FUNCTOR-ORIENTED FINITE ELEMENT PROGRAMMING WITH APPLICATION TO STRUCTURAL TOPOLOGY OPTIMIZATION

P. Tauzowski[1], B. Blachowski[1], and J. Logo[2]

[1]*Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland*
[2]*Budapest University of Technology and Economics, Budapest, Hungary*

e-mail: ptauzow@ippt.pan.pl

1. Problem statement

The subject of this study is an efficient approach to the development of a finite element framework, which is intended to be used for solving a variety of problems in computational solid mechanics. One of such problems, recently becoming an active field of research, is topology optimization of structures made of elastic-plastic materials. For finding the optimal topology of real, practical and complex structures the knowledge of a number of numerical algorithms is required, to mention a few: modification of finite element meshes, aggregation of tangent stiffness matrices, or direct and iterative solvers.

The classical computer implementation of the original Classical Optimality Criteria method (COC) of the topology optimization problem given by Bendsoe and Sigmund [1] is relatively simple and contains 99 lines of code in the MATLAB language. However, it assumes that there exists only a single loading case, single displacement (compliance) constraint, the material is linearly elastic and the optimal topology can be found using the so-called Solid Isotropic Material with Penalization (SIMP) algorithm, which is based on the original COC method. In reality, engineers face a slightly different problem. They need to find the topology of a minimum weight structure subjected to multiple loading cases, made of an elasto-plastic material, and with a limit on stresses. The above mentioned SIMP approach may not lead to an optimal solution [2].

To avoid this obstacle in this study we reformulate the minimum compliance problem so that we look for a minimum weight structure subjected to multiple loading under stress constraints instead of volume fraction constraint. In this way several local constraints are used instead of a single global one.

Mathematically, we can express our approach for topology optimization in the following form

$$\begin{aligned} &\text{Find minimum of} && V(\rho) = \int_V \, dV \\ &\text{Subject to} && \int_V \, \sigma_{ij}\delta\varepsilon_{ij} \, dV - \int_{\partial V} f_i \delta u_i \, d\Gamma = 0 \\ & && |\sigma_{red}| \leq \sigma_0 \\ & && \rho_{min} \leq \rho \leq \rho_{max} \end{aligned}$$

where V is the volume of the structure, σ_{ij} is the stress tensor, $\delta\varepsilon_{ij}$ is the virtual strain, f_i represents surface loading, δu_i is the virtual displacement, σ_{red} is the von Mises stress, σ_0 denotes the yield limit and finally ρ represents the density of the material distribution.

In computational mechanics the above formulation is usually discretized using the finite element method. Then, the structural topology optimization investigated in this study can be expressed in the following form

$$\begin{aligned} &\text{Find minimum of} && V(\boldsymbol{\rho}) = \textstyle\sum_e V_e \\ &\text{Subject to} && \mathbf{K}\mathbf{u} - \mathbf{f} = \mathbf{0} \\ & && |\boldsymbol{\sigma}_{red}| \leq \sigma_0 \mathbf{I}_\rho \\ & && \rho_{min}\mathbf{I}_\rho \leq \boldsymbol{\rho} \leq \rho_{max}\mathbf{I}_\rho \end{aligned}$$

where $\mathbf{K}$ is the tangent stiffness matrix, $\mathbf{u}$ is the vector of displacement, $\boldsymbol{\rho}$ is the vector of material distribution density, and $\mathbf{I}_\rho$ is the identity vector.

To find the optimal structural topology using the above formulation a large number of iterations and finite element analyses are required. To compensate this great demand on computational power we propose an effective architecture of our FE code, which is based on the functor-oriented programming paradigm [3].

2. Functor based finite element programming

The classical formulation of finite elements usually contains a class called finite element whose purpose is not only to approximate some fields (displacements, temperature or own geometry) but also to define matrices necessary for a particular analysis (Listing 1). It often leads to a sophisticated class hierarchy of finite elements.

Listing 1.

```
class FiniteElement
{
     vector<Node> nodes;
     Material *mat
public:
...
  virtual Vector getX( Vector ksi)  = 0;
  virtual Matrix getStifnessMatrix()= 0;
  virtual Matrix getMassMatrix()    = 0;
  virtual Matrix getTangentMatrix() = 0;
...
};
```

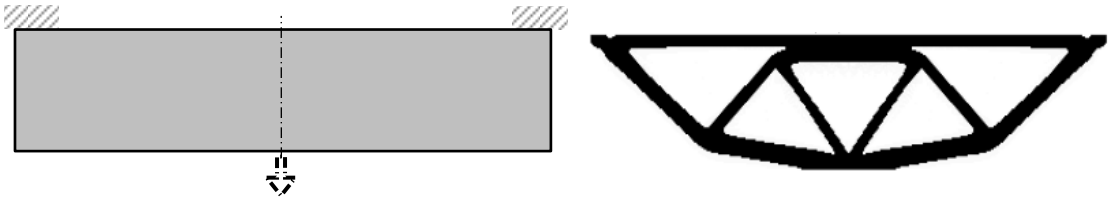
In our approach matrices necessary for an analysis (tangent matrix, mass matrix) are in separate classes. The hierarchy of these classes can be developed almost separately from finite elements. Also the finite elements hierarchy is much smaller. Because each class represents one kind of matrix computed in the analysis, the best (in our opinion) kind of object to use in this case is a functor. A functor represents one subroutine and it can also be called as a function: `functor(FiniteElement)`. (Listing 2). By contrast, in the classical approach each matrix function in a finite element has its own unique name. It makes aggregation process more complex. The connection of the functor with the finite element is only accomplished through the call parameter. In the other words, the finite element is the parameter of the functor. Another advantage of our approach is the use of templates. This allows us to avoid the overhead associated with virtual function call in those places of the code, where it is especially important (e.g. calculations at Gauss points).

Listing 2.

```
template <class T>
class TElemFunctor
{
public:
  TElemFunctor( const DTvec &dv):m_dofs( dv){ }
  virtual TElemFunctor* Clone() const = 0;
  const T& GetValue() const { return m_value; }
  virtual const T& operator()
       (  const CFEInstance &ielem  ) = 0;
protected:
    T m_value;
};
```

3. Topology optimization under stress constraints

The effectiveness of functor-oriented programming in optimal design has been demonstrated on several examples including benchmark problems like cantilever or simply supported beam (Figure 1).



a) Design domain b) Optimal solution

Figure 1. Optimal topology of the simple supported beam.

Acknowledgments The Authors are grateful for financial support provided by the National Research, Development and Innovation Office (grant K 119440) - Joint grant of the Hungarian and the Polish Academy of Sciences.

References

[1] M. P. Bendsoe, O. Sigmund, *Topology Optimization: Theory, Methods, and Applications*, Springer-Verlag, 2004.
[2] K. Mela, J. Koski, *On the equivalence of minimum compliance and stress-constrained minimum weight design of trusses under multiple loading conditions*, Struct Multidisc Optim, 46: 679, 2012.
[3] D. Vandevoorde, N. M. Josuttis, D. Gregor, *C++ Templates - The Complete Guide*, 2nd ed., Addison-Wesley, 2017.