OBJECT-ORIENTED DEVELOPMENT ENVIRONMENT TO COMPUTE MULTIAXIAL FATIGUE CRITERIA

Jean-Luc CHARLES, Thierry PALIN-LUC and Thomas DELAHAY
ENSAM, Laboratoire Matériaux Endommagement Fiabilité
et Ingénierie des Procédes (LAMEFIP) EA 2727
33405 Talence, France

ABSTRACT

This paper focuses on an object-oriented development environment used to apply various multiaxial fatigue criteria on mechanical structures under cyclic loading. Computing acts as a post-processing of data obtained from usual Finite Element Analysis (FEA) of the structure, using an appropriate FEA software. The goal of this programming workbench is to provide mechanicians with efficient tools to develop multiaxial fatigue criteria softwares based on the knowledge of stress/strain tensor fields values over the structure, as obtained from FEA. This project is developed on PCs running Linux, using free tools such as GNU development tools, and Blitz++ numerical library.

Two libraries are briefly presented. They are designed to provide classes corresponding to the various mechanical concepts and mathematical tools used to describe fatigue criteria:

– a specific "Fatigue Criteria" library supports various concepts involved in fatigue criteria definition;
– a "Finite Element" library is used to instantiate the required common objects such as meshes, elements, nodes, ... Special attention is given to scalar and tensor fields (static or dynamic), which are of particular interest for fatigue criteria formulation.

A specific "Interface" library provides classes useful to read data from files created by FEA softwares Arrays manipulations (storage, dynamic allocation, numerical computing ...) are provided by the Blitz++ library. The main fatigue criterion implemented is the volumetric energy based high cycle multiaxial fatigue criterion proposed by Banvillet, Palin-luc and Lasserre (Banvillet [1]). This programming workbench is currently being used for a PhD thesis at LAMEFIP (Delahay [2]) to compute the LAMEFIP volume criterion on various specimens. Validation results are presented here (with elastic behavior hypothesis) against experimental fatigue data from specimens in cast iron EN-GJ8800-2. Other multiaxial fatigue criteria are also programmed.

1 INTRODUCTION

Object-oriented design has taken a growing part among recent scientific programming projects due to the numerous advantages given by this approach and associated tools. This project presents an object-oriented development environment intended to provide mechanicians with classes as close as possible to concepts involved in the mechanical and mathematical description of multiaxial fatigue criteria. The design of this workbench conforms to the "Unified Modeling Language" (UML) standards (Booch [3]), its implementation relies on C++ language (Stroustrup [4]) using GNU C++ compiler (http://www.gnu.org). This paper is focused on the volume energy based high cycle multiaxial fatigue criterion proposed by Banvillet *et al.* (Banvillet [1]).

2 THE VOLUME MULTIAXIAL FATIGUE CRITERION

Details on the mechanical formulation of the volume energy based high cycle multiaxial fatigue criterion can be found in [1]. The main concepts are briefly listed:

– The damage parameter at each point M of the structure is the strain work density given to each elementary volume of material per loading period, Wg(M):

$$W_g(M)=\sum_{i=1}^{3}\sum_{j=1}^{3}\int_T <\sigma_{ij}(M,t)\dot{\varepsilon}_{ij}(M,t)>dt \qquad (1)$$

where $\dot{\varepsilon}_{ij}(M,t)$ is the elastic strain rate tensor field after elastic shakedown; $< a > = a$ if $a > 0$ and $< a > = 0$ if $a \leq 0$.

- The critical nodes C_i are defined as nodes of the structure where Wg has a local maximum.
- A threshold value of Wg (called Wg^*) exists : it corresponds to a stress limit σ^* (lower than the conventional endurance limit σ^D) above which micro-damage initiation can occur.
- The volume $V^*(C_i)$ influencing fatigue crack initiation around each critical node C_i is defined as the set of points M around C_i where $W_g(N) \geq W_g^*(C_i)$.
- The damaging part of the strain work density given per cycle around a critical point C_i is :

$$\varpi_g(C_i)=\frac{1}{V^*(C_i)}\int\int\int_{V^*(C_i)}[W_g(M)-W_g(C_i)]dv \qquad (2)$$

- The multiaxial fatigue criterion is then expressed by $\varpi_g(C_i)\leqslant\varpi_g^D(C_i)$ (3), where $\varpi_g^D(C_i)$ is the value of $\varpi_g(C_i)$ at endurance limit. If (3) is false then a macro crack occurs at the node C_i.

3 CLASSES LIBRARIES

3.1 Fatigue Criteria Library

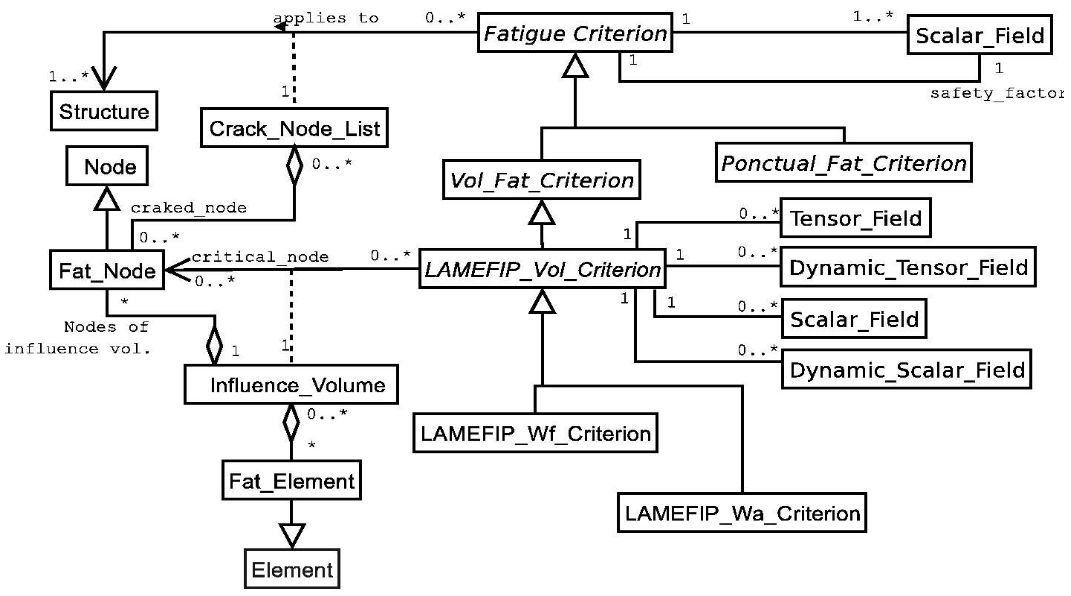


Figure 1: The Fatigue Library class diagram

This library povides classes corresponding to fatigue criteria and associated concepts. Abstract class `LAMEFIP_Vol_Criterion` provides all the main methods used to compute the LAMEFIP volume fatigue criterion:

- `find_critical_node()` finds all the critical nodes of the structure;
- for each critical node C_i, `v_star()` creates an object `Influence_Volume` and uses an iterative algorithm to locate elements belonging to this volume: given the initial set of elements owning C_i , $V^*(C_i)$ is extended to adjacent elements having at least one node N verifying $W_g(N)\geq W_g^*(C_i)$, and so on ... Elements at the boundaries may have all their node critical (fully critical element) or not (partially critical element).

Methods to compute field Wg at both nodes and gauss points are abstract in `LAMEFIP_Vol_Criterion,` so implementation is only given in derived classes (such as

LAMEFIP_Wf_Criterion): this gives an efficient way to implement various multiaxial volume criteria with different definitions of the damage parameter (Von Mises equivalent stress as in Sonsino [5], or effective stress as in Adib [6]).

Class LAMEFIP_Wf_Criterion provides methods to compute Wg scalar field:

– compute_dot_strain_inst_tensor_field(): computes the strain rate tensor field $\dot{\varepsilon}_{ij}$;
– compute_WorkDensity_at_nodes()and compute_WorkDensity_at_gauss_p(): compute scalar field Wg for all nodes and gauss points.

Data are written in output files using classes of the "Interface library". Values written are scalar fields values at nodes and at gauss points of the structure:

– Wg(M) and Wg*(M),
– dT(M): the triaxility degree of loading,
– Rank(M): the influence volume flag (zero if node M is not in an influence volume or else the rank of the influence volume containing node M),
– Crit(M): the critical node flag (boolean value: 1 if node is critical, else 0).
– Crack(M): the crack flag (boolean value: 1 if node is in a cracked volume, else 0).

For each critical node the values of some variables are also printed on screen such as the safety factor $\sqrt{\varpi_g(C_i)/\varpi_g^D(C_i)}$ (homogeneous with the ratio $\sqrt{\sigma/\sigma^D}$ used for stress based criteria).

3.2 Finite Element Library

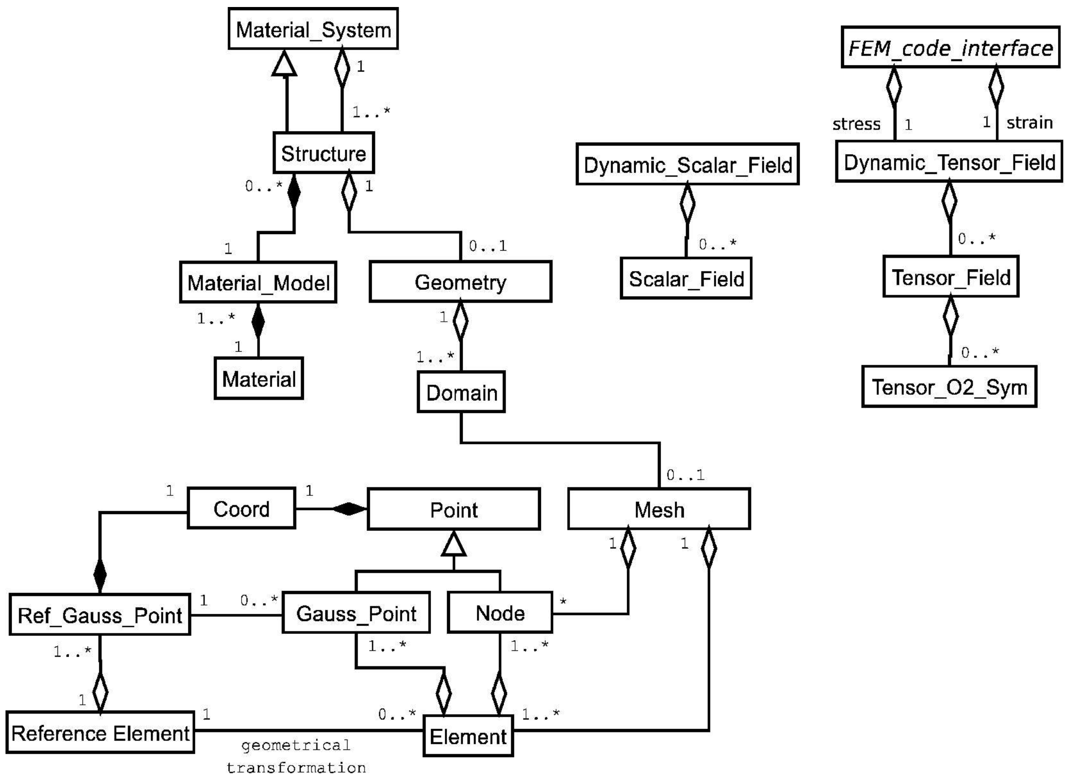


Figure 2: The Finite Element Library class diagram

This library gives the required common finite element classes. Time dependent quantities such as dynamic scalar or tensor fields are represented as an aggregation of these quantities at each time step. Arrays, vectors and all numerical computations on arrays are provided by Blitz++ numerical library (http://www.oonumeric.org/blitz) whose source code is available under the GPL license. Abstract class `FEM_code_interface` provides the conceptual model for the interaction between FEA software and FE library, in order to instantiate stress/strain tensor fields. This class is specialized in the "Interface library" to give concrete classes implementing the interface with specific FEA software.

3.3 Interface Library

Data computed with an appropriate FEA software (mainly stress/strain tensor field and structure nodes) are imported using a set of classes from this "Interface library" so that specific file formats and input/ouput methods are encapsulated within these classes. In the present state, classes have been implemented to interface this workbench with ZéBuLon (FEA software from the «Centre des Matériaux, École de Mînes de Paris, FRANCE») but any other FEA code could be interfaced as soon as it stores computed stress/strain tensor field values in files of known format. Classes allow reading binary data file created by ZéBuLon and writing fatigue criterion data into ZéBuLon files. These files can be graphically post-processed by ZéBuLon to produce various graphs.

4 RESULTS

Results are computed (under elastic behaviour hypothesis) for specimen in SG cast iron EN-GJ8800-2 loaded under combined plane bending and torsion. Figure 3 shows the specimen geometry.

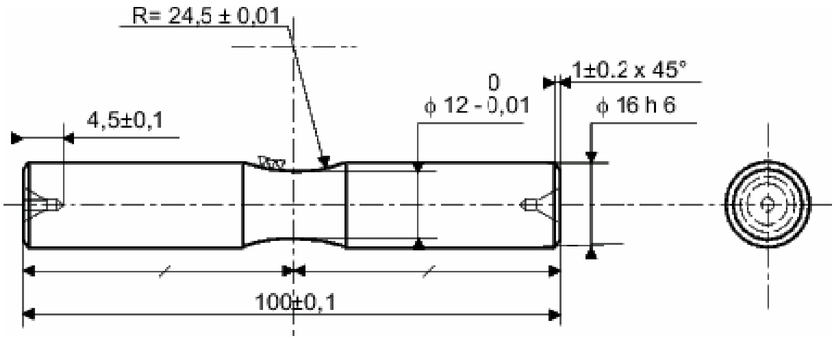


Figure3: Specimen geometry

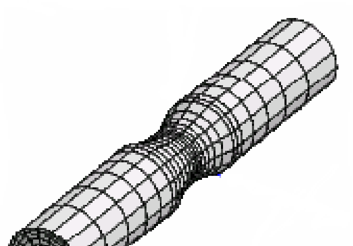


Figure 4: Specimen meshing

Specimen meshing with ZéBuLon 3D quadratic elements gives 2184 elements and 27123 degrees of freedom (the 3 nodal displacements). Figure 5 shows boundary conditions used for FEA :

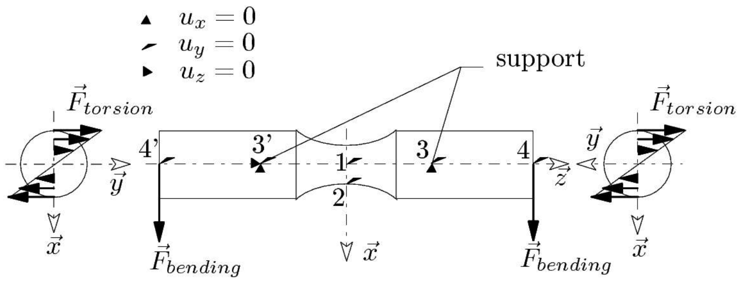


Figure 5: Specimen Loading (plane bending + torsion).

FEA of this specimen using ZéBuLon gives stress/strain tensors at every node and every gauss point. Values of $F_{bending}$ and $F_{torsion}$ are tuned so that the normal stress amplitude σ_a and the shear stress amplitude τ_a at surface nodes located at the lowest diameter torus section have approximately the values of the experimental endurance limit (at 10^6 cycles or more). For such a loading with this specimen geometry Banvillet [1] found $\sigma_a = 199$ MPa and $\tau_a = 147$ Mpa (using the stair-case method with 15 specimens).

Once the FEA run, the LAMEFIP multiaxial volume fatigue criterion is applied to the specimen: figures 6 and 7 show critical node and influence volume. Values of the safety factor is about 1.0 which indicates that some damage occurs inside these volumes.

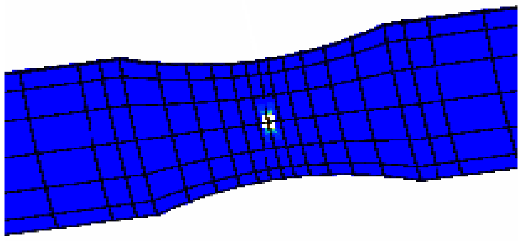


Figure 6: One of the two critical nodes located at the surface median torus

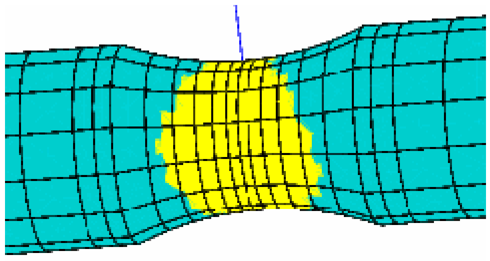


Figure 7: Projection of the influence volume on the specimen surface

The volume width under the specimen surface is plotted on figure 8: scalar field `Vrank` is shown for the nodes along the specimen radius in the cross section containing C1. We can deduce from this plot that the width of the influence volume in which damage occurs is about 1.5 mm.

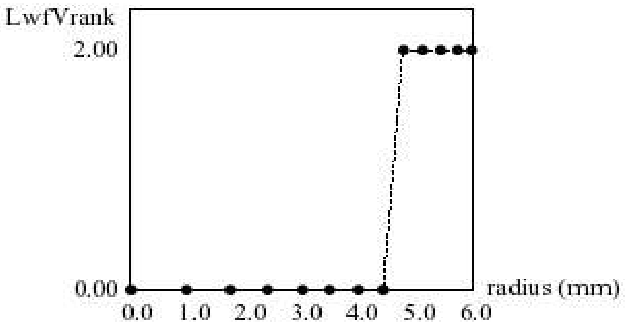


Figure 7: nodes in influence volume number 2 versus radius location

5 CONCLUSION

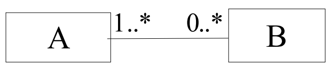
In this paper we have illustrated the use of an object-oriented approach to design and implement an environment development for multiaxial fatigue criteria computing (specially the LAMEFIP volume fatigue criterion). Computed predictions are in good agreement with experimental results obtained from specimens. We plan to improve the design and the implementation of this workbench to take into account more complex behaviors (elastic-plastic behavior) to apply the LAMEFIP volume fatigue criterion on more complex specimen (notched specimens) and on real whole structures. Other usual multiaxial fatigue criteria are also (Crossland, DangVan ...) implemented .

5 REFERENCES

[1] A. Banvillet, T. Palin-luc and S. Lasserre, “A volumetric energy based high cycle multiaxial fatigue criterion”, Int. J. Fatigue, vol 25 (2003) 755-769.
[2] T. Delahay, “Développement d'une méthode probabiliste de calcul en fatigue multiaxiale prenant en compte les gradients de contraintes”, to be published on july 2004, PhD Thesis, ENSAM CER de Bordeaux, France.
[3] G. Booch, J. Rumbaugh, I. Jacobson, “The Unified Modeling Language User Guide”, Addison-Wesley, 1998
[4] B. Stroustrup, The C++ Programming Language, 2nd edition, Addison-Wesley,
[5] C.M. Sonsino, H. Kaufmann, V. Grubisic, “Transferability of material data for the example of a randomly loaded forged truck stub axle”, SAE Tech. Paper Series 970708,1-22, Detroit Feb. 1977
[6] H. Adib, J. Gilbert, G. Pluvinage, “Fatigue life prediction for welded spots by volumetric approach”, Int. J. Fatigue, 26, 81-94, 2004.

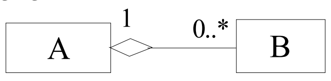
5 APPENDIX : UML graphical notations

Association : structural relation between two classes



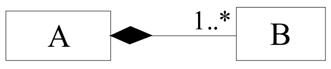
Class A is associated with zero or more occurrences of class B
class B is associated with one or more occurrences of class A

Aggregation : one class contains others



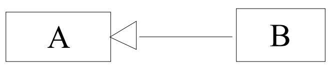
Class A contains 0 or more occurrences of class B
class B is contained in only class A

Composition : one class is “made of ” others



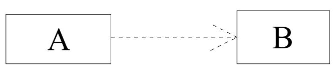
Class A is made of 1 or more occurrences of class B

Generalization : one class is derived from another



Class B ”is a” class A
A generalizes B, class B is a specialization of class A
A is the mother class, B is the daughter class

Dependence : one class depends on another



Class A depends on class B

Association-class : the association between two classes A an B gives raise to another class C