Exercise Session 10

January 18, 2013

We would like to intragrate in the program `AllDynamic` the possibility to handle also the integrands with a factory. In particular we would like to collect some dynamic libraries with different integrands, using both classes functor and free functions. The number of the libraries is unknown and their loading still have to be dynamic. The chosen of the integrand is done in the data file, handled by GetPot, using the key associated in the registration process. Each library has to be independent from the others one and has to resister autonomously their products. In this way the adding of a new library is just a modification in the data file, the main program has not to be recompiled.

Modifications on the class **class** Factory are not allowed. In the sequel there are present some useful points that might be followed. Beware that each modification of the code might also affect the `Makefile`.

1. Do not use the class **class** udfHandler, using the new factory instead.
2. Using the function wrapper std :: function, of the standard library, to uniformly register the integrands into the factory.
3. Create a new proxy class for the registration of the integrands. Discuss why it is useful to split the builder function from the class proxy.
4. Remove the **extern** "C" from the header files containing the free functions. Create a new file, or use the corresponding source file, for the registration using the proxy previously introduced.
5. Introduce also a class functor, in a separate library, and use it.

To increase the robustness of using the quadrature rules with the factory, statically check if the template parameter ConcreteProduct inherits from AbstractProduct_type, defined into the factory. Furthermore the static member Build has to be converted in the corresponding type defined into the factory. Use the std :: traits and the static_assert for this purpose. Statically checked means that these features have to be checked at compile time.

How to use the program The structure of the folder `exercice` is the following.

```
.
`-- src
    |-- QuadratureRule
    |   |-- AllDynamic
    |   |-- baseVersion
    |   `-- generic_factory
    `-- Utilities
        `-- include
```

To use the programm edit the `Makefile.inc` in the folder `exercice` specifying the absolute path of the folder `exercice` in the variable `PACS_ROOT`. Moreover beware the location of your compiler, specify it in the variable `CXX`. Create two empty folder called `include` and `lib` at the same level of `src`. Go in the directory `src/Utilities` and type `make install`. Now go in the directory `src/QuadratureRule/baseVersion` and type `make dynamic; make install`. In the directory `src/QuadratureRule/generic_factory` type `make install`. Go in the directory `src/QuadratureRule/AllDynamic` and type `make dynamic; make exec`. Before running the program set the `LD_LIBRARY_PATH`. You can use the command `make` to suggest the correct path for the command.