AFEPACK SOLVER CONSTRUCTION BASED ON OPENFOAM

CAO YUAN[1], YAO CHENGBAO[2], SU JUNWEI[3]
[1]*Northwest Institute of Nuclear Technology, caoyuan@nint.ac.cn*
[2]*Northwest Institute of Nuclear Technology, yaochengbao@nint.ac.cn*
[3]*Xi'an Jiaotong University, sujunwei@mail.xjtu.edu.cn*

Keywords: *OpenFOAM, AFEPack, dictionary, runtime selection, solver*

As the strongest open source CFD package now, OpenFOAM[1] has open and complete upper layer. It also provides a lot of compiled solvers, so it gets widespread applications. Especially, its dictionary and hash table function make parameter queries faster and more convenient than before. Its runtime selection mechanism uses virtual function of C++, which makes program dynamically selecting different models during runtime without compiling the program again. It is useful to build solvers and expand functions.

AFEPack[2] is an adaptive finite element package developed by professor Li Ruo of Beijing University with C++ language. It provides basic and general support to finite element method and finite volume method. It helps AFEPack users to achieve finite element program and finite volume program with parallel and adaptive function very conveniently. Unfortunately, AFEPack doesn't has model selection mechanism, so programs based on AFEPack can't form solvers, which goes against for function expansion of AFEPack programs.

Based on the dictionary, hash table and runtime selection mechanism of OpenFOAM, we reconstruct AFEPack program to applicable solver. Specifically, aiming at the characteristics and flow of numerical simulation using AFEPack finite element and finite volume methods, selectors are provided for links needed selections, then parameter dictionary and flow controller are formulated for parameter and flow control, finally the conversion from program to solver is achieved. The specific modification plan is shown in Figure 1.

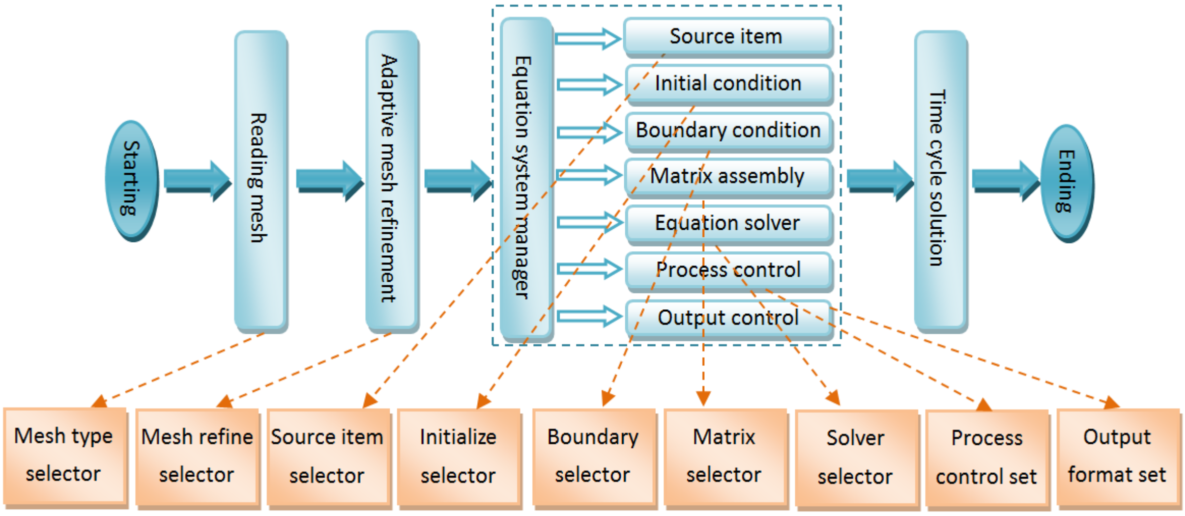
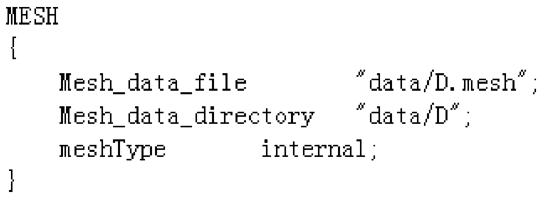


Figure 1: Modification scheme of AFEPack program to solver

There are nine modules in specific programming, such as mesh reading module, mesh refinement module, source item handling module, initialize module, boundary handing module, matrix assembly/flux calculation module, algebraic equations solving module, flow control module, output format setting module. Here we take mesh reading module as an example to explain its implementation process.

Mesh reading module is used to select the way to generate mesh for users. Mesh dictionary shown in Figure 2 is read to get key word after meshType.



```
MESH
{
    Mesh_data_file          "data/D.mesh";
    Mesh_data_directory     "data/D";
    meshType        internal;
}
```

Figure 2: Mesh dictionary

The code call structure of mesh reading module is shown in Figrue 3.

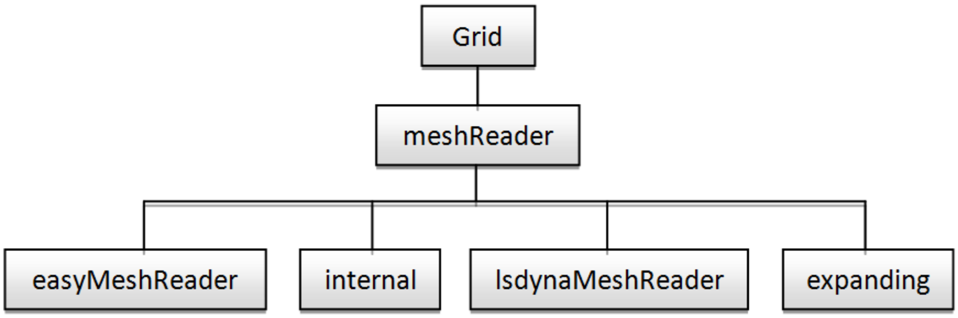


Figure 3: Code call structure of mesh reading module

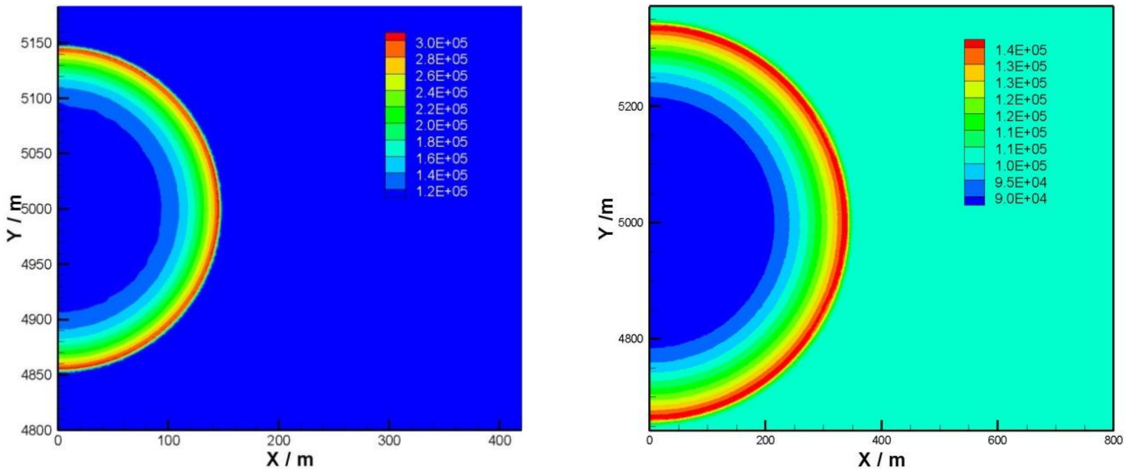
First meshreader object is used in Grid class, then the way of building meshReader is select by runtime selection table. There are three format to build meshReader now, such as easyMeshReader， internal and lsdynaMeshReader. Other formats can be expanded too, which is only need to add subclasses, without modifications of existing codes. The main code files and their functions is shown in Table 1.

Table 1: Main code file and their functions of mesh reading module

Code file	functions
meshReader.C	Definite hash table, read mesh from string information flow and generate file.
meshReader.H	Declare hash table, and select different mesh type during runtime.
meshReaderNew.H	Read mesh type string after meshType.
internal.C	Add internal subclass to hash table.
internal.H	Achieve virtual function of mesh reading， and read mesh file of internal format to string information flow.
easyMeshReader.C	Add easyMeshReader subclass to hash table.
easyMeshReader.H	Achieve virtual function of mesh reading， and read mesh file of easymesh format to string information flow.
lsdynaMeshReader.C	Add lsdynaMeshReader subclass to hash table.
lsdynaMeshReader.H	Achieve virtual function of mesh reading， and read mesh file of lsdyna format to string information flow.

The realization principle of other modules are similar to mesh reading module, here omitted. Then the AFEPack solver is formed.

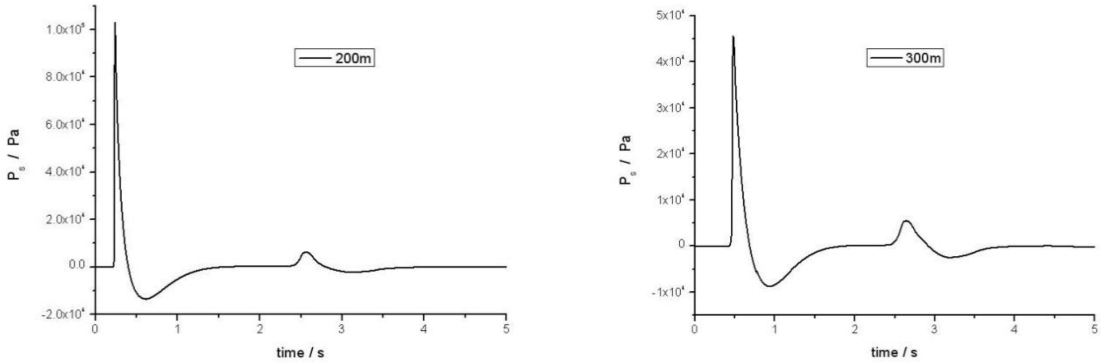
At last we use an example to validate the correctness of the reconstruction, which is an AFEPack program to simulate strong explosion shock wave. The conditions are even atmosphere and 1kt TNT equivalent. We use dictionary file of args.txt to set selector options, which can be specified on demand by users, here omitted. The shock wave pressure of flow field on different time are shown in Figure 4.



(a) **(b)**

Figure 4: Flow field cloud of shock wave pressure on different time

Shock wave over pressure history curve at different distance to burst heart by calculation shown in Figure 5 can correctly describe shock wave positive pressure, negative pressure and secondary shock at certain positions in strong explosion. It shows the calculation is reasonable and the result is reliable.



(a) 200m

(b) 300m

Figure 5: Shock wave over pressure history curve at different distance to burst heart

Shock wave overpressure peak, positive pressure impulse, shock wave arrival time and positive pressure action time of solver result are all same with that of the program before. It shows the correctness of the program is not changing in the solver reconstruction. At the same time, the selection functions of the selectors, such as mesh file, initial condition, boundary condition and solution algorithm are all achieve scheduled goals.

As a solver, it has more selectivity and applicability than the program before. Users only need to assignment parameters by dictionary file, then different solution functions can be implemented dynamically. So the solver has stronger functions than before.

Based on the dictionary, hash table and runtime selection mechanism of OpenFOAM, we reconstruct AFEPack program to applicable solver, which expanding dynamical selection functions of program largely. It shows the feasibility and correctness of the reconstruction through example validation. The reconstruction ideas and technologies are universal, and can be applied to the reconstruction from programs based on other open source package to solvers.

References

[1] OpenCFD, OpenFOAM: The Open Source CFD Toolbox. User Guide Version 1.4, OpenCFD Limited. Reading UK, Apr. 2007.

[2] Li R., Liu W. B.. http://circus.math.pku.edu.cn/AFEPack, 2013-08-12.