OFELI

An Object Oriented Finite Element Library

Release 3.1.0

Rachid Touzani
Laboratoire de Mathématiques Blaise Pascal
Université Clermont Auvergne
63177 Aubière, France
e-mail: Rachid.Touzani@univ-bpclermont.fr

This document is a brief presentation of a library of C++ classes for the development of object oriented finite element codes. The library (called **OFELI** for Object Finite Element Library) is a toolkit of utility functions. The numerical solution of the linear system of equations can be performed either using direct methods or more sophisticated preconditioned iteration techniques.

The package contains a set of examples of finite element codes that can serve as prototypes for more sophisticated applications.

The **OFELI** library was mainly developed by R. TOUZANI. Other people have contributed to this development either by writing a particular class, or by testing some parts of the library or simply by fruitful discussions.

Rachid Touzani

March 2016

Contents

1 Introduction

Object oriented programming is widely used in scientific computing and particularly in finite element coding. The finite element method is well adapted to object oriented coding for the following reasons:

1. Finite element mesh data structuring is handled in an elegant and efficient way. Moreover, for adaptive mesh strategies, object techniques are well adapted to manipulating mesh elements and nodes. For instance, removing or adding an element can be easily handled by using list structures to store element data.

2. Implementation of algorithms for the linear system of equations has to take matrix storage schemes into account. For this, overloading and template features in C++ are well adapted and help implement storage independent codes.

3. Using other C++ finite element libraries is handled more efficiently thanks to encapsulation properties of this language.

This document presents an object oriented library of classes for finite element solution of boundary and initial value problems. The library is called **OFELI** (*Object Finite Element Library*). It provides a finite element developer tools to write concise and user friendly codes either for simple finite element programs or for large scale applications using sophisticated solution techniques like preconditioned iteration algorithms.

The library **OFELI** is written in ANSI C++ and was tested on the most used computer systems.

As in any C++ code, **OFELI** is programmed in such a way that a finite element code developer manipulates objects rather than data. Hence, a collection of classes is defined. Each class corresponds to a particular type of information in a Finite Element computation.

This document constitutes a general overview in which we review some considerations about the implementation of the **OFELI** library and then describe the most used classes. The structure of our standard mesh data is then given. We then define the installation procedure for both WINDOWS and UNIX–like systems.

2 The OFELI Package

The **OFELI** package is <u>more than a finite element code</u>. Actually, **OFELI** is a (toolkit) library of classes for Finite Element developments.

The current version of the package contains extensive documentation:

1. **OFELI**, *An object Finite Element Library*: The present document.

2. **OFELI**, *User's Guide* : Here we present some examples and then a tutorial to develop finite element codes and new classes in **OFELI**.

3. **OFELI**, *Reference Guide* : This document contains a detailed description of each class of the kernel library. Naturally, only public members and attributes of classes are outlined. Notice that problem dependent classes are not in this document. For this, the following documents are to be consulted.

4. **OFELI**, *Mathematical Foundations*: This document is currently in preparation.

3 License Agreement

OFELI is free but Copyrighted software. It is distributed under the terms of the **GNU Lesser General Public License (LGPL)**. Details of the LGPL are in the file COPYING that comes with the **OFELI** software package. More details can be found in the site http://www.gnu.org/licenses/.

4 Programming Considerations

Let us outline some of the principles we have followed in programming the **OFELI** library.

4.1 Namespaces

The whole package **OFELI** is included in the *namespace* OFELI. This feature allows combining the use of **OFELI** with other packages.

4.2 Variable Names

To clarify the programming, all classes have names that start with a capital letter like Node or Mesh. Moreover, all public members and attributes share this property, private or protected members having no capital letters and beginning with an "underscore" sign _.

4.3 Inheritance

For efficiency reasons, we did not make an extensive use of inheritance between classes. The only inheritances are from abstract classes and are therefore transparent for a developer. Of course inheritance advantages can be exploited by a developer to complete the **OFELI** classes.

1. All matrix classes inherit from a template abstract matrix class called Matrix<>. This was useful for implementing some general purpose manipulation of matrices that do not depend on the storage.

2. All finite element equation classes derive from the template abstract template class AbsEqua<>.

3. Finite element shape classes inherit from an abstract class (called FEShape).

4. Mesh oriented vectors all inherit from an abstract class called AbsVect.

4.4 Template classes

We have made use of template classes in vector and matrix classes as well as in finite element equation abstract class. This feature enables implementing for instance complex valued problems.

4.5 Pre and Post Processing

The **OFELI** library contains a two-dimensional mesh generator without graphics facilities. The package also contains utility applications to convert input and output files for most popular mesh generators and graphical postprocessors. For any pre or postprocessing, we recommend the use of most popular opensource software (see documentation on file converters).

5 Classes in OFELI

To each type of data and each phase in a finite element computation corresponds a C++ class. For the sake of clarity, we shall outline hereafter these classes through the steps of execution of a finite element code. The reader can consult the details of each class and its members in the reference manual.

5.1 Mesh data

Mesh data are introduced by a class called `Mesh`. This class allows to read, manipulate and store mesh data. Moreover, to each type of mesh data corresponds a class. Thus, an instance of class `Mesh` is a collection of instances of classes `Node`, `Element`, and `Side`. A finite element mesh is defined by a list of nodes given by their coordinates and a list of elements given by their node numbers. Moreover, to each node is associated its number of degrees of freedom and a code to each degree of freedom. This code is useful to prescribe node boundary conditions data (Dirichlet). In addition, to take Neumann–like boundary conditions into account, a mesh can contain a collection of sides. In practice these must be sides on the boundary of the domain but any side can be defined with the help of mesh nodes.

5.2 Vectors

These classes were developed to facilitate basic operations on vectors. An overloading of operators (), [] and most algebraic operations is implemented to simplify this access.

5.3 Matrices

In order to consider several types of storage for finite element matrices, the library **OFELI** contains a class for each storage type. Moreover, to handle operations that are independent of these storage types we have a template abstract class called `Matrix`. The implemented storage schemes are tridiagonal, skyline and sparse.

5.4 Boundary conditions, Forces, . . .

A finite element computation needs several types of data. These data can be either given *manually* or via a class written by the developer and that inherits from an abstract class called `UserData`. Another way to introduce data is to describe them through *prescription files*.

5.5 Material data

In order to define material properties, material data files are given in a specific directory. Each material has its own file that contains corresponding physical constants and expressions. A default material is defined with default values for constants.

5.6 Finite Element Equations

A crucial step in the implementation of the finite element method consists in building up finite element equations for each element and assembling them into a global linear system of equations. A nonlinear problem is solved by an iteration algorithm where each iteration consists in solving a linear problem. At the element level, finite element equations are to be declared as instances of a class of finite element equations. Clearly, the finite element equation depends on the problem to solve. For this, some classes of typical problems (Thermics, Elasticity, Fluid, . . .) are already developed and can serve as prototypes for developing classes for new problems.

5.7 Finite Element Shapes

The above classes need information about finite element interpolation (choice of element geometry, shape functions, . . .). The **OFELI** library provides a family of classes that implement several types for finite element shapes. For instance, class `Triang3` corresponds to P_1 (or three-node) triangle.

5.8 OFELI Files

The library defines a standard for input and output files: All input and output files are XML files as described in the User's guide of the library.

6 Installation

The **OFELI** library can be installed on any computer provided with an ANSI C++ compiler. The library is hosted by the Sourceforge server with the home page:

```
http://www.ofeli.net
```

This web page contains a brief presentation of **OFELI** and offers the possibility to download the package.

6.1 Installation Procedure on UNIX-like systems

The following instructions apply to UNIX (like) systems only. The only thing you need is a C++ compiler.

A clean installation can be performed in the following steps:

- 'gunzip' and 'untar' the downloaded file.

- Execute the configuration script by typing :

  ```
  ./configure
  ```

  A first execution with argument `--help` displays a short manual of the script. A typical execution looks like:

  ```
  ./configure --prefix=/home/me/ofeli --libdir=/home/me/lib
  --bindir=/home/me/bin --enable-release
  ```

- Then you can build the library by executing the commands

  ```
  make
  ```

  ```
  make install
  ```

- The test suite can be run through the command

  ```
  test_ofeli.sh
  ```

6.2 Installation Procedure on Windows systems

To install on Windows systems, you can use either the Visual C++ compiler or the `CodeBlocks` free software. The installation procedure depends on the file you have downloaded.

- If this is a zip file then you must unzip it, go the subdirectory `Win32` where you can find Visual C++ workspace to install the library, utilities, examples, and demos.

  You can also go to directory `Win32/DevCpp` and make use of `DevCpp` projects.

- If this is an exe file, than this is a standard windows setup installer. The library, example files and demo files are already compiled. You can also recompile using Visual C++ workspace.