:
// Additional Protected Declarations
/// begin IntervalH %4DCA4DB700A8 . protected preserve = yes
/// end IntervalH %4DCA4DB700A8 . protected

private:
// Additional Private Declarations
/// begin IntervalH %4DCA4DB700A8 . private preserve = yes
/// end IntervalH %4DCA4DB700A8 . private

private: /// implementation
// Data Members for Class Attributes
Data Members for Associations

```java
//## begin IntervalH::TColdIn%4DCA52F50390.attr preserve=
   no public: double {U}
   double _tColdIn;
//## end IntervalH::TColdIn%4DCA52F50390.attr

//## begin IntervalH::TColdOut%4DCA53170101.attr preserve=
   no public: double {U}
   double _tColdOut;
//## end IntervalH::TColdOut%4DCA53170101.attr

//## begin IntervalH::THotOut%4DCA531D0211.attr preserve=
   no public: double {U}
   double _tHotOut;
//## end IntervalH::THotOut%4DCA531D0211.attr

//## begin IntervalH::THotIn%4DCA532302B3.attr preserve=
   no public: double {U}
   double _tHotIn;
//## end IntervalH::THotIn%4DCA532302B3.attr

//## begin IntervalH::DTlm%4DCA532E009A.attr preserve=
   no public: double {U}
   double _dTlm;
//## end IntervalH::DTlm%4DCA532E009A.attr

//## begin IntervalH::hLeft%4DCA57AE01E1.attr preserve=
   no public: double {U}
   double _HLeft;
//## end IntervalH::hLeft%4DCA57AE01E1.attr

//## begin IntervalH::hRight%4DCA57B701A8.attr preserve=
   no public: double {U}
   double _HRight;
//## end IntervalH::hRight%4DCA57B701A8.attr

//## begin IntervalH::Area%4DD4FF5C0038.attr preserve=
   no public: double {U}
   double _area;
//## end IntervalH::Area%4DD4FF5C0038.attr

//## begin IntervalH::QhHotStreams%4DD5023903D7.attr
   preserve=no public: double {U}
   double _qhHotStreams;
//## end IntervalH::QhHotStreams%4DD5023903D7.attr

//## begin IntervalH::QhColdStreams%4DD5024D034F.attr
   preserve=no public: double {U}
   double _qhColdStreams;
//## end IntervalH::QhColdStreams%4DD5024D034F.attr

// Data Members for Associations
```

Daniel Guitart Pagola
```cpp
//## Association: <unnamed>%4DCA5566036E
//## begin IntervalH::pb%4DCA55670362.role preserve=no
    public: Problem {1..* -> 1RFHGK}
    Problem *pb;
//## end IntervalH::pb%4DCA55670362.role

// Additional Implementation Declarations
//## begin IntervalH%4DCA4DB700A8.implementation preserve=yes
//## end IntervalH%4DCA4DB700A8.implementation

};

//## begin IntervalH%4DCA4DB700A8.postscript preserve=yes
//## end IntervalH%4DCA4DB700A8.postscript

// Class IntervalH

/// Get and Set Operations for Class Attributes (inline)

inline const double IntervalH::gettColdIn () const
{
    //## begin IntervalH::gettColdIn%4DCA52F50390.get preserve=no
    return _tColdIn;
    //## end IntervalH::gettColdIn%4DCA52F50390.get
}

inline void IntervalH::settColdIn (double value)
{
    //## begin IntervalH::settColdIn%4DCA52F50390.set preserve=no
    _tColdIn = value;
    //## end IntervalH::settColdIn%4DCA52F50390.set
}

inline const double IntervalH::gettColdOut () const
{
    //## begin IntervalH::gettColdOut%4DCA53170101.get preserve=no
    return _tColdOut;
    //## end IntervalH::gettColdOut%4DCA53170101.get
}

inline void IntervalH::settColdOut (double value)
{
    //## begin IntervalH::settColdOut%4DCA53170101.set preserve=no
    _tColdOut = value;
    //## end IntervalH::settColdOut%4DCA53170101.set
}

inline const double IntervalH::gettHotOut () const
{
    //## begin IntervalH::gettHotOut%4DCA59E9593F.get preserve=no
    return _tHotOut;
    //## end IntervalH::gettHotOut%4DCA59E9593F.get
}
```
```cpp
inline void IntervalH::setHotOut (double value) {
    _tHotOut = value;
}
inline const double IntervalH::getHotIn () const {
    return _tHotIn;
}
inline void IntervalH::setHotIn (double value) {
    _tHotIn = value;
}
inline const double IntervalH::getdTlm () const {
    return _dTlm;
}
inline void IntervalH::setdTlm (double value) {
    _dTlm = value;
}
inline const double IntervalH::getHLeft () const {
    return _HLeft;
}
inline void IntervalH::setHLeft (double value) {
    _HLeft = value;
}
```
inline const double IntervalH::getHRight () const
{
    // ## begin IntervalH::getHRight %4 DCA57B701A8 . get preserve = no
    return _HRight ;
    // ## end IntervalH::getHRight %4 DCA57B701A8 . get
}

inline void IntervalH::setHRight ( double value )
{
    // ## begin IntervalH::setHRight %4 DCA57B701A8 . set preserve = no
    _HRight = value ;
    // ## end IntervalH::setHRight %4 DCA57B701A8 . set
}

inline const double IntervalH::getarea () const
{
    // ## begin IntervalH::getarea %4 DD4FF5C0038 . get preserve = no
    return _area ;
    // ## end IntervalH::getarea %4 DD4FF5C0038 . get
}

inline void IntervalH::setarea ( double value )
{
    // ## begin IntervalH::setarea %4 DD4FF5C0038 . set preserve = no
    _area = value ;
    // ## end IntervalH::setarea %4 DD4FF5C0038 . set
}

inline const double IntervalH::getqhHotStreams () const
{
    // ## begin IntervalH::getqhHotStreams %4 DD5023903D7 . get preserve = no
    return _qhHotStreams ;
    // ## end IntervalH::getqhHotStreams %4 DD5023903D7 . get
}

inline void IntervalH::setqhHotStreams ( double value )
{
    // ## begin IntervalH::setqhHotStreams %4 DD5023903D7 . set preserve = no
    _qhHotStreams = value ;
    // ## end IntervalH::setqhHotStreams %4 DD5023903D7 . set
}

inline const double IntervalH::getqhColdStreams () const
{
    // ## begin IntervalH::getqhColdStreams %4 DD5024D034F . get preserve = no
    return _qhColdStreams ;
    // ## end IntervalH::getqhColdStreams %4 DD5024D034F . get
inline void IntervalH::setqhColdStreams (double value) {
    // ## begin IntervalH::setqhColdStreams %4 DD5024D034F . set
    _qhColdStreams = value;
    // ## end IntervalH::setqhColdStreams %4 DD5024D034F . set
}

// ## Get and Set Operations for Associations (inline)
inline const Problem * IntervalH::get_pb () const {
    // ## begin IntervalH::get_pb %4 DCA55670362 . get
    return pb;
    // ## end IntervalH::get_pb %4 DCA55670362 . get
}
inline void IntervalH::set_pb (Problem * value) {
    // ## begin IntervalH::set_pb %4 DCA55670362 . set
    pb = value;
    // ## end IntervalH::set_pb %4 DCA55670362 . set
}

D.1.4 PinchDlg.h

#ifndef __AFXWIN_H__
#define AFX_PINCHDLG_H__4684CE99_E32A_4109_96C5_6C43AE9DFFD5__INCLUDED_
#endif

#pragma once

#ifndef __AFXWIN_H__
#define __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH

#include "resource.h"    // main symbols

_measurement
Measurement spca();

PCH

class CPinchDlgApp : public CWinApp
{
    public:
        CPinchDlgApp();

    // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPinchDlgApp)
        public:
            virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

    // Implementation

    //{{AFX_MSG(CPinchDlgApp)
        DECLARE_MESSAGE_MAP()
    //}}AFX_MSG
};

D.1.5 PinchDlgDlg.h

// PinchDlgDlg.h : header file
//
#ifndef AFX_PINCHDLGDLG_H__BF42688E_A7ED_4EC6_A028_CB6689772744__INCLUDED_
#define AFX_PINCHDLGDLG_H__BF42688E_A7ED_4EC6_A028_CB6689772744__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
#include "Problem.h"

//////////////////////////////////////////////////////////////////////
// CPinchDlgDlg dialog

class CPinchDlgDlg : public CDialog
{
// Construction
public:
CPinchDlgDlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
//{{AFX_DATA(CPinchDlgDlg)
enum { IDD = IDD_PINCHDLG_DIALOG };  
CButton m_buttonCostOpen;  
CButton m_buttonCostClean;  
CButton m_buttonCostAdd;  
CButton m_buttonTcost;  
CButton m_buttonCurvePoints;  
CButton m_buttonPlotCC;  
CButton m_buttonPlotBGCC;  
CButton m_buttonPlotBCC;  
CButton m_buttonPlotGCC;  
CButton m_calculatePinchAndMER;  
CButton m_loadData;  
double m_DTmin;  
double m_TPinch;  
double m_HotMER;  
double m_ColdMER;  
double m_targetArea;  
CString m_PointsType;  
double m_coef_a;  
double m_coef_b;  
double m_coef_c;  
double m_targetCost;  
double m_interest;  
double m_years;  
double m_annualCost;  
double m_costutilitycold;
//}}AFX_DATA

Daniel Guitart Pagola
double m_costutilityhot;
double m_totalcost;
double m.utilitiescost;
int m_.coefsType;
BOOL m_utilitiesCheck;
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//}}AFX_VIRTUAL(CPinchDlgDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);
//}}AFX_VIRTUAL

// Implementation

protected:
HICON m_hIcon;

Problem * PROBLEM ;

//}}AFX_INSERT_LOCATION

// Generated message map functions
//}}AFX_MSG(CPinchDlgDlg)
virtual BOOL OnInitDialog();
afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
afx_msg void OnPaint();
afx_msg HCURSOR OnQueryDragIcon();
afx_msg void OnChangeEditTPinch();
afx_msg void OnPlot();
afx_msg void OnButtonPlot();
afx_msg void OnButtonPlotcc();
afx_msg void OnButtonLoaddata();
afx_msg void OnExit();
afx_msg void OnButtonPlotbcc();
afx_msg void OnButtonHintervals();
afx_msg void OnButtonTcost();
afx_msg void OnButtonStreams();
afx_msg void OnButtonUtilities();
afx_msg void OnButtonPointsgraph();
afx_msg void OnButtonCostfileopen();
afx_msg void OnButtonCostfileadd();
afx_msg void OnButtonCostfileclean();
afx_msg void OnSelchangeComboCoefselect();
afx_msg void OnButtonChoosedir();
afx_msg void OnCheckUtilities();
afx_msg void OnButtonCalculatepinch();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
D.1.6 Point.h

```cpp
#ifndef Point_h
#define Point_h

// ## begin module %4 DD11CD00173 . additionalDeclarations preserve = yes
// ## end module %4 DD11CD00173 . additionalDeclarations

#include <vector>

// ## begin module %4 DD11CD00173 . includes preserve = yes
// ## end module %4 DD11CD00173 . includes

class Point {

}
```

Daniel Guitart Pagola
public:
    /// Constructors (specified)
    /// Operation: Point%4DD2764202DC
    Point (double H, double T);

    /// Destructor (generated)
    ~Point();

    /// Get and Set Operations for Class Attributes (generated)

    /// Attribute: H%4DD11D5F02AD
    const double getH () const;
    void setH (double value);

    /// Attribute: T%4DD11D6D0391
    const double getT () const;
    void setT (double value);

protected:
    /// Additional Protected Declarations

private:
    /// Additional Private Declarations

private: //### implementation
    /// Data Members for Class Attributes

    /// begin Point::H%4DD11D5F02AD.attr preserve=no public
    : double {_h};//### end Point::H%4DD11D5F02AD.attr

    /// begin Point::T%4DD11D6D0391.attr preserve=no public
    : double {_t};//### end Point::T%4DD11D6D0391.attr

    /// Additional Implementation Declarations
    /// begin Point%4DD11CD00173.implementation preserve=yes
## Get and Set Operations for Class Attributes (inline)

```cpp
inline const double Point::geth () const
{
    // ## begin Point::geth %4DD11D5F02AD . get preserve=no
    return _h;
    // ## end Point::geth %4DD11D5F02AD . get
}

inline void Point::seth (double value)
{
    // ## begin Point::seth %4DD11D5F02AD . set preserve=no
    _h = value;
    // ## end Point::seth %4DD11D5F02AD . set
}

inline const double Point::gett () const
{
    // ## begin Point::gett %4DD11D6D0391 . get preserve=no
    return _t;
    // ## end Point::gett %4DD11D6D0391 . get
}

inline void Point::sett (double value)
{
    // ## begin Point::sett %4DD11D6D0391 . set preserve=no
    _t = value;
    // ## end Point::sett %4DD11D6D0391 . set
}

// ## end Point%4DD11CD00173 .implementation
```

### D.1.7 Problem.h

```cpp
// ## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
```
## Problem %4 D89B5CD0251

This class is the core of the program. It is responsible of calling the methods to search for the problem solution. It also stores values later used in the MFC app.

### Category: <Top Level>

### Persistence: Transient

### Cardinality/Multiplicity: n

```cpp
class Problem {
```

Daniel Guitart Pagola
//## begin Problem%4D89B5CD0251.initialDeclarations preserve=yes
//## end Problem%4D89B5CD0251.initialDeclarations

public:
//## Constructors (generated)
Problem();

//## Destructor (generated)
~Problem();

//## Other Operations (specified)
//## Operation: readData%4D907A410344
// Method where the data is read from the streams.
txt and utilities.txt file and stored in the Stream class
void readData();

//## Operation: calculateProblemArray%4D907ABC0002
// Method where the data is sorted by temperature to create
// the intervals for the calculation of the MER (and the GCC). The data is stored in the Interval Class.
void calculateProblemArray();

//## Operation: defineProblem%4D89B5ED03D8
// Method that calls the readData method. Doing so defines
// the problem (and changes the ProblemDefined variable),
// so other methods can be used.
void defineProblem();

//## Operation: calculateFCpSum%4D9047EE0045
// Method that adds the FCp on each interval based on the streams that belong to it.
void calculateFCpSum();

//## Operation: calculateTPinch%4D904824003F
// Method that calculates the problem Pinch temperature and
// also the hot utility.
void calculateTPinch();

//## Operation: calculateCascade%4D90483500DD
// Method that calculates the cold utility, the final T-H
// cascade, and stores the points in Curves[0],
// which
// corresponds to the GCC curve. It also calls
// PlotToFile
// in Curve[0] so the new points are stored in a
data file,
// so they can be used by a graphical app
// afterwards.
void calculateCascade();

// Method that the user calls to calculate both
// MER and
// inevitably also the points from the GCC. Can
// only be
// called if the problem is defined. It calls all
// the
// methods involved in calculating the MER.
void calculateMER();

// Method that calculates the points from the
// composite
// curves and dumps them into the Curve class.
//
// Curves[1] and Curves[3] store the cold CC and
// BCC
// Curves[2] and Curves[4] store the hot CC and
// BCC
// It also calls the PlotToFile function in each
curve so
// the points are stored in a file and can be
graphed
// afterwards.
// The method can only be called if the problem is
defined
// and should be called after calculating the MER
// so the
// cold curve is already at a distance DTmin from
// the hot
// curve (otherwise it will start at H=0)
void calculateCCpoints();

// Method that uses information stored in the BCC
curves to
// calculate intervals of enthalpy. The easiest
way to
// describe a is this: if one looks at the BCC
curves from
// left to right an interval of enthalpy occurs each time a
// new point on a curve is reached, whether it has a point
// in the other the other curve at the same enthalpy or not.
// It is necessary to store data this way because to find a
// solution we need DTlm, and to find it the four temperatures in each of this intervals are needed
void CalculateEnthalpyIntervals();

//## Operation: CalculateTargetArea%4DD51568008D
// Method that calculates the target area of the exchanger
// network. It is an approximation but practice tells us that the real network is under 10% away from the estimated value, unless the film coefficients of the cold vs the hot streams are more than an order of magnitude away, in such case the error might go up to 30%.
void CalculateTargetArea();

//## Operation: CalculateTargetCost%4DDE6AAC0287
// Method that calculates the cost of the network of exchangers based on the area. The relation between the area and the cost is controlled by three coefficients that the user inputs based on experience or existing data.
void CalculateTargetCost (double a, double b, double c);

//## Operation: FindNumExchangers%4DDF7CDC004E
// Finds the minimum number of exchangers used in a network, based on the approximation given that if there is a pinch point, the number of exchangers will be N=(Sa−1)+(Sb−1) where Sa and Sb are the number of streams that are above and below pinch temperature.
void FindNumExchangers();

//## Operation: CalculateInterest
/**
 * Function that returns the coefficient which will be multiplied to cost if we consider this capital cost
 * annualized at 'i' interest for 'n' years.
 */
double CalculateInterest(double i, double n);

//## Operation: CalculateInteriorCurves
/**
 * The goal of this function is that if we got 2 curves
 * like this example, we recover the only the X area they both share and we store this new trimmed curves
 */
void CalculateInteriorCurves();

//## Operation: RetrievePaths
void RetrievePaths();

//## Get and Set Operations for Class Attributes (generated)

//## Attribute: DtMin
/**
 * Stores the DTmin of the problem.
 */
const double getdtMin() const;
void setdtMin(double value);

//## Attribute: HotMER
/**
 * Stores the hot utility minimum energy requirement
 */
const double gethotMER() const;
void sethotMER(double value);

//## Attribute: ColdMER
/**
 * Stores the cold utility minimum energy requirement
 */
const double getcoldMER() const;
void setcoldMER(double value);
/// Attribute: NColdStreams%4D907F0702D6
/// Stores the number of cold streams in the problem.
const int getnColdStreams () const;
void setnColdStreams (int value);

/// Attribute: NHotStreams%4D907F0E0100
/// Stores the number of Hot Streams in the problem
const int getnHotStreams () const;
void setnHotStreams (int value);

/// Attribute: ProblemDefined%4D92F5890119
/// Control variable that assures that the problem is defined before calling methods that require so.
const bool getproblemDefined () const;
void setproblemDefined (bool value);

/// Attribute: NIntervals%4D933F310160
/// Stores the number of temperature intervals in the problem (for the GCC calculation)
const int getnIntervals () const;
void setnIntervals (int value);

/// Attribute: TPinch%4D944C880059
/// Stores the Pinch temperature of the problem.
const double gettPinch () const;
void settPinch (double value);

/// Attribute: NUtilitiesCold%4DC7D8BC0013
/// Stores the amount of cold utilities.
const int getnUtilitiesCold () const;
void setnUtilitiesCold (int value);

/// Attribute: NUtilitiesHot%4DC947E2037A
/// Stores the amount of hot utilities.
const int getnUtilitiesHot () const;
void setnUtilitiesHot (int value);

/// Attribute: IncludeUtilities%4DC94D06036D
/// A boolean that tells the program if we will include utilities in the calculations or not. If utilities are not included, calculations regarding area of exchangers and therefore cost will be done using only the range of the curves that is totally internal to the process. To
// calculate this new curves inside the range, 
    Calculate
    // InteriorCurves() will be called.
    const bool getincludeUtilities () const;
    void setincludeUtilities (bool value);

    /// Attribute: TargetArea%4DD6705E025B
    // The targeted area of the network of exchangers.
    It is an
    // estimation of the real one.
    const double gettargetArea () const;
    void settargetArea (double value);

    /// Attribute: NExchangers%4DDE572E01F6
    // Stores the minimum number of exchangers 
    // required in the
    // network.
    const int getnExchangers () const;
    void setnExchangers (int value);

    /// Attribute: TargetCost%4DDE574B01D3
    // Stores the targeted cost of the network. It 
    // shares the 
    // same criteria as the target area as the co
    const double gettargetCost () const;
    void settargetCost (double value);

    /// Attribute: OctavePath%4DEF738E038F
    // Contains the path to GNU Octave bin folder, in 
    // order to 
    // be able to use GNUplot for plotting graphs 
    // 
    // The string contains escape secuences (\ 
    // character)
    // before the \ and " characters, as it is 
    // required by the 
    // system.
    // 
    // Defaults at 
    "C:\\Octave\\3.2.4_gcc-4.4.0\\
    bin"
    // the directory is 
    "C:\\Octave\\3.2.4_gcc-4.4.0\\
    bin"
    const std::string getoctavePath () const;
    void setoctavePath (std::string value);

    /// Attribute: ProgramPath%4DEF75390375
    const std::string getprogramPath () const;
    void setprogramPath (std::string value);

    /// Get and Set Operations for Associations (generated)

    /// Association: <unnamed>%4DD11CEF032E
const std::vector<Curve> get_Curves () const;
void set_Curves (std::vector<Curve> value);

// Additional Public Declarations
// ## begin Problem%4D89B5CD0251.public preserve=yes
// ## end Problem%4D89B5CD0251.public

protected:
// Additional Protected Declarations
// ## begin Problem%4D89B5CD0251.protected preserve=yes
// ## end Problem%4D89B5CD0251.protected

private:
// Additional Private Declarations
// ## begin Problem%4D89B5CD0251.private preserve=yes
// ## end Problem%4D89B5CD0251.private

public: // ## implementation
// Data Members for Class Attributes

// ## begin Problem::DtMin%4D89B8530222.attr preserve=no
public: double {U}
double _dtMin;
// ## end Problem::DtMin%4D89B8530222.attr

// ## begin Problem::HotMER%4D904893011D.attr preserve=no
public: double {U}
double _hotMER;
// ## end Problem::HotMER%4D904893011D.attr

// ## begin Problem::ColdMER%4D90495001B6.attr preserve=no
public: double {U}
double _coldMER;
// ## end Problem::ColdMER%4D90495001B6.attr

// ## begin Problem::NColdStreams%4D907F07202D6.attr preserve=no public: int {U}
int _nColdStreams;
// ## end Problem::NColdStreams%4D907F07202D6.attr

// ## begin Problem::NHotStreams%4D907F0E0100.attr preserve=no public: int {U}
int _nHotStreams;
// ## end Problem::NHotStreams%4D907F0E0100.attr

// ## begin Problem::ProblemDefined%4D92F5890119.attr preserve=no public: bool {U}
bool _problemDefined;
// ## end Problem::ProblemDefined%4D92F5890119.attr

Daniel Guitart Pagola
Master Thesis - Development of a software dedicated to pinch analysis

Daniel Guitart Pagola
```cpp
public: Stream { -> 1..* VHgN }
```

```cpp
public: Interval { -> 1..* VHgN }
```

```cpp
public: Stream { -> 1..* VHgN }
```

```cpp
public: Interval { -> 1..* VHgN }
```

```cpp
};
```

```cpp
inline const double Problem::getdtMin () const
{
```

```cpp
//## begin Problem%4D89B5CD0251.postscript preserve=yes
//## end Problem%4D89B5CD0251.postscript
```

```cpp
// Class Problem
```

```cpp
//## Get and Set Operations for Class Attributes (inline)
```

```cpp
inline const double Problem::getdtMin%4D89B5CD0251.postscript preserve=no
return _dtMin;
/// ## end Problem::getdtMin4D89B8530222.get
}

inline void Problem::setdtMin (double value) {
/// ## begin Problem::setdtMin4D89B8530222.set preserve=no
Dt_MIN = value;
/// ## end Problem::setdtMin4D89B8530222.set
}

inline const double Problem::gethotMER () const {
/// ## begin Problem::gethotMER%4D904893011D.get preserve=no
return _hotMER;
/// ## end Problem::gethotMER%4D904893011D.get
}

inline void Problem::sethotMER (double value) {
/// ## begin Problem::sethotMER%4D904893011D.set preserve=no
_hotMER = value;
/// ## end Problem::sethotMER%4D904893011D.set
}

inline const double Problem::getcoldMER () const {
/// ## begin Problem::getcoldMER%4D90495001B6.get preserve=no
return _coldMER;
/// ## end Problem::getcoldMER%4D90495001B6.get
}

inline void Problem::setcoldMER (double value) {
/// ## begin Problem::setcoldMER%4D90495001B6.set preserve=no
_coldMER = value;
/// ## end Problem::setcoldMER%4D90495001B6.set
}

inline const int Problem::getnColdStreams () const {
/// ## begin Problem::getnColdStreams%4D907F0702D6.get preserve=no
return _nColdStreams;
/// ## end Problem::getnColdStreams%4D907F0702D6.get
}

inline void Problem::setnColdStreams (int value) {
/// ## begin Problem::setnColdStreams%4D907F0702D6.set preserve=no
_nColdStreams = value;
}
inline const int Problem::getnHotStreams () const
{
    // ## begin Problem::getnHotStreams%4D907F0E0100.get preserve=
    return _nHotStreams;
    // ## end Problem::getnHotStreams%4D907F0E0100.get
}

inline void Problem::setnHotStreams (int value)
{
    // ## begin Problem::setnHotStreams%4D907F0E0100.set preserve=
    _nHotStreams = value;
    // ## end Problem::setnHotStreams%4D907F0E0100.set
}

inline const bool Problem::getproblemDefined () const
{
    // ## begin Problem::getproblemDefined%4D92F5890119.get preserve=
    return _problemDefined;
    // ## end Problem::getproblemDefined%4D92F5890119.get
}

inline void Problem::setproblemDefined (bool value)
{
    // ## begin Problem::setproblemDefined%4D92F5890119.set preserve=
    _problemDefined = value;
    // ## end Problem::setproblemDefined%4D92F5890119.set
}

inline const int Problem::getnIntervals () const
{
    // ## begin Problem::getnIntervals%4D933F310160.get preserve=
    return _nIntervals;
    // ## end Problem::getnIntervals%4D933F310160.get
}

inline void Problem::setnIntervals (int value)
{
    // ## begin Problem::setnIntervals%4D933F310160.set preserve=
    _nIntervals = value;
    // ## end Problem::setnIntervals%4D933F310160.set
}

inline const double Problem::gettPinch () const
```cpp
{   ///< Problem :: gettPinch %4 D944C880059. get preserve=no
    return _tPinch;
    ///</Problem :: gettPinch %4 D944C880059. get
}

inline void Problem :: settPinch (double value) {
    ///< Problem :: settPinch %4 D944C880059. set preserve=no
    _tPinch = value;
    ///</Problem :: settPinch %4 D944C880059. set
}

inline const int Problem :: getnUtilitiesCold () const {
    ///< Problem :: getnUtilitiesCold %4 DC7D8BC0013 . get
    return _nUtilitiesCold;
    ///</Problem :: getnUtilitiesCold %4 DC7D8BC0013 . get
}

inline void Problem :: setnUtilitiesCold (int value) {
    ///< Problem :: setnUtilitiesCold %4 DC7D8BC0013 . set
    _nUtilitiesCold = value;
    ///</Problem :: setnUtilitiesCold %4 DC7D8BC0013 . set
}

inline const int Problem :: getnUtilitiesHot () const {
    ///< Problem :: getnUtilitiesHot %4 DC947E2037A . get
    return _nUtilitiesHot;
    ///</Problem :: getnUtilitiesHot %4 DC947E2037A . get
}

inline void Problem :: setnUtilitiesHot (int value) {
    ///< Problem :: setnUtilitiesHot %4 DC947E2037A . set
    _nUtilitiesHot = value;
    ///</Problem :: setnUtilitiesHot %4 DC947E2037A . set
}

inline const bool Problem :: getincludeUtilities () const {
    ///< Problem :: getincludeUtilities %4 DC94D06036D . get
    return _includeUtilities;
    ///</Problem :: getincludeUtilities %4 DC94D06036D . get
}
}  

Daniel Guitart Pagola
```
inline void Problem::setincludeUtilities (bool value)
{
    // ## begin Problem::setincludeUtilities%4DC94D06036D . set
    _includeUtilities = value;
    // ## end Problem::setincludeUtilities%4DC94D06036D . set
}

inline const double Problem::gettargetArea () const
{
    // ## begin Problem::gettargetArea%4DD6705E025B . get preserve= no
    return _targetArea;
    // ## end Problem::gettargetArea%4DD6705E025B . get
}

inline void Problem::settargetArea (double value)
{
    // ## begin Problem::settargetArea%4DD6705E025B . set preserve= no
    _targetArea = value;
    // ## end Problem::settargetArea%4DD6705E025B . set
}

inline const int Problem::getnExchangers () const
{
    // ## begin Problem::getnExchangers%4DDE572E01F6 . get preserve= no
    return _nExchangers;
    // ## end Problem::getnExchangers%4DDE572E01F6 . get
}

inline void Problem::setnExchangers (int value)
{
    // ## begin Problem::setnExchangers%4DDE572E01F6 . set preserve= no
    _nExchangers = value;
    // ## end Problem::setnExchangers%4DDE572E01F6 . set
}

inline const double Problem::gettargetCost () const
{
    // ## begin Problem::gettargetCost%4DDE574B01D3 . get preserve= no
    return _targetCost;
    // ## end Problem::gettargetCost%4DDE574B01D3 . get
}

inline void Problem::settargetCost (double value)
//## begin Problem::settargetCost%4DDE574B01D3.set preserve=
_no_targetCost = value;
//## end Problem::settargetCost%4DDE574B01D3.set

inline const std::string Problem::getoctavePath () const
{
    //## begin Problem::getoctavePath%4DEF738E038F.get preserve=
    _octavePath;
    //## end Problem::getoctavePath%4DEF738E038F.get
}

inline void Problem::setoctavePath (std::string value)
{
    //## begin Problem::setoctavePath%4DEF738E038F.set preserve=
    _octavePath = value;
    //## end Problem::setoctavePath%4DEF738E038F.set
}

inline const std::string Problem::getprogramPath () const
{
    //## begin Problem::getprogramPath%4DEF75390375.get preserve=
    _programPath;
    //## end Problem::getprogramPath%4DEF75390375.get
}

inline void Problem::setprogramPath (std::string value)
{
    //## begin Problem::setprogramPath%4DEF75390375.set preserve=
    _programPath = value;
    //## end Problem::setprogramPath%4DEF75390375.set

    //## Get and Set Operations for Associations (inline)

    inline const std::vector<Curve> Problem::get_Curves () const
    {
        //## begin Problem::get_Curves%4DD11CF000DC.get preserve=
        Curves;
        //## end Problem::get_Curves%4DD11CF000DC.get
    }

    inline void Problem::set_Curves (std::vector<Curve> value)
    {
        //## begin Problem::set_Curves%4DD11CF000DC.set preserve=
        Curves = value;
        //## end Problem::set_Curves%4DD11CF000DC.set

    Daniel Guitart Pagola
D.1.8 resource.h

[Strings and definitions related to resource handling in the software project]
D.1.9 StdAfx.h

// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but
// are changed infrequently
//
#if !defined (AFX_STDAFX_H__92EA7EC2_B37A_44C5_907D_E109E052F448__INCLUDED_)
#define AFX_STDAFX_H__92EA7EC2_B37A_44C5_907D_E109E052F448__INCLUDED_ 1
#endif

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN // Exclude rarely-used stuff from Windows headers
#include <afxwin.h> // MFC core and standard components
#include <afxext.h> // MFC extensions
#include <afxdisp.h> // MFC Automation classes
#include <afxdtctl.h> // MFC support for Internet Explorer 4 Common Controls
#ifndef _AFX_NO_AFXCMN_SUPPORT
#define AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h> // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.
#endif // !defined(AFX_STDafx_H__92EA7EC2_B37A_44C5_907D_E109E052F448__INCLUDED_)

D.1.10 Stream.h

//## begin module %1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
// versioning.
//## end module %1.4%.codegen_version

//## begin module %1.4%.T
//## end module %1.4%.T

//## begin module %4D89B60D0280.cm preserve=no
// %X% %Q% %Z% %W%
//## end module %4D89B60D0280.cm

//## begin module %4D89B60D0280.cp preserve=no
//## end module %4D89B60D0280.cp

//## Module: Stream%4D89B60D0280; Pseudo Package specification
//## Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\Stream.h

#ifndef Stream_h
#define Stream_h

/// begin module %4D89B60D0280.additionalIncludes preserve=no
/// end module %4D89B60D0280.additionalIncludes

/// begin module %4D89B60D0280.includes preserve=yes
/// end module %4D89B60D0280.includes

// STREAM_TYPE
#include "STREAM_TYPE.h"

/// begin module %4D89B60D0280.additionalDeclarations preserve=yes
/// end module %4D89B60D0280.additionalDeclarations

/// begin Stream%4D89B60D0280.prelude preserve=yes
/// end Stream%4D89B60D0280.prelude

Daniel Guitart Pagola
class Stream
{
    // ## begin Stream%4D89B60D0280. initialDeclarations preserve=
    // ## end Stream%4D89B60D0280. initialDeclarations

public:
    // ## Constructors (specified)
    // ## Operation : Stream%4DC7CBED02CE
    Stream();

    // ## Operation : Stream%4D8C73750315
    // Class constructor
    Stream(double Tin, double Tout, double FCp, double h);

    // ## Destructor (generated)
    ~Stream();

    // ## Get and Set Operations for Class Attributes (generated)

    // ## Attribute : TIn%4D8C72E10382
    // Stores the starting temperature of a stream
    const double getTIn () const;
    void setTIn (double value);

    // ## Attribute : TOut%4D8C72EE032E
    // Stores the final temperature of a stream
    const double getTOut () const;
    void setTOut (double value);

    // ## Attribute : Fcp%4D8C730502E8
    // Stores the FCp of a stream (FCp stands for F*Cp
    // that
    // means massic flux * specific heat)
    const double getFCp () const;
    void setFCp (double value);

    // ## Attribute : h%4DD4FF3A02D8
    const double getH () const;
    void setH (double value);

    // ## Get and Set Operations for Associations (generated)
const STREAM_TYPE get_type () const;
void set_type (STREAM_TYPE value);

// Additional Public Declarations
## begin Stream %4 D89B60D0280 . public preserve=yes
## end Stream %4 D89B60D0280 . public

protected:
// Additional Protected Declarations
## begin Stream %4 D89B60D0280 . protected preserve=yes
## end Stream %4 D89B60D0280 . protected

private:
// Additional Private Declarations
## begin Stream %4 D89B60D0280 . private preserve=yes
## end Stream %4 D89B60D0280 . private

private: //## implementation
// Data Members for Class Attributes

## begin Stream :: TIn %4 D8C72E10382 . attr preserve=no
public: double {U}
double _tIn;
## end Stream :: TIn %4 D8C72E10382 . attr

## begin Stream :: TOut %4 D8C72EE032E . attr preserve=no
public: double {U}
double _tOut;
## end Stream :: TOut %4 D8C72EE032E . attr

## begin Stream :: Fcp %4 D8C730502E8 . attr preserve=no
public: double {U}
double _fcp;
## end Stream :: Fcp %4 D8C730502E8 . attr

## begin Stream :: h %4 DD4FF3A02D8 . attr preserve=no
public: double {U}
double _H;
## end Stream :: h %4 DD4FF3A02D8 . attr

// Data Members for Associations

## Association: <unnamed> %4 DA2EFF800EF
## begin Stream :: type %4 DA2EFF802A4 . role preserve=no
public: STREAM_TYPE { -> VHgN}
STREAM_TYPE type;
## end Stream :: type %4 DA2EFF802A4 . role

// Additional Implementation Declarations
//## begin Stream%4D89B60D0280.implementation preserve=yes
//## end Stream%4D89B60D0280.implementation

};

//## begin Stream%4D89B60D0280.postscript preserve=yes
//## end Stream%4D89B60D0280.postscript

// Class Stream

//## Get and Set Operations for Class Attributes (inline)

inline const double Stream::gettIn () const
{
    //## begin Stream::gettIn%4D8C72E10382.get preserve=no
    return _tIn;
    //## end Stream::gettIn%4D8C72E10382.get
}

inline void Stream::settIn (double value)
{
    //## begin Stream::settIn%4D8C72E10382.set preserve=no
    _tIn = value;
    //## end Stream::settIn%4D8C72E10382.set
}

inline const double Stream::gettOut () const
{
    //## begin Stream::gettOut%4D8C72EE032E.get preserve=no
    return _tOut;
    //## end Stream::gettOut%4D8C72EE032E.get
}

inline void Stream::settOut (double value)
{
    //## begin Stream::settOut%4D8C72EE032E.set preserve=no
    _tOut = value;
    //## end Stream::settOut%4D8C72EE032E.set
}

inline const double Stream::getfcp () const
{
    //## begin Stream::getfcp%4D8C730502E8.get preserve=no
    return _fcp;
    //## end Stream::getfcp%4D8C730502E8.get
}

inline void Stream::setfcp (double value)
{
    //## begin Stream::setfcp%4D8C730502E8.set preserve=no
    _fcp = value;
}
```cpp
// ## end Stream::setfcp%4D8C730502E8.set

inline const double Stream::getH () const
{
    // ## begin Stream::getH%4DD4FF3A02D8.get preserve=no
    return _H;
    // ## end Stream::getH%4DD4FF3A02D8.get
}

inline void Stream::setH (double value)
{
    // ## begin Stream::setH%4DD4FF3A02D8.set preserve=no
    _H = value;
    // ## end Stream::setH%4DD4FF3A02D8.set
}

// ## Get and Set Operations for Associations (inline)

inline const STREAM_TYPE Stream::get_type () const
{
    // ## begin Stream::get_type%4DA2EFF802A4.get preserve=no
    return type;
    // ## end Stream::get_type%4DA2EFF802A4.get
}

inline void Stream::set_type (STREAM_TYPE value)
{
    // ## begin Stream::set_type%4DA2EFF802A4.set preserve=no
    type = value;
    // ## end Stream::set_type%4DA2EFF802A4.set
}

// ## begin module%4D89B60D0280.epilog preserve=yes
// ## end module%4D89B60D0280.epilog

#endif

D.1.11 STREAM_TYPE.h

```
### begin module %4 DA2EFE80005 .cp preserve=no
### end module %4 DA2EFE80005 .cp

### Module: STREAM_TYPE %4 DA2EFE80005; Pseudo Package specification
### Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg\STREAM_TYPE.h

#ifndef STREAM_TYPE_h
#define STREAM_TYPE_h 1

### begin module %4 DA2EFE80005 .additionalIncludes preserve=no
### end module %4 DA2EFE80005 .additionalIncludes

### begin module %4 DA2EFE80005 .includes preserve=yes
### end module %4 DA2EFE80005 .includes

### begin module %4 DA2EFE80005 .additionalDeclarations preserve=yes
### end module %4 DA2EFE80005 .additionalDeclarations

### begin STREAM_TYPE %4 DA2EFE80005 .preface preserve=yes
### end STREAM_TYPE %4 DA2EFE80005 .preface

### Class: STREAM_TYPE %4 DA2EFE80005
### Stores if the stream is hot or cold.
### Category: <Top Level>
### Persistence: Transient
### Cardinality/Multiplicity: n

typedef enum{UNKNOWN=0,HOT,COLD} STREAM_TYPE;

### begin STREAM_TYPE %4 DA2EFE80005 .postscript preserve=yes
### end STREAM_TYPE %4 DA2EFE80005 .postscript

### begin module %4 DA2EFE80005 .epilog preserve=yes
### end module %4 DA2EFE80005 .epilog

#endif

D.1.12 Utility.h

### begin module %1.4%.codegen_version preserve=yes
### Read the documentation to learn more about C++ code generator

Daniel Guitart Pagola
## Module : Utility %4 DC7A71F00E7; Pseudo Package specification
### Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg\Utility.h

```cpp
#ifndef Utility_h
#define Utility_h 1

/// begin module%4DC7A71F00E7.additionalIncludes preserve=no
/// end module%4DC7A71F00E7.additionalIncludes

/// begin module%4DC7A71F00E7.includes preserve=yes
/// end module%4DC7A71F00E7.includes

// Stream
#include "Stream.h"

/// begin module%4DC7A71F00E7.additionalDeclarations preserve=yes
/// end module%4DC7A71F00E7.additionalDeclarations

/// begin Utility%4DC7A71F00E7.preface preserve=yes
/// end Utility%4DC7A71F00E7.preface

/// Class: Utility%4DC7A71F00E7
/// Category: <Top Level>
/// Persistence: Transient
/// Cardinality/Multiplicity: n

class Utility : public Stream  /// Inherits: <unnamed>%4 DC7A73700F8
{
    /// begin Utility%4DC7A71F00E7.initialDeclarations preserve=yes
    /// end Utility%4DC7A71F00E7.initialDeclarations

    public:
        /// Constructors (specified)
        /// Operation: Utility%4DC7CBA00EA
        Utility (double Tin, double Tout, double FCP);

    /// Destructor (generated)
```
-Utility();

// Additional Public Declarations
### begin Utility%4DC7A71F00E7.public preserve=yes
### end Utility%4DC7A71F00E7.public

protected:
// Additional Protected Declarations
### begin Utility%4DC7A71F00E7.protected preserve=yes
### end Utility%4DC7A71F00E7.protected

private:
// Additional Private Declarations
### begin Utility%4DC7A71F00E7.private preserve=yes
### end Utility%4DC7A71F00E7.private

private: //### implementation
// Additional Implementation Declarations
### begin Utility%4DC7A71F00E7.implementation preserve=yes
### end Utility%4DC7A71F00E7.implementation

};

### begin Utility%4DC7A71F00E7.postscript preserve=yes
### end Utility%4DC7A71F00E7.postscript

// Class Utility

### begin module%4DC7A71F00E7.epilog preserve=yes
### end module%4DC7A71F00E7.epilog

#endif

D.2 Sources

D.2.1 Curve.cpp

### begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
// versioning.
### end module%1.4%.codegen_version

### begin module%4DD11CC70100.cm preserve=no
// %X% %Q% %Z% %W%
### end module%4DD11CC70100.cm
```cpp
// Curve
#include "Curve.h"

Curve::Curve (std::string name)
// ## begin Curve::Curve %4DD2432400CD . hasinit preserve=no
// ## end Curve::Curve %4DD2432400CD . hasinit

// ## begin Curve::Curve %4DD2432400CD . initialization preserve=yes
// ## end Curve::Curve %4DD2432400CD . initialization
{
// ## begin Curve::Curve %4DD2432400CD . body preserve=yes
// ## end Curve::Curve %4DD2432400CD . body
}

Curve::~Curve ()
// ## begin Curve::~Curve %4DD11CC70100_dest . body preserve=yes
// ## end Curve::~Curve %4DD11CC70100_dest . body

// # Other Operations (implementation)
void Curve::Extrapolate (double H, double T)
// ## begin Curve::Extrapolate %4DD11D8903B4 . body preserve=yes
// ## end Curve::Extrapolate %4DD11D8903B4 . body
{
    for (int j=0; j<this->Points.size(); j++) {
        this->Points[j].seth(this->Points[j].geth() + H);
    }
}
```
```cpp
    this->Points[j].sett(this->Points[j].gett() + T);
}

    //## end Curve::Extrapolate%4DD11D8903B4.body
}

void Curve::PlotToFile (bool addToExisting, std::string filename)
{
    //## begin Curve::PlotToFile%4DD11DB0013D.body
    if(addToExisting==false){
        ofstream f;
        f.open(filename.c_str(),ios::out);
        f.setf(ios::fixed,ios::floatfield);
        f.precision(2);
        f << "#Enthalpy Temperature" << endl;
        for (int j=0;j<this->Points.size();j++) {
            f << this->Points[j].geth() << " " << this->Points[j].gett() << endl;
        }
        f.close();
    }
    else if (addToExisting==true){
        ofstream f;
        f.open(filename.c_str(),ios::app);
        f.setf(ios::fixed,ios::floatfield);
        f.precision(2);
        f << endl;
f << endl;
        for (int j=0;j<this->Points.size();j++) {
            f << this->Points[j].geth() << " " << this->Points[j].gett() << endl;
        }
        f.close();
    }
    //## end Curve::PlotToFile%4DD11DB0013D.body

    // Additional Declarations
}
```

Daniel Guitart Pagola
D.2.2 Interval.cpp

```cpp
//## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
//## end module%1.4%.codegen_version

/// Module: Interval; Pseudo Package body
/// Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg\Interval.cpp

//## begin module%4D8C721B002A.additionalIncludes preserve=no
//## end module%4D8C721B002A.additionalIncludes

#include "stdafx.h"

//## begin module%4D8C721B002A.additionDeclarations preserve=yes
//## end module%4D8C721B002A.additionDeclarations

// Class Interval
Interval::Interval (double Ts, double Te, double FCp)
```

Daniel Guitart Pagola
{ //## begin Interval::Interval%4D8C923800E2.body preserve=yes
this->_ts = Ts;
this->_te = Te;
this->_dT = Te-Ts;
//## end Interval::Interval%4D8C923800E2.body
}

Interval::~Interval()
{
//## begin Interval::~Interval%4D8C721B002A_dest.body
  preserve=yes
//## end Interval::~Interval%4D8C721B002A_dest.body
}

//## Other Operations (implementation)
double Interval::calculateHFlux ()
{
//## begin Interval::calculateHFlux%4D944B510260.body
  preserve=yes
    return (this->_dT*this->_fcp);
//## end Interval::calculateHFlux%4D944B510260.body

// Additional Declarations
//## begin Interval%4D8C721B002A.declarations preserve=yes
//## end Interval%4D8C721B002A.declarations

//## begin module%4D8C721B002A.epilog preserve=yes
//## end module%4D8C721B002A.epilog

D.2.3 IntervalH.cpp

//## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code
  generator
// versioning.
//## end module%1.4%.codegen_version

//## begin module%4DCA4DB700A8.cm preserve=no
  // %X% %Q% %Z% %W%
//## end module%4DCA4DB700A8.cm

//## begin module%4DCA4DB700A8.cp preserve=no
//## end module%4DCA4DB700A8.cp

//## Module: IntervalH%4DCA4DB700A8; Pseudo Package body

Daniel Guitart Pagola
// ## Source file: C:\Documents and Settings\user\Mes documents\ 
          DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\IntervalH.cpp

// ## begin module%4DCA4DB700A8.additionalIncludes preserve=no
// ## end module%4DCA4DB700A8.additionalIncludes

// ## begin module%4DCA4DB700A8.includes preserve=yes
#include "stdafx.h"
#include <math.h>
#include "problem.h"
// ## end module%4DCA4DB700A8.includes

// IntervalH
#include "IntervalH.h"
// ## begin module%4DCA4DB700A8.additionalDeclarations preserve=yes
// ## end module%4DCA4DB700A8.additionalDeclarations

// Class IntervalH

IntervalH::IntervalH (Problem* pb, double TColdIn, double TColdOut, double THotOut, double THotIn, double hLeft, double hRight)
    // ## begin IntervalH::IntervalH%4DCA559F0375.hasinit preserve=no
    // ## end IntervalH::IntervalH%4DCA559F0375.hasinit
    // ## begin IntervalH::IntervalH%4DCA559F0375.initialization preserve=yes
    // ## end IntervalH::IntervalH%4DCA559F0375.initialization
{
    // ## begin IntervalH::IntervalH%4DCA559F0375.body preserve=yes
    this->_tColdIn = TColdIn;
    this->_tColdOut = TColdOut;
    this->_tHotOut = THotOut;
    this->_tHotIn = THotIn;
    this->_HLeft = hLeft;
    this->_HRight = hRight;
    this->pb = pb;
    // ## end IntervalH::IntervalH%4DCA559F0375.body
}

IntervalH::~IntervalH()
{
    // ## begin IntervalH::~IntervalH%4DCA4DB700A8_dest.body preserve=yes
    // ## end IntervalH::~IntervalH%4DCA4DB700A8_dest.body
}
/// Other Operations (implementation)

void IntervalH::FindDTlm ()
{
    //## begin IntervalH::FindDTlm%4DCA573A024E.body preserve=yes
    this->_dTlm=((this->_tHotIn-this->_tColdOut)-(this->_tHotOut-this->_tColdIn))/(log((this->_tHotIn-this->_tColdOut)/(this->_tHotOut-this->_tColdIn)));
    //## end IntervalH::FindDTlm%4DCA573A024E.body
}

void IntervalH::FindSumQh ()
{
    //## begin IntervalH::FindSumQh%4DD5021303E1.body preserve=yes
    double SumHot = 0, SumCold = 0;
    int i;
    //double deltaT, FCP, Q, H;

    for (i=0; i<(pb->getnColdStreams())+pb->
    getnHotStreams()); i++){
        if (pb->Streams[i]->get_type()==COLD &&
        (this->_tColdOut<=pb->Streams[i]->
        getOut() && this->_tColdIn>=pb->
        Streams[i]->getIn())){
            SumCold=SumCold+((this->_tColdOut-this->_tColdIn)*pb
            ->Streams[i]->getfcp() /pb->
            Streams[i]->getH());
        }
        if (pb->Streams[i]->get_type()==HOT &&
        (this->_tHotIn<=pb->Streams[i]->
        getIn() && this->_tHotOut>=pb->
        Streams[i]->getOut())){
            SumHot=SumHot+((this->_tHotIn-
            this->_tHotOut)*pb->Streams[ i ]->getfcp() /pb->Streams[ i ]->getH());
        }
    }
    for (i=0; i<(pb->getnUtilitiesCold())+pb->
    getnUtilitiesHot()); i++){
        if (pb->Utilities[i]->get_type()==COLD &&
        (this->_tColdOut<=pb->Utilities[i ]->getOut() && this->_tColdIn>=pb->
        Utilities[i]->getIn())){
            SumCold=SumCold+((this->_tColdOut-this->_tColdIn)*pb
            ->Utilities[i]->_getfcp() /pb->Utilities[i]->_getH());
        }
    }
}

Daniel Guitart Pagola
->Streams[i]->getH());

if (pb->Utilities[i]->getType() == HOT
     && (this->_tHotIn <= pb->Utilities[i]->getIn() && this->_tHotOut >= pb->Utilities[i]->getOut())
     ){
    SumHot = SumHot + ((this->_tHotIn - this->_tHotOut)*pb->Utilities[i]->getfcp() / pb->Utilities[i]->getH());
}

this->_qhHotStreams = SumHot;
this->_qhColdStreams = SumCold;

// ## end IntervalH::FindSumQh %4 DD5021303E1.body
}

void IntervalH::FindArea ()
{
    // ## begin IntervalH::FindArea%4 DD4FFAC0056.body preserve=yes
    // ## end IntervalH::FindArea%4 DD4FFAC0056.body
}

// Additional Declarations
// ## begin module%4 DCA4DB700A8.declarations preserve=yes
// ## end module%4 DCA4DB700A8.declarations
// ## begin module%4 DCA4DB700A8.epilog preserve=yes
// ## end module%4 DCA4DB700A8.epilog

D.2.4 PinchDlg.cpp

// PinchDlg.cpp : Defines the class behaviors for the application.
//
#
#include "stdafx.h"
#include "PinchDlg.h"
#include "PinchDlgDlg.h"

#ifdef _DEBUG
# define new DEBUG_NEW
#endif
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
BEGIN_MESSAGE_MAP(CPinchDlgApp, CWinApp)
    // {{AFX_MSG_MAP(CPinchDlgApp)
    // NOTE - the ClassWizard will add and remove
    // mapping macros here.
    //   DO NOT EDIT what you see in these blocks
    // of generated code!
    // }}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

CPinchDlgApp::CPinchDlgApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

// The one and only CWinApp object
CPinchDlgApp theApp;

// CPinchDlgApp initialization
BOOL CPinchDlgApp::InitInstance()
{
    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to
    // reduce the size
    // of your final executable, you should remove from
    // the following
    // the specific initialization routines you do not
    // need.

    #ifdef _AFXDLL
    Enable3dControls(); // Call this
    when using MFC in a shared DLL
    #else
    Enable3dControlsStatic(); // Call this when
    linking to MFC statically
    #endif

    CPinchDlgDlg dlg;

m_pMainWnd = &dlg;
int nResponse = dlg.DoModal();
if (nResponse == IDOK)
{
    // TODO: Place code here to handle when the
dialog is
    // dismissed with OK
}
else if (nResponse == IDCANCEL)
{
    // TODO: Place code here to handle when the
dialog is
    // dismissed with Cancel
}

// Since the dialog has been closed, return FALSE so
that we exit the
// application, rather than start the application’s
message pump.
return FALSE;

D.2.5 PinchDlgDlg.cpp

// PinchDlgDlg.cpp : implementation file
//
#
#include "stdafx.h"
#include "PinchDlg.h"
#include "PinchDlgDlg.h"
#include <stdio.h>
#include <stdlib.h>
#include <fstream.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif
static char THIS_FILE[] = __FILE__;
#endif

// include "api_scilab.h" // SCILAB for Raphaëlle -> not using
// stack-c.h because according to scilab documentation, its
// old and will be discarded in next version... In fact if you
// try to compile including that library it shows an error with
// this exact message
#include "call_scilab.h" // SCILAB

#ifndef _DEBUG
#define new NEW_NEW
#endif

static char THIS_FILE[] = __FILE__;
#endif

// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

    // Dialog Data
    // {{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX }; // }}AFX_DATA

    // ClassWizard generated virtual function overrides
    // {{AFX_VIRTUAL(CAboutDlg)
    protected:
        virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
    // }}AFX_VIRTUAL

    // Implementation
    protected:
    // {{AFX_MSG(CAboutDlg)
    DECLARE_MESSAGE_MAP()
    // }}AFX_MSG

    CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
    { // {{AFX_DATA_INIT(CAboutDlg)
        // }}AFX_DATA_INIT
    }

    void CAboutDlg::DoDataExchange(CDataExchange* pDX)
    {
        CDialog::DoDataExchange(pDX);
        // {{AFX_DATA_MAP(CAboutDlg)
        // }}AFX_DATA_MAP
    }

    BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    // {{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
    // }}AFX_MSG_MAP
    END_MESSAGE_MAP()

    // ///////////////////////////////////////
    // CPinchDlgDlg dialog

    CPinchDlgDlg::CPinchDlgDlg(CWnd* pParent /*=NULL*/) : CDialog(CPinchDlgDlg::IDD, pParent)
    { // {{AFX_DATA_INIT(CPinchDlgDlg)
        m_DTmin = 10.0;
    }
m_TPinch = 0.0;
m_HotMER = 0.0;
m_ColdMER = 0.0;
m_targetArea = 0.0;
m_PointsType = _T(""");
m_coef_a = 40000.0;
m_coef_b = 500.0;
m_coef_c = 1.0;
m_targetCost = 0.0;
m_interest = 0.1;
m_years = 5.0;
m_annualCost = 0.0;
m_costutilitycold = 10000.0;
m_costutilityhot = 120000.0;
m_totalcost = 0.0;
m_utilitiescost = 0.0;
m_coefsType = -1;
m_utilitiesCheck = TRUE;
//}}AFX_DATA_INIT
// Note that LoadIcon does not require a subsequent
// DestroyIcon in Win32
m_hIcon = AfxGetApp() -> LoadIcon(IDR_MAINFRAME);
}

void CPinchDlgDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CPinchDlgDlg)
    DDX_Control(pDX, IDC_BUTTON_COSTFILEOPEN, m_buttonCostOpen);
    DDX_Control(pDX, IDC_BUTTON_COSTFILECLEAN, m_buttonCostClean);
    DDX_Control(pDX, IDC_BUTTON_COSTFILEADD, m_buttonCostAdd);
    DDX_Control(pDX, IDC_BUTTON_TCOST, m_buttonTcost);
    DDX_Control(pDX, IDC_BUTTON_POINTSGRAPH, m_buttonCurvePoints);
    DDX_Control(pDX, IDC_BUTTON_PLOTCC, m_buttonPlotCC);
    DDX_Control(pDX, IDC_BUTTON_PLOTBGCC, m_buttonPlotBGCC);
        DDX_Control(pDX, IDC_BUTTON_PLOTBCC, m_buttonPlotBCC);
    DDX_Control(pDX, IDC_BUTTON_PLOT, m_buttonPlotGCC);
    DDX_Control(pDX, IDC_BUTTON_CALCULATEPINCH, m_calculatePinchAndMER);
    DDX_Control(pDX, IDC_BUTTON_LOADDATA, m_loadData);
    DDX_Text(pDX, IDC_EDIT_DTMIN, m_DTmin);
    DDX_Text(pDX, IDC_EDIT1, m_TPinch);
    DDX_Text(pDX, IDC_EDIT2, m_HotMER);
    DDX_Text(pDX, IDC_EDIT3, m_ColdMER);
    DDX_Text(pDX, IDC_EDIT4, m_targetArea);
    DDX_CBString(pDX, IDC_COMBO_POINTSTYPELIST, m_PointsType);
DDX_Text(pDX, IDC_EDIT_COEFa, m_coef_a);
DDX_Text(pDX, IDC_EDIT_COEFb, m_coef_b);
DDX_Text(pDX, IDC_EDIT_COEFc, m_coef_c);
DDX_Text(pDX, IDC_EDIT_COST, m_targetCost);
DDX_Text(pDX, IDC_EDIT_INTEREST, m_interest);
DDX_Text(pDX, IDC_EDIT_YEARS, m_years);
DDX_Text(pDX, IDC_EDIT_ANNUALCOST, m_annualCost);
DDX_Text(pDX, IDC_EDIT_UT_COLD, m_costutilitycold);
DDX_Text(pDX, IDC_EDIT_UT_HOT, m_costutilityhot);
DDX_Text(pDX, IDC_EDIT_TOTALCOST, m_totalcost);
DDX_Text(pDX, IDC_EDIT_UTCOST, m_utilitiescost);
DDX_CBIndex(pDX, IDC_COMBO_COEFSELECT, m_coefsType);
DDX_Check(pDX, IDC_CHECK_UTILITIES, m_utilitiesCheck);
//}}AFX_DATA_MAP
BEGIN_MESSAGE_MAP(CPinchDlgDlg, CDialog)
//}}AFX_MSG_MAP(CPinchDlgDlg)
ON_WM_SYSCOMMAND()
ON_WM_PAINT()
ON_WM_QUERYDRAGICON()
ON_EN_CHANGE(IDC_EDIT1, OnChangeEditTPinch)
ON_BN_CLICKED(IDC_BUTTON_PLOT, OnButtonPlot)
ON_BN_CLICKED(IDC_BUTTON_PLOTCC, OnButtonPlotcc)
ON_BN_CLICKED(IDC_BUTTON_LOADDATA, OnButtonLoaddata)
ON_BN_CLICKED(BUTTON_EXIT, OnExit)
ON_BN_CLICKED(IDC_BUTTON_PLOTBCC, OnButtonPlotbcc)
ON_BN_CLICKED(IDC_BUTTON_HINTERVALS, OnButtonHintervals)
ON_BN_CLICKED(IDC_BUTTON_TCOST, OnButtonTcost)
ON_BN_CLICKED(IDC_BUTTON_STREAMS, OnButtonStreams)
ON_BN_CLICKED(IDC_BUTTON_UTILITIES, OnButtonUtilities)
ON_BN_CLICKED(IDC_BUTTON_POINTSGRAPH, OnButtonPointsgraph)
ON_BN_CLICKED(IDC_BUTTON_COSTFILEOPEN, OnButtonCostfileopen)
ON_BN_CLICKED(IDC_BUTTON_COSTFILEADD, OnButtonCostfileadd)
ON_BN_CLICKED(IDC_BUTTON_COSTFILECLEAN, OnButtonCostfileclean)
ON_CBN_SELCHANGE(IDC_COMBO_COEFSELECT, OnSelchangeComboCoefselect)
ON_BN_CLICKED(IDC_BUTTON_CHOOSEDIR, OnButtonChoosedir)
ON_BN_CLICKED(IDC_CHECK_UTILITIES, OnCheckUtilities)
ON_BN_CLICKED(IDC_BUTTON_CALCULATEPINCH, OnButtonCalculatepinch)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

Daniel Guitart Pagola
BOOL CPinchDlgDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.
    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING,
                IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog. The framework does
    // this automatically
    // when the application’s main window is not a dialog
    SetIcon(m_hIcon, TRUE);   // Set big icon
    SetIcon(m_hIcon, FALSE);  // Set small icon

    // TODO: Add extra initialization here

    PROBLEM = new Problem();
    PROBLEM->RetrievePaths();

    return TRUE; // return TRUE unless you set the focus
to a control
}

void CPinchDlgDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}
// If you add a minimize button to your dialog, you will need
// the code below
// to draw the icon. For MFC applications using the document/
// view model,
// this is automatically done for you by the framework.

void CPinchDlgDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for
        // painting

        SendMessage(WM_ICONERASEBKGND, (WPARAM) dc. GetSafeHdc(), 0);
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(& rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}

// The system calls this to obtain the cursor to display while
// the user drags
// the minimized window.
HCURSOR CPinchDlgDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CPinchDlgDlg::OnChangeEditTPinch()
{
    // TODO: If this is a RICHEDIT control, the control
    // will not
    // send this notification unless you override the
    // CDialog::OnInitDialog()
    // function and call CRichEditCtrl().SetEventMask()
    // with the ENM_CHANGE flag ORed into the mask.

// TODO: Add your control notification handler code here

void CPinchDlgDlg::OnButtonPlot()
{
    std::string tempFilename;
    tempFilename = "cd ";
    tempFilename.append(PROBLEM->getoctavePath());
    tempFilename.append(" & wgnuplot.exe ");
    tempFilename.append(PROBLEM->getprogramPath());
    tempFilename.append("gnuplotSettingsGCC.txt" - persist ");
    PROBLEM->Curves[0].PlotToFile (false, "gccPoints.dat");
    system(tempFilename.c_str());

    //system(" cd "C:\\Octave\\3.2.4_gcc-4.4.0\\bin" &
    wgnuplot.exe "C:\\Documents and Settings\\user\\Mes
documents\\DOCUMENTS\\STAGE D. GUITART\\
PinchSoftware\\PinchDlg2\\gnuplotSettingsGCC.txt" - persist");
    
    /*
    //EXPERIMENTAL CODE FOR SCILAB INTEGRATION
    
    #ifdef _MSC_VER
    if ( StartScilab(NULL,NULL,NULL) == FALSE )
    #else
    if ( StartScilab(getenv("SCI"),NULL,NULL) == FALSE )
    #endif
    {
        fprintf(stderr,"Error while calling StartScilab\n");
    }
    SendScilabJob("cd "C:\\Documents and Settings\\user\\
    Mes documents\\DOCUMENTS\\STAGE D. GUITART\\
PinchSoftware\\PinchDlg\\");
    SendScilabJob("exec("PlotScilabGCC.sci")");
    if ( TerminateScilab(NULL) == FALSE )
    {
        fprintf(stderr,"Error while calling TerminateScilab\n");
    }
    */
}
void CPinchDlgDlg::OnButtonPlotcc()
{
    std::string tempFilename;
    tempFilename = "cd ";
    tempFilename.append(PROBLEM->getoctavePath());
    tempFilename.append(" & wgnuplot.exe ");
    tempFilename.append(PROBLEM->getprogramPath());
    tempFilename.append(" gnuplotSettingsCC .txt\" -persist")
    ;
    PROBLEM->Curves[2].PlotToFile(false, "pointsCC.dat");
    PROBLEM->Curves[1].PlotToFile(true, "pointsCC.dat");
    system(tempFilename.c_str());
    // system("cd \"C:\Octave\3.2.4_gcc-4.4.0\bin\" & wgnuplot.exe \"C:\Documents and Settings\user\Mes
documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\gnuplotSettingsCC .txt\" - persist");
}

void CPinchDlgDlg::OnButtonLoaddata()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE) ;
    PROBLEM->setincludeUtilities(m_utilitiesCheck != 0);
    PROBLEM->setdtMin(m_DTmin) ;
    PROBLEM->defineProblem() ;
    m_calculatePinchAndMER.EnableWindow(TRUE);
    UpdateData(FALSE) ;
}

void CPinchDlgDlg::OnExit()
{
    // TODO: Add your control notification handler code here
    // PROBLEM->~Problem() ;
    AfxGetMainWnd()->PostMessage(WM_CLOSE);
}
void CPinchDlgDlg::OnButtonPlotbcc()
{
    // TODO: Add your control notification handler code here

    std::string tempFilename;
    tempFilename = "cd ";
    tempFilename.append(PROBLEM->getoctavePath());
    tempFilename.append(" & wgnuplot.exe ");
    tempFilename.append(PROBLEM->getprogramPath());
    tempFilename.append("gnuplotSettingsBCC.txt " - persist ");

    PROBLEM->Curves[4].PlotToFile (false, "pointsBCC.dat");
    PROBLEM->Curves[3].PlotToFile (true, "pointsBCC.dat");
    system(tempFilename.c_str());

    //system("cd "C:\Octave\3.2.4_gcc-4.4.0\bin" &
    wgnuplot.exe "C:\Documents and Settings\user\Mes
documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\gnuplotSettingsBCC.txt" - persist");
}

void CPinchDlgDlg::OnButtonHintervals()
{
    // TODO: Add your control notification handler code here

    PROBLEM->CalculateEnthalpyIntervals();
}

void CPinchDlgDlg::OnButtonTcost()
{
    // TODO: Add your control notification handler code here

    UpdateData(TRUE);

    //if (m_utilitiesCheck == 0) {
    //    PROBLEM->CalculateInteriorCurves();
    //}

    PROBLEM->CalculateEnthalpyIntervals();
    PROBLEM->CalculateTargetArea();
PROBLEM->CalcuateTargetCost(m_coef_a,m_coef_b,m_coef_c);

m_targetArea=PROBLEM->getTargetArea();
m_targetCost=PROBLEM->getTargetCost();

m_annualCost=m_targetCost*PROBLEM->CalculateInterest(m_interest,m_years);
m_utilitiescost=m_ColdMER*m_costutilitycold+m_HotMER*m_costutilityhot;
m_totalcost=m_annualCost+m_utilitiescost;
m_buttonCostAdd.EnableWindow(TRUE);
UpdateData(FALSE);
}

void CPinchDlgDlg::OnButtonStreams()
{
    // TODO: Add your control notification handler code here
    system("notepad.exe streams.txt");
}

void CPinchDlgDlg::OnButtonUtilities()
{
    // TODO: Add your control notification handler code here
    system("notepad.exe utilities.txt");
}

void CPinchDlgDlg::OnButtonPointsgraph()
{
    // TODO: Add your control notification handler code here

    UpdateData(TRUE);

    if (m_PointsType == "GCC"){
        PROBLEM->Curves[0].PlotToFile(false, "gccPoints.dat");
        system("notepad.exe gccPoints.dat");
    }
    if (m_PointsType == "CC"){
        PROBLEM->Curves[2].PlotToFile(false, "pointsCC.dat");
        PROBLEM->Curves[1].PlotToFile(true, "pointsCC.dat");
        system("notepad.exe PointsCC.dat");
    }
    if (m_PointsType == "BCC"){
        PROBLEM->Curves[4].PlotToFile(false, "pointsBCC.dat");
    }
PROBLEM->Curves[3].PlotToFile (true, "pointsBCC.dat");
  system("notepad.exe PointsBCC.dat");
}

void CPinchDlgDlg::OnButtonCostfileopen()
{
  // TODO: Add your control notification handler code here
  system("notepad.exe costData.txt");
}

void CPinchDlgDlg::OnButtonCostfileadd()
{
  // TODO: Add your control notification handler code here
  ofstream f;
  f.open("costData.txt",ios::app);
  f <<m_DTmin<<"\t"<<m_HotMER<<"\t"<<m_HotMER*
      m_costutilityhot<<"\t"<<m_ColdMER<<"\t"<<m_ColdMER*
      m_costutilitycold<<"\t"<<m_targetArea<<"\t"<<PROBLEM
      ->getnExchangers()<<"\t"<<m_annualCost<<"\t"<<
      m_totalcost<<endl;
  f.close();
}

void CPinchDlgDlg::OnButtonCostfileclean()
{
  // TODO: Add your control notification handler code here
  ofstream f;
  f.open("costData.txt",ios::out);
  f <<"# DTmin\tHotMER\tHotUtility\tColdMER\tColdUtility\n      \tNetwork\tNunits\tAnnualCap\tAnnual" << endl;
  f <<"#\t\tCost\t\tCost\t\tArea\t\tNunits\t\tTotalCost\t\tTotalCost" << endl;
  //f <<"#DTmin HotMER HotUtilityCost ColdMER ColdUtilityCost NetworkArea Nunits
  // AnnualCapitalCost AnnualTotalCost" << endl;
  f.close();
}

void CPinchDlgDlg::OnSelchangeComboCoefselect()
{
  // TODO: Add your control notification handler code here

// Data extracted from S. G. Hall, S. Ahmad and R. Smith "Capital cost target for heat exchanger..." article

UpdateData(TRUE);

switch ( m_coefsType )
{
    case 1:
        m_coef_a = 30800;
        m_coef_b = 750;
        m_coef_c = 0.81;
        break;
    case 2:
        m_coef_a = 30800;
        m_coef_b = 1644;
        m_coef_c = 0.81;
        break;
    case 3:
        m_coef_a = 30800;
        m_coef_b = 1339;
        m_coef_c = 0.81;
        break;
    case 4:
        m_coef_a = 30800;
        m_coef_b = 4407;
        m_coef_c = 0.81;
        break;
    case 5:
        m_coef_a = 30800;
        m_coef_b = 3349;
        m_coef_c = 0.81;
        break;
    case 6:
        m_coef_a = 30800;
        m_coef_b = 3749;
        m_coef_c = 0.81;
        break;
    case 9:
        m_coef_a = 30800;
        m_coef_b = 750;
        m_coef_c = 0.81;
        break;
    case 10:
        m_coef_a = 30800;
        m_coef_b = 890;
        m_coef_c = 0.81;
        break;
    case 11:
        m_coef_a = 30800;
        m_coef_b = 1089;
        m_coef_c = 0.81;
}
break;
case 12:
m_coef_a = 30800;
m_coef_b = 983;
m_coef_c = 0.81;
break;
case 13:
m_coef_a = 30800;
m_coef_b = 1438;
m_coef_c = 0.81;
break;
case 14:
m_coef_a = 30800;
m_coef_b = 1201;
m_coef_c = 0.81;
break;
default:
break;
}
UpdateData(FALSE);
}
void CPinchDlgDlg::OnButtonChoosedir()
{
    // TODO: Add your control notification handler code here
    CFileDialog fileDlg(TRUE, NULL, NULL,
    OFN_ALLOWMULTISELECT | OFN_HIDEREADONLY, "All Files (*.*)|*.*||", this);
    fileDlg.m_ofn.lpstrTitle = "Select work directory:"
    if (fileDlg.DoModal() == IDOK)
    {
        CString szlstfile = fileDlg.GetPathName(); // This is your selected file name with path
        //AfxMessageBox("Your file name is:" + szlstfile);
    }
}
void CPinchDlgDlg::OnCheckUtilities()
{
    // TODO: Add your control notification handler code here
}
void CPinchDlgDlg::OnButtonCalculatepinch()
{
// TODO: Add your control notification handler code here
UpdateData(TRUE);
PROBLEM->setdtMin(m_DTmin);
PROBLEM->calculateMER();
PROBLEM->calculateCCpoints();
m_TPinch=PROBLEM->gettPinch();
m_HotMER=PROBLEM->gethotMER();
m_ColdMER=PROBLEM->getcoldMER();
m_buttonCurvePoints.EnableWindow(TRUE);
m_buttonPlotGCC.EnableWindow(TRUE);
m_buttonPlotCC.EnableWindow(TRUE);
m_buttonTcost.EnableWindow(TRUE);

if(m_utilitiesCheck){
    m_buttonPlotBGCC.EnableWindow(TRUE);
    m_buttonPlotBCC.EnableWindow(TRUE);
} else if (!m_utilitiesCheck) {
    m_buttonPlotBGCC.EnableWindow(FALSE);
    m_buttonPlotBCC.EnableWindow(FALSE);
}

UpdateData(FALSE);

D.2.6 Point.cpp

//## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
// versioning.
//## end module%1.4%.codegen_version

//## begin module%4DD11CD00173.cm preserve=no
// %X% %Q% %Z% %W%
//## end module%4DD11CD00173.cm

//## begin module%4DD11CD00173.cp preserve=no
//## end module%4DD11CD00173.cp

//## Module: Point%4DD11CD00173; Pseudo Package body
//## Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\Point.cpp

Daniel Guitart Pagola
```cpp
//## begin module%4DD11CD00173 .additionalIncludes preserve=no
//## end module%4DD11CD00173 .additionalIncludes

//## begin module%4DD11CD00173 .includes preserve=yes
#include "stdafx.h"
//## end module%4DD11CD00173 .includes

// Point
#include "Point.h"

//## begin module%4DD11CD00173 .additionalDeclarations preserve=yes
//## end module%4DD11CD00173 .additionalDeclarations

// Class Point

Point::Point (double H, double T)
//## begin Point::Point %4DD2764202DC . hasinit preserve=no
//## end Point::Point %4DD2764202DC . hasinit

//## begin Point::Point %4DD2764202DC . initialization preserve=yes
//## end Point::Point %4DD2764202DC . initialization
{
//## begin Point::Point %4DD2764202DC . body preserve=yes
    this->_h=H;
    this->_t=T;
//## end Point::Point %4DD2764202DC . body
}

Point::~Point ()
{
//## begin Point::~Point %4DD11CD00173_dest . body preserve=yes
//## end Point::~Point %4DD11CD00173_dest . body
}

// Additional Declarations
//## begin Point%4DD11CD00173 .declarations preserve=yes
//## end Point%4DD11CD00173 .declarations

//## begin module%4DD11CD00173 .epilog preserve=yes
//## end module%4DD11CD00173 .epilog
```

D.2.7 Problem.cpp

```cpp
//## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code
generator
// versioning.
```
Problem::Problem()

/// begin Problem::Problem%4D89B5CD0251_const.hasinit preserve=no
/// _octavePath("C:\Octave\3.2.4_gcc-4.4.0\bin")
/// end Problem::Problem%4D89B5CD0251_const.hasinit
/// begin Problem::Problem%4D89B5CD0251_const.initialization preserve=yes
/// end Problem::Problem%4D89B5CD0251_const.initialization
{
/// begin Problem::Problem%4D89B5CD0251_const.body preserve=yes

    this->_problemDefined=false;

/// end Problem::Problem%4D89B5CD0251_const.body
Problem::~Problem()
{
    //## begin Problem::~Problem%4D89B5CD0251_dest.body preserve=yes
    //## end Problem::~Problem%4D89B5CD0251_dest.body
}

//## Other Operations (implementation)
void Problem::readData ()
{
    //## begin Problem::readData%4D907A410344.body preserve=yes
    // Variables
    // ------------------------
    int    i=0, j=0;
    int    NH = 0, NC = 0, NT=0, NU=0, NCU=0, NHU=0;
    double Tin,Tout,FCp, h;

    //Open file
    FILE * streamfile;
    streamfile = fopen ("streams.txt","r");
    FILE * streamfile2;
    streamfile2 = fopen ("utilities.txt","r");

    //Loop to search the total number of currents
    fscanf(streamfile,"%*[\n]");
    while( !feof(streamfile) ){
        fscanf(streamfile,"%lf %lf %lf %lf\n", &Tin, &Tout, &FCp, &h);
        i++;
    }
    fscanf(streamfile2,"%*[\n]");
    while( !feof(streamfile2) ){
        fscanf(streamfile2,"%lf %lf %lf %lf\n", &Tin, &Tout, &FCp, &h);
        j++;
    }

    rewind(streamfile);
    rewind(streamfile2);
NT=i;
NU=j;

Streams = new Stream* [NT];
Utilities = new Stream* [NU];

//Loop to call the stream class constructor and store the data
fscanf(streamfile,"%*[\n]");
fscanf(streamfile2,"%*[\n]");
j=0;
i=0;

while (i<NT)
{
    fscanf(streamfile,"%lf %lf %lf %lf\n", &Tin, &Tout, &FCp, &h);

    Streams[i] = new Stream(Tin,Tout,FCp,h);

    if (Tin<Tout)
    {
        Streams[i]->set_type(COLD);
        NC++;
    } else if (Tin>Tout)
    {
        Streams[i]->set_type(HOT);
        NH++;
    }
    //fprintf(debugTXT,"%c %lf %lf %lf %d\n", ID, Tin, Tout, FCp, Streams[i]->get_type());
    i++;
    //fprintf(debugTXT,"%d\n", i);
}

////////////////

while (j<NU)
{
    fscanf(streamfile2,"%lf %lf %lf %lf\n", &Tin, &Tout, &FCp, &h);

    Utilities[j] = new Stream(Tin,Tout,FCp,h);

    if (Tin<Tout)
    {
        Utilities[j]->set_type(COLD);
    } else if (Tin>Tout)
    {
        Utilities[j]->set_type(HOT);
        NH++;
    }
    //fprintf(debugTXT,"%d\n", i);
}
else if (Tin > Tout) {
    Utilities[j]->set_type(HOT);
    NHU++;
}
// fprintf(debugTXT,"%c %lf %lf %lf %d\n ", ID, Tin, Tout, FCp, Utilities[j]->get_type());
j++;
// fprintf(debugTXT,"%d\n", j);
}

setnHotStreams(NH);
setnColdStreams(NC);
setnUtilitiesCold(NCU);
setnUtilitiesHot(NHU);
fclose(streamfile);
fclose(streamfile2);

return 0;
}

void Problem::calculateProblemArray ()
{
    //## begin Problem::calculateProblemArray%4D907ABC002.body
    int i, j = 0;
    std::vector<double> Temperatures((getnColdStreams() + 
            getnHotStreams())*2);
    std::vector<double>::iterator it;

    // Temporary use of a vector to store the information, 
    // due to ease of manipulation (standard library 
    // functions can be used)
    // The final info is stored at Interval class
    for (i = 0; i < (getnColdStreams() + getnHotStreams()); i++) {
        if (Streams[i]->get_type() == COLD) {
            Temperatures[j] = (Streams[i]->getIn() + 
               (_dtMin/2));
            Temperatures[j+1] = (Streams[i]->getOut () +(_dtMin/2));
        } else if (Streams[i]->get_type() == HOT) {
            Temperatures[j] = (Streams[i]->getIn() + 
               (_dtMin/2));
            Temperatures[j+1] = (Streams[i]->getOut () +(_dtMin/2));
        }
    }
    //## end Problem::calculateProblemArray%4D907ABC002.body
}
Temperatures[j] = (Streams[i]->getIn() - (_dtMin/2));
Temperatures[j+1] = (Streams[i]->getOut() - (_dtMin/2));

j = j + 2;

// Streams[i]->getIn()<Streams[i]->getOut()
// Streams[i]->getIn()<=Streams[i]->getOut()

std::sort(Temperatures.begin(), Temperatures.end());
it = std::unique(Temperatures.begin(), Temperatures.end());
Temperatures.resize(it - Temperatures.begin());

// call of the Interval constructor

_nIntervals = Temperatures.size() - 1;
Intervals = new Interval*[_nIntervals];

for (j = 0; j < _nIntervals; j++) {
    Intervals[j] = new Interval(Temperatures[j],
    Temperatures[j+1], 0);
    // cout << "Interval: " << Intervals[j]->getcom() << " " << Intervals[j]->getfin() << " " << Intervals[j]->getdT() << endl;
}

void Problem::defineProblem ()
{
    //## begin Problem::defineProblem%4D89B5ED03D8.body preserve=yes
    readData();
    _problemDefined = true;
    //## end Problem::defineProblem%4D89B5ED03D8.body

    void Problem::calculateFCpSum ()
    {
        //## begin Problem::calculateFCpSum%4D9047EE0045.body preserve=yes
        int i, j;
    }
for(j=0;j<_nHotStreams+_nColdStreams);j++){ //If the interval is inside the stream boundary temperatures, add the FCp of the stream to the interval //this happens if Ts INTERVAL >= Tin STREAM and Te INTERVAL < Tout STREAM (case of a cold stream)
if (Streams[j]->get_type()==COLD){
if (Intervals[i]->getts() >= Streams[j]->getIn() + (_dtMin/2) && Intervals[i]->gette() <= Streams[j]->getOut() + (_dtMin/2)){
    Intervals[i]->setfcp( Intervals[i]->getfcp() - Streams[j]->getfcp());
}
} else if (Streams[j]->get_type()==HOT) {
if (Intervals[i]->getts() >= Streams[j]->getOut() - (_dtMin/2) && Intervals[i]->gette() <= Streams[j]->getIn() + (_dtMin/2)){
    Intervals[i]->setfcp( Intervals[i]->getfcp() + Streams[j]->getfcp());
}
}

} //## end Problem::calculateFCpSum%4D9047EE0045.body
}

void Problem::calculateTPinch ()
{
    //## begin Problem::calculateTPinch%4D904824003F.body
    preserve=yes
    double R=0;
    double minR= 0;
for (int j = _nIntervals - 1; j >= 0; j --)
{
    R = R + Intervals[j]->calculateHFlux();
    if (R < minR)
    {
        minR = R;
        _tPinch = Intervals[j]->getts();
    }
}

_hotMER = -minR;
cout << "TPinch: " << _tPinch << endl;

/// end Problem::calculateTPinch %4D904824003F.body

void Problem::calculateCascade()
{
    /// begin Problem::calculateCascade %4D90483500DD.body
    preserve = yes

    FILE * gccPoints;
gccPoints = fopen("gccPoints.dat","w");

    if(Curves.empty() == false)
    {
        Curves.clear();
    }

    Curves.push_back(Curve("GCC"));

double R = _hotMER;
double minR = 0;
int i = 0;

    //fprintf(gccPoints,"# H T \n"); // Text file title
    //fprintf(gccPoints,"%lf %lf\n", _hotMER, Intervals[_nIntervals-1]->gette()); // Text file point data
    Curves[0].Points.push_back(Point(_hotMER, Intervals[_nIntervals-1]->gette()));
    //fprintf(gccPoints,"%lf %lf\n", Curves[0].Points[i].geth(), Curves[0].Points[i].gett());
    i = 1;
    for (int j = _nIntervals-1; j >= 0; j --)
    {
        R = R + Intervals[j]->calculateHFlux();
        //fprintf(gccPoints,"%lf %lf\n", R, Intervals[j]->getts()); // Text file point data
Curves[0].Points.push_back(Point(R, Intervals[j]->getts()));
// fprintf(gccPoints,"%lf %lf\n", Curves[0].Points[i].geth(), Curves[0].Points[i].gett());
i++;
}
_coldMER=R;
fclose(gccPoints);
cout << "Hot MER: " << _hotMER << endl;
cout << "Cold MER: " << _coldMER << endl;

//## end Problem::calculateCascade%4D90483500DD.body
}

void Problem::calculateMER()
{
  //## begin Problem::calculateMER%4D90487401BD.body preserve=yes

  if (_problemDefined==false){
    cout << "Problem not defined!" << endl;
  }

  calculateProblemArray();
calculateFCpSum();
calculateTPinch();
calculateCascade();
FindNumExchangers();

  //## end Problem::calculateMER%4D90487401BD.body
}

void Problem::calculateCCpoints()
{
  //## begin Problem::calculateCCpoints%4DA43CE7004E.body preserve=yes

  // variable declarations
  int i, j=0, k=0;
  int NIntervalsH, NIntervalsC;
  double R=0;
  std::string filename;
  std::vector<double> ColdTemps (getnColdStreams() * 2);
  std::vector<double> HotTemps (getnHotStreams() * 2);
  std::vector<double>::iterator it;
356 //if(_includeUtilities==true){
357     int     NIntervalsHU, NIntervalsCU;
358     std::vector<double> ColdTempsU (getnColdStreams ()*2);
359     std::vector<double> HotTempsU (getnHotStreams ()*2);
360 //}
361
362 //four vectors are created, all the temperatures
363 //corresponding to cold and hot streams are saved into
364 //them
365 for (i=0;i<(_nHotStreams+_nColdStreams);i++){
366     if (Streams[i]->get_type()==COLD){
367         ColdTemps[j]=(Streams[i]->getIn());
368         ColdTemps[j+1]=(Streams[i]->getOut());
369         j=j+2;
370     } else if (Streams[i]->get_type()==HOT) {
371         HotTemps[k]=(Streams[i]->getIn());
372         HotTemps[k+1]=(Streams[i]->getOut());
373         k=k+2;
374     }
375 }
376
377 if(_includeUtilities==true){
378     ColdTempsU=ColdTemps;
379     HotTempsU=HotTemps;
380     ColdTempsU.resize((getnColdStreams()+
381                          getnUtilitiesCold())*2);
382     HotTempsU.resize((getnHotStreams()+
383                          getnUtilitiesHot())*2);
384  
385  //Repeat for the utilities
386  for (i=0;i<(_nUtilitiesCold+_nUtilitiesHot);i++){
387     if (Utilities[i]->get_type()==COLD){
388         ColdTempsU[j]=(Utilities[i]->
389                          getIn());
390         ColdTempsU[j+1]=(Utilities[i]->
391                          getOut());
392         j=j+2;
393     } else if (Utilities[i]->get_type()==HOT) {
394         HotTempsU[k]=(Utilities[i]->
395                          getIn());
396     }
397 }
HotTempsU[k+1]=(Utilities[i]->
    gettOut());
    
k=k+2;

// vectors are sorted to create intervals

std::sort (ColdTemps.begin(), ColdTemps.end());
std::sort (HotTemps.begin(), HotTemps.end());
it = std::unique (ColdTemps.begin(), ColdTemps.end());
ColdTemps.resize( it - ColdTemps.begin() );
it = std::unique (HotTemps.begin(), HotTemps.end());
HotTemps.resize( it - HotTemps.begin() );

if(_includeUtilities==true){
    std::sort (ColdTempsU.begin(), ColdTempsU.end());
    std::sort (HotTempsU.begin(), HotTempsU.end());
it = std::unique (ColdTempsU.begin(),
    ColdTempsU.end());
    ColdTempsU.resize( it - ColdTempsU.begin() );
it = std::unique (HotTempsU.begin(), HotTempsU.
    end());
    HotTempsU.resize( it - HotTempsU.begin() );
}

// Interval constructor is called using the temperatures
// in the sorted vector, for cold and hot streams

NIntervalsH = HotTemps.size()-1; //!!do not mistake the
    // name with enthalpy, it stands for number of HOT
intervals
NIntervalsC = ColdTemps.size()-1;

if(_includeUtilities==true){
    NIntervalsHU = HotTempsU.size()-1;
    NIntervalsCU = ColdTempsU.size()-1;
}

Interval** HotIntervals;
Interval** ColdIntervals;
HotIntervals = new Interval* [NIntervalsH];
ColdIntervals = new Interval* [NIntervalsC];

Interval** HotIntervalsU;
Interval** ColdIntervalsU;
if(_includeUtilities==true){
    HotIntervalsU = new Interval*[NIntervalsHU];
    ColdIntervalsU = new Interval*[NIntervalsCU];
}

for (j=0;j<NIntervalsC;j++) {
    ColdIntervals[j] = new Interval(ColdTemps[j],
            ColdTemps[j+1],0);
    // cout << "IntervalF: " << ColdIntervals[j]->
        gettcom() << " " << ColdIntervals[j]->
        gettfin() << " " << ColdIntervals[j]->getdT() << endl;
}

for (j=0;j<NIntervalsH;j++) {
    HotIntervals[j] = new Interval(HotTemps[j],
            HotTemps[j+1],0);
    // cout << "IntervalC: " << HotIntervals[j]->
        gettcom() << " " << HotIntervals[j]->gettfin()
        () << " " << HotIntervals[j]->getdT() << endl;
}

if(_includeUtilities==true){
    for (j=0;j<NIntervalsCU;j++) {
        ColdIntervalsU[j] = new Interval(
            ColdTempsU[j],ColdTempsU[j+1],0);
        // cout << "IntervalF: " <<
            ColdIntervals[j]->gettcom() << " " <<
            ColdIntervals[j]->gettfin() << " " <<
            ColdIntervals[j]->getdT() << endl;
    }

    for (j=0;j<NIntervalsHU;j++) {
        HotIntervalsU[j] = new Interval(
            HotTempsU[j],HotTempsU[j+1],0);
        // cout << "IntervalC: " << HotIntervals
            [j]->gettcom() << " " <<
            HotIntervals[j]->gettfin() << " " <<
            HotIntervals[j]->getdT() << endl;
    }
}
// the FCp is calculated and saved in the intervals according to the streams that cross them

for (j = 0; j < (_nHotStreams + _nColdStreams); j++) {
    for (i = 0; i < NIntervalsC; i++) {
        if (Streams[j]->getType() == COLD) {
            if (ColdIntervals[i]->getTs() >= Streams[j]->getTIn() &&
                ColdIntervals[i]->getTe() <= Streams[j]->getTOut()) {
                ColdIntervals[i]->setFCp(ColdIntervals[i]->getFCp() -
                                         Streams[j]->getFCp());
            }
        } else if (Streams[j]->getType() == HOT) {
            
        }
    }

    for (i = 0; i < NIntervalsH; i++) {
        if (Streams[j]->getType() == COLD) {
            
        } else if (Streams[j]->getType() == HOT) {
            
            if (HotIntervals[i]->getTs() >= Streams[j]->getTOut() &&
                HotIntervals[i]->getTe() <= Streams[j]->getTIn()) {
                HotIntervals[i]->setFCp(HotIntervals[i]->getFCp() + Streams[j]->getFCp());
            }
        }
    }
}

if (_includeUtilities == true) {
//the F Cp is calculated and saved in the
intervals according to the streams that cross
them for balanced streams

for (j=0;j<(_nHotStreams+_nColdStreams);j++){
  for (i=0;i<NIntervalsCU;i++){
    if (Streams[j]->get_type() == COLD){
      if (ColdIntervalsU[i]->getts() >= Streams[j]->gettIn() &&
          ColdIntervalsU[i]->gette() <= Streams[j]->gettOut()){
        ColdIntervalsU[i]->setfcp(ColdIntervalsU[i]->getfcp()
          -Streams[j]->getfcp());
      }
    } else if (Streams[j]->get_type() ==HOT) {
    }
  }
  for (i=0;i<NIntervalsHU;i++){
    if (Streams[j]->get_type() == COLD){
    } else if (Streams[j]->get_type() ==HOT) {
      if (HotIntervalsU[i]->getts() >= Streams[j]->gettOut() &&
          HotIntervalsU[i]->gette() <= Streams[j]->gettIn()){
        HotIntervalsU[i]->setfcp(HotIntervalsU[i]->getfcp()
          +Streams[j]->getfcp());
      }
    }
  }
}

Daniel Guitart Pagola
for (j=0; j<(_nUtilitiesCold+_nUtilitiesHot); j++) {
    
    for (i=0; i<NIntervalsCU; i++) {
        if (Utilities[j]->get_type() == COLD) {
            if (ColdIntervalsU[i]->getts() >= Utilities[j]->getIn() &&
                ColdIntervalsU[i]->gette() <= Utilities[j]->getOut()) {
                ColdIntervalsU[i]->setfcp(ColdIntervalsU[i]->getfcp() - Utilities[j]->getfcp());
            }
        } else if (Utilities[j]->get_type() == HOT) {
        }
    }
    
    for (i=0; i<NIntervalsHU; i++) {
        if (Utilities[j]->get_type() == COLD) {
        } else if (Utilities[j]->get_type() == HOT) {
            if (HotIntervalsU[i]->getts() >= Utilities[j]->getOut() &&
                HotIntervalsU[i]->gette() <= Utilities[j]->getIn()) {
                HotIntervalsU[i]->setfcp(HotIntervalsU[i]->getfcp());
            }
        }
    }
}

Daniel Guitart Pagola
½ new curves are created in Curves array

Curves.push_back(Curve("ColdCC"));
Curves.push_back(Curve("HotCC"));
Curves.push_back(Curve("ColdBCC"));
Curves.push_back(Curve("HotBCC"));

//ColdCC
//fprintf(pointsCC,"%lf %lf\n", _coldMER, ColdIntervals [0]->getts());
R=_coldMER;
Curves[1].Points.push_back(Point(R, ColdIntervals[0]->getts()));
for (j=0; j<NIntervalsC-1; j++)
{
   R=R-ColdIntervals[j]->calculateHFlux();
   //fprintf(pointsCC,"%lf %lf\n", R,
   ColdIntervals[j]->gette());
   Curves[1].Points.push_back(Point(R,
   ColdIntervals[j]->gette()));
}
//Hot CC
R=0;
//fprintf(pointsCC,"%lf %lf\n", R, HotIntervals[0]->getts());
Curves[2].Points.push_back(Point(R, HotIntervals[0]->getts()));
for (j=0; j<NIntervalsH-1; j++)
{
   R=R+HotIntervals[j]->calculateHFlux();
   //fprintf(pointsCC,"%lf %lf\n", R, HotIntervals
   [j]->gette());
   Curves[2].Points.push_back(Point(R, HotIntervals
   [j]->gette()));
}

if (_includeUtilities==true){
   //Hot BCC
R=0;
// fprintf(hotPoints,"%lf %lf\n", R, HotIntervals[0]->
   getts());
Curves[4].Points.push_back(Point(R,HotIntervalsU[0]->
   getts()));
for (j=0;j<=NIntervalsHU-1;j++)
{
   R=R+HotIntervalsU[j]->calculateHFlux();
   // fprintf(hotPoints,"%lf %lf\n", R,
      HotIntervals[j]->gette());
   Curves[4].Points.push_back(Point(R,
      HotIntervalsU[j]->gette()));
}

// ColdBCC
// fprintf(coldPoints,"%lf %lf\n", R, ColdIntervals[0]->
   getts());
R=0; // _coldMER + ColdIntervalsU[0]->calculateHFlux();
   // !!!!
Curves[3].Points.push_back(Point(R,ColdIntervalsU[0]->
   getts()));
for (j=0;j<=NIntervalsCU-1;j++)
{
   R=R-ColdIntervalsU[j]->calculateHFlux();
   // fprintf(coldPoints,"%lf %lf\n", R,
      ColdIntervals[j]->gette());
   Curves[3].Points.push_back(Point(R,
      ColdIntervalsU[j]->gette()));
}

if (!_includeUtilities==false){
   CalculateInteriorCurves();
}

// ---------------------------------
// ## end Problem :: calculateCCpoints %4DA43CE7004E . body

void Problem::CalculateEnthalpyIntervals ()
{
   //## begin Problem :: CalculateEnthalpyIntervals %4DCA79A00127.
      body preserve=yes

double ThotLeft, ThotRight, TcoldLeft, TcoldRight,
   Ttemp;
double HhotLeft, HhotRight, HcoldLeft, HcoldRight,
   Htemp;
int i = 0, j=0, k=0, test;
bool temporalCold=false, temporalHot=false;
IntervalsH.clear();
test=IntervalsH.size();

HcoldLeft=Curves[3].Points[j].geth();
TcoldLeft=Curves[3].Points[j].gett();
HhotLeft=Curves[4].Points[k].geth();
ThotLeft=Curves[4].Points[k].gett();
j++;
k++;

while( j<=Curves[3].Points.size() && k<=Curves[4].Points.size() ){
    if(temporalHot==false){
        HhotRight=Curves[4].Points[k].geth();
        ThotRight=Curves[4].Points[k].gett();
        k++;
    }
    if(temporalCold==false){
        HcoldRight=Curves[3].Points[j].geth();
        TcoldRight=Curves[3].Points[j].gett();
        j++;
    }

    if(HhotRight==HcoldRight) { // the end of the interval has two points of the same enthalpy
        IntervalsH.push_back(IntervalH(this,
            TcoldLeft,TcoldRight,ThotLeft,
            ThotRight,HcoldLeft,HcoldRight));

        TcoldLeft=TcoldRight;
        HcoldLeft=HcoldRight;
        ThotLeft=ThotRight;
        HhotLeft=HhotRight;

        temporalCold=false;
        temporalHot=false;
    } else if (HcoldRight>HhotRight) { // the point on the hot curve creates the interval (has lower enthalpy)
        Htemp=HhotRight;
        Ttemp=((TcoldRight-TcoldLeft)*(Htemp-
            HcoldLeft)/(HcoldRight-HcoldLeft))+
            TcoldLeft;
        IntervalsH.push_back(IntervalH(this,
            TcoldLeft,Ttemp,ThotLeft,ThotRight,
            HcoldLeft,HhotRight));

        TcoldLeft=Ttemp;
        HcoldLeft=Htemp;
ThotLeft = ThotRight;
HhotLeft = HhotRight;
temporalCold = true;
temporalHot = false;
}

} else if (HhotRight > HcoldRight) { // the point on the cold curve creates the interval (has lower enthalpy)
    Htemp = HcoldRight;
    Ttemp = ((ThotRight - ThotLeft) * (Htemp - HhotLeft)) / (HhotRight - HhotLeft) + ThotLeft;
    IntervalsH.push_back(IntervalH(this, TcoldLeft, TcoldRight, ThotLeft, Ttemp, HcoldLeft, HcoldRight));

    TcoldLeft = TcoldRight;
    HcoldLeft = HcoldRight;
    ThotLeft = Ttemp;
    HhotLeft = Htemp;

temporalHot = true;
temporalCold = false;
}

IntervalsH[i].FindDTlm();

i++;
}

cout << endl; // for the debugger

// fclose(hotPoints);
// fclose(coldPoints);

/*
 double ThotLeft, ThotRight, TcoldLeft, TcoldRight, Ttemp;
 double HhotLeft, HhotRight, HcoldLeft, HcoldRight,
     Htemp;
 int i = 0;
 bool temporalCold = false, temporalHot = false;

 IntervalH** IntervalsH;
 IntervalsH = new IntervalH*[7];
 //!!!!!!!!!!!!!!!!!!!!!!!!!!! temporal solution to the size

 FILE * hotPoints;
 FILE * coldPoints;
hotPoints = fopen("hotCurvePoints.dat","r");
coldPoints = fopen("coldCurvePoints.dat","r");

fscanf(hotPoints,"%lf %lf\n", &HhotLeft, &ThotLeft);
fscanf(coldPoints,"%lf %lf\n", &HcoldLeft, &TcoldLeft);

while( !(feof(hotPoints) && feof(coldPoints))){
    if(temporalHot==false){
        fscanf(hotPoints,"%lf %lf\n", &HhotRight, &ThotRight);
    }
    if(temporalCold==false){
        fscanf(coldPoints,"%lf %lf\n", &HcoldRight, &TcoldRight);
    }
    if(HhotRight==HcoldRight){ // the end of the interval has two points of the same enthalpy
        IntervalsH[i]= new IntervalH(TcoldLeft, TcoldRight,ThotLeft,ThotRight, HcoldLeft,HcoldRight);
        TcoldLeft=TcoldRight;
        HcoldLeft=HcoldRight;
        ThotLeft=ThotRight;
        HhotLeft=HhotRight;
        temporalCold=false;
        temporalHot=false;
    } else if (HcoldRight>HhotRight){ // the point on the hot curve creates the interval (has lower enthalpy)
        Htemp=HhotRight;
        Ttemp=((TcoldRight-TcoldLeft)*(Htemp-HcoldLeft)/(HcoldRight-HcoldLeft))+TcoldLeft;
        IntervalsH[i]= new IntervalH(TcoldLeft, Ttemp,ThotLeft,ThotRight,HcoldLeft, HhotRight);
        TcoldLeft=Ttemp;
        HcoldLeft=Htemp;
        ThotLeft=ThotRight;
        HhotLeft=HhotRight;
        temporalCold=true;
        temporalHot=false;
    } else if (HhotRight>HcoldRight){ // the point on the cold curve creates the interval (has higher enthalpy)
        Htemp=HcoldRight;
        Ttemp=((HhotRight-HhotLeft)*(TcoldRight-TcoldLeft)/(HhotRight-HcoldLeft))+TcoldLeft;
        IntervalsH[i]= new IntervalH(TcoldLeft, Ttemp,ThotLeft,ThotRight,HcoldLeft, HhotRight);
        TcoldLeft=Ttemp;
        HcoldLeft=Htemp;
        ThotLeft=ThotRight;
        HhotLeft=HhotRight;
        temporalCold=false;
        temporalHot=true;
    }
}
} else if (HhotRight > HcoldRight) { // the point on the cold curve creates the interval (has lower enthalpy)
    Htemp = HcoldRight;
    Ttemp = ((ThotRight - ThotLeft) * (Htemp - HhotLeft) / (HhotRight - HhotLeft)) + ThotLeft;
    IntervalsH[i] = new IntervalH(TcoldLeft,
        TcoldRight, ThotLeft, Ttemp, HcoldLeft,
        HcoldRight);

    TcoldLeft = TcoldRight;
    HcoldLeft = HcoldRight;
    ThotLeft = Ttemp;
    HhotLeft = Htemp;

    temporalHot = true;
    temporalCold = false;

    IntervalsH[i]->FindDTlm();
    i++;
}

fclose(hotPoints);
fclose(coldPoints); /*

//## end Problem::CalculateEnthalpyIntervals%4DCA79A00127.
body

void Problem::CalculateTargetArea ()
{
    //## begin Problem::CalculateTargetArea%4DD51568008D.body
    preserve=yes

    int i;
    double TargetA=0;

    for(i=0; i<IntervalsH.size()-1; i++){
        IntervalsH[i].FindSumQh();
        IntervalsH[i].FindArea();
        TargetA = TargetA + IntervalsH[i].getarea();
    }

    this->_targetArea = TargetA;
}
```c++
void Problem::CalculateTargetCost(double a, double b, double c )
{
    // ## begin Problem::CalculateTargetCost%4DDE6AAC0287.body
    _targetCost = _nExchangers*(a+b* pow(( _targetArea/_nExchangers),c));
    // ## end Problem::CalculateTargetCost%4DDE6AAC0287.body
}

void Problem::FindNumExchangers()
{
    // ## begin Problem::FindNumExchangers%4DDF7CDC004E.body
    double TpinchHot, TpinchCold;
    int Above=0, Below=0, i;

    TpinchHot = _tPinch + ( _dtMin/2);
    TpinchCold = _tPinch - ( _dtMin/2);

    for (i=0; i< (getnColdStreams() + getnHotStreams()); i++){
        if (Streams[i]->getType()==COLD){
            if( Streams[i]->getIn() < TpinchCold){
                Below++;
            }
            if( Streams[i]->getOut() > TpinchCold){
                Above++;
            }
        }
        if (Streams[i]->getType()==HOT){
            if( Streams[i]->getIn() > TpinchHot){
                Above++;
            }
            if( Streams[i]->getOut() < TpinchHot){
                Below++;
            }
        }
    }
    // ## end Problem::FindNumExchangers%4DDF7CDC004E.body
    if(_includeUtilities==true){
        // Code for _includeUtilities
    }
}
```
for (i=0; i<(getnUtilitiesCold() +
getnUtilitiesHot()); i++){
    if (Utilities[i]->get_type() ==
COLD){
        if (Utilities[i]->getIn
() < TpinchCold){
            Below++;
        }
        if (Utilities[i]->
getOut() >
TpinchCold){
            Above++;
        }
    }
    if (Utilities[i]->get_type() ==
HOT){
        if (Utilities[i]->getIn
() > TpinchHot){
            Above++;
        }
        if (Utilities[i]->
getOut() <
TpinchHot){
            Below++;
        }
    }
}

_nExchangers=(Above-1)+(Below-1);

// ## end Problem :: FindNumExchangers%4DDF7CDC004E.body
}

double Problem::CalculateInterest (double i, double n)
{
    // ## begin Problem :: CalculateInterest%4DDF9C810259.body
    preserve=yes
    return (i* pow((1+i),n))/(pow((1+i),n)-1);
    // ## end Problem :: CalculateInterest%4DDF9C810259.body
}

void Problem::CalculateInteriorCurves ()
{
    // ## begin Problem :: CalculateInteriorCurves%4DE5E2CC0313.body
    preserve=yes

double Ttemp;
double Htemp;
int j=0, k=0;
int coldsize = Curves[1].Points.size() - 1;
int hotsize = Curves[2].Points.size() - 1;

// picks the first point in the cold curve and compares it to the points in the hot curve to find the point corresponding to the same enthalpy
j++;
}

Htemp = Curves[1].Points[0].geth();

// We add this point to the new trimmed hot curve. We use Curves[4] for two reasons. It is empty of points but the curve is already created, and the method to create enthalpy intervals gets data from curves[3] and curves[4] so we can leave it unchanged.
Curves[4].Points.push_back(Point(Htemp, Ttemp));

j++;
for(j; j < Curves[2].Points.size(); j++){
}

// we do the same process but picking the last point in the hot curve, and we repeat the process but traveling the indexes backwards.
j = coldsize;

    j--;
}

Htemp = Curves[2].Points[hotsize].geth();

for(j = 0; j < Curves[1].Points.size(); j++){

if(Curves[1].Points[j].geth()<=Htemp && Curves[1].Points[j+1].geth()>=Htemp){
    Curves[3].Points.push_back(Point(Htemp,Ttemp));
    break;
}

// ## end Problem::CalculateInteriorCurves%4DE5E2CC0313.body

void Problem::RetrievePaths(){
    // ## begin Problem::RetrievePaths%4DEF790202E4.body preserve=yes
    char temp[256];
    ifstream tempFile("pathsettings.txt",ifstream::in);

    for(int i=0; i<5; i++){
        tempFile.getline(temp,256);
    }
    tempFile.getline(temp,256);
    setOctavePath(temp);
    tempFile.getline(temp,256);
    setProgramPath(temp);

    tempFile.close();
    // ## end Problem::RetrievePaths%4DEF790202E4.body

    // Additional Declarations
    // ## begin Problem%4D89B5CD0251.declarations preserve=yes
    // ## end Problem%4D89B5CD0251.declarations
    //## begin module%4D89B5CD0251.epilog preserve=yes
    //## end module%4D89B5CD0251.epilog

D.2.8 StdAfx.cpp

// stdafx.cpp : source file that includes just the standard
// includes
// PinchDlg.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type
// information

Daniel Guitart Pagola
D.2.9 Stream.cpp

```cpp
#include "stdafx.h"

// ## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code
// generator
// ## end module%1.4%.codegen_version

// ## begin module%4D89B60D0280.cm preserve=no
// %X% %Q% %Z% %W%
// ## end module%4D89B60D0280.cm

// ## begin module%4D89B60D0280.cp preserve=no
// ## end module%4D89B60D0280.cp

// ## Module: Stream%4D89B60D0280; Pseudo Package body
// ## Source file: C:\Documents and Settings\user\Mes documents\DOCUMENTS\STAGE D. GUITART\PinchSoftware\PinchDlg2\Stream.cpp

// ## begin module%4D89B60D0280.additionalIncludes preserve=no
// ## end module%4D89B60D0280.additionalIncludes

// ## begin module%4D89B60D0280.includes preserve=yes
#include "stdafx.h"
// ## end module%4D89B60D0280.includes

// Stream
#include "Stream.h"

// ## begin module%4D89B60D0280.additionalDeclarations preserve=yes
// ## end module%4D89B60D0280.additionalDeclarations

// Class Stream
Stream::Stream ()

{ //## begin Stream::Stream%4DC7CBED02CE.hasinit preserve=no
//## end Stream::Stream%4DC7CBED02CE.hasinit

//## begin Stream::Stream%4DC7CBED02CE.initialization
//## end Stream::Stream%4DC7CBED02CE.initialization

}
```

Daniel Guitart Pagola
Stream::Stream (double Tin, double Tout, double FCp, double h)

// ## begin Stream::Stream%4D8C73750315.hasinit preserve=no
// ## end Stream::Stream%4D8C73750315.hasinit

// ## begin Stream::Stream%4D8C73750315.initialization
// preserve=yes
// ## end Stream::Stream%4D8C73750315.initialization

{
// ## begin Stream::Stream%4D8C73750315.body preserve=yes
    this->_tIn = Tin;
    this->_tOut = Tout;
    this->_FCp = FCp;
    this->_H = h;
// ## end Stream::Stream%4D8C73750315.body
}

Stream::~Stream() {
    // ## begin Stream::~Stream%4D89B60D0280_dest.body preserve=yes
    // ## end Stream::~Stream%4D89B60D0280_dest.body

    // Additional Declarations
    // ## begin Stream%4D89B60D0280.declarations preserve=yes
    // ## end Stream%4D89B60D0280.declarations

    // ## begin module%4D89B60D0280.epilog preserve=yes
    // ## end module%4D89B60D0280.epilog

D.2.10 STREAM_TYPE.cpp

// ## begin module%1.4%.codegen_version preserve=yes
// Read the documentation to learn more about C++ code generator
// versioning.
// ## end module%1.4%.codegen_version

// ## begin module%4DA2EFE80005.cm preserve=no
// %X% %Q% %Z% %W%
// ## end module%4DA2EFE80005.cm

// ## begin module%4DA2EFE80005.cp preserve=no
// ## end module%4DA2EFE80005.cp

// ## Module: STREAM_TYPE%4DA2EFE80005; Pseudo Package body
// ## Source file: C:\Documents and Settings\Raphaele.Thery\Bureau\STAGE D. GUITART\PinchSoftware\PROJET C++\STREAM_TYPE.cpp

Daniel Guitart Pagola
D.2.11 Utility.cpp

```cpp
#include "stdafx.h"

// Class STREAM_TYPE

STREAM_TYPE::~STREAM_TYPE()
{
    // Additional Declarations
}
```

Daniel Guitart Pagola
cpp

// ## begin module%4DC7A71F00E7.additionalIncludes preserve=no
// ## end module%4DC7A71F00E7.additionalIncludes

// ## begin module%4DC7A71F00E7.includes preserve=yes
#include "stdafx.h"
// ## end module%4DC7A71F00E7.includes

// Utility
#include "Utility.h"
// ## begin module%4DC7A71F00E7.additionalDeclarations preserve=yes
// ## end module%4DC7A71F00E7.additionalDeclarations

// Class Utility
Utility::Utility (double Tin, double Tout, double F Cp)
{  
    this->_tIn = Tin;
    this->_tOut = Tout;
    this->_fcp = FCp;
}
Utility::~Utility()
{
    // ## begin Utility::~Utility%4DC7A71F00E7_dest . body preserve=yes
    // ## end Utility::~Utility%4DC7A71F00E7_dest . body
}

// Additional Declarations
// ## begin Utility%4DC7A71F00E7.declarations preserve=yes
// ## end Utility%4DC7A71F00E7.declarations
// ## begin module%4DC7A71F00E7.epilog preserve=yes
// ## end module%4DC7A71F00E7.epilog