Modelling Microstructures with OOF2

Andrew C.E. Reid

Information Technology Laboratory,
National Institute of Standards and Technology,
100 Bureau Drive, Stop 8910,
Gaithersburg, MD, 20899-8910, USA
E-mail: andrew.reid@nist.gov

Rhonald C. Lua

Materials Science and Engineering Laboratory,
National Institute of Standards and Technology,
100 Bureau Drive, Stop 8555,
Gaithersburg, MD, 20899-8555, USA
E-mail: rhonald.lua@nist.gov

R. Edwin García

School of Materials Engineering,
Purdue University,
West Lafayette, IN, 47907-2044, USA
E-mail: redwing@purdue.edu

Valerie R. Coffman

Materials Science and Engineering Laboratory,
National Institute of Standards and Technology,
100 Bureau Drive, Stop 8555,
Gaithersburg, MD, 20899-8555, USA
E-mail: valerie.coffman@nist.gov

Stephen A. Langer*

Information Technology Laboratory,
National Institute of Standards and Technology,
100 Bureau Drive, Stop 8910,
Gaithersburg, MD, 20899-8910, USA
Fax: 301 975 3553 E-mail: stephen.langer@nist.gov
*Corresponding author

Abstract: `OOF2` is a program designed to compute the properties and local behaviour of material microstructures, starting from a two-dimensional representation, an image, of arbitrary geometrical complexity. `OOF2` uses the finite element method to resolve the local behaviour of a material, and is designed to be used by materials scientists with little or no computational background. It can solve for a wide range of physical phenomena and can be easily extended. This paper is an introduction to some of its most basic and important features.

Keywords: finite element; microstructure.

Reference to this paper should be made as follows: Reid, A.C.E., Lua, R.C., Edwin García, R., Coffman, V.R. and Langer, S.A. (xxxx) 'Modelling Microstructures with OOF2', *Int. J. Materials and Product Technology*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Andrew C.E. Reid has been an active developer of computational modelling tools in the field of materials science for over ten years, including six years with the OOF project at NIST. He holds a PhD in Condensed-Matter Physics from Queen's University in Kingston, Canada.

Rhonald C. Lua developed computational tools for studying knots in lattices, proteins and curves in his PhD research at the University of Minnesota before joining the OOF group at NIST. He also did analytical and computational work on first passage times in diffusion.

R. Edwin García is an Assistant Professor in Materials Engineering at Purdue University in West Lafayette, Indiana, USA. He earned his PhD in Materials Science and Engineering at the Massachusetts Institute of Technology in 2003. His research interests include the theoretical and numerical modelling of materials of complex microstructural features in batteries, actuators, and nanowires.

Valerie R. Coffman is an NRC post-doctoral researcher at NIST. She earned her PhD in Physics from Cornell University in 2006, studying the macroscopic effects of atomic scale defects in crystals.

Stephen A. Langer is a Physicist in the Mathematical and Computational Sciences Division of the Information Technology Laboratory at the National Institute of Standards and Technology. He received his PhD in Physics from Cornell University in 1989.

1 Introduction

There are two general approaches to modelling the properties of materials with complex microstructures. One is to use mean-field approximations and compute the average properties of a statistically representative microstructure. The second is to accurately model a specific microstructure (or set of microstructures), using all available geometrical data. While the mean field method provides useful bounds to calculate the effective properties and response of materials, the second method is useful in situations in which

- the macroscopic properties are a non-linear function of the underlying properties
- the underlying properties are spatially non-uniform
- it is not clear which properties should be averaged (do all grains in an 'average' microstructure have the same size, shape, orientation, and modulus, or do they have a distribution of all or some of those properties?), or
- the properties of interest arise from the extremes of a property distribution (a failure mechanism might depend on the presence of a single point of extreme stress).

The goal of the Object Oriented Finite (OOF) element analysis project at NIST (http://www.ctcms.nist.gov/oof/) is to model microstructures using the second, or direct-computation, approach. With OOF, the user assigns material properties to the features in an experimental or simulated micrograph, generates a finite element mesh, and performs virtual experiments. OOF is used to visualise the microscopic response of the microstructure to external conditions, to compute effective macroscopic material properties, or to design material microstructures of optimal (tailored) performance.

Historically, OOF was developed to perform linear elasticity and thermal conductivity calculations on arbitrary two-dimensional microstructures (Langer et al., 2001). As additional physics was added to OOF, it became clear that extending the OOF1 framework would be a complex and cumbersome task that would make the code difficult to maintain. OOF2, a complete re-write of OOF1, was designed to have a more powerful and much more easily extendible software infrastructure while retaining OOF1's user-friendly Graphical User Interface (GUI). It differs from OOF1 in its object-oriented modular design, which makes use of the Python and C++ programming languages.

OOF2 currently solves linear elasticity, thermal conductivity, electric polarisation, and piezoelectricity problems in two dimensions, and can easily be extended to include any problem that can be expressed in the *schematic* form

$$\Psi_j = \sum_i K_{ij} \nabla \varphi_i$$

$$\nabla \bullet \Psi_j = f_j$$

where Ψ_j is a flux, φ_i is a field, K_{ij} is a modulus or coupling constant, and f_j is a generalised applied force. For example, to describe the mechanical equilibrium of a material subjected to an external load, the field is displacement, the modulus is the elastic stiffness tensor, the 'gradient' of the field is the strain, and the 'flux' is the stress. In general, the flux is a sum over contributions from many thermodynamic fields.

OOF2 material properties are specified in three-dimensional form, and after assigning them to an image they are reduced to two dimensions by specifying either that fields have no out-of-plane gradients (e.g., plane-strain) or that the fluxes have no out-of-plane components (e.g., plane-stress).

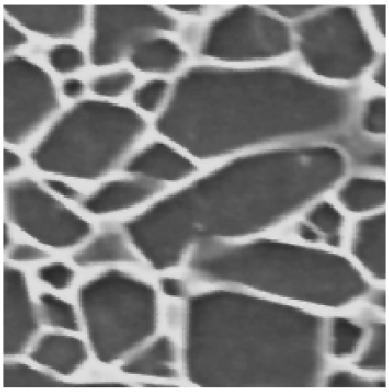
This paper briefly discusses some of the most important and fundamental features of OOF2. For more details, the reader is referred to the OOF2 manual, which may be found at the OOF website (http://www.ctcms.nist.gov/oof/), along with the source code and installation instructions. OOF2 is distributed at no charge. It will run on any Unix system, including Linux, Mac OS X, and, with modifications, on Windows with the Cygwin environment (http://www.cygwin.com/). Installation requires the presence of a C++ compiler, a Python interpreter (http://www.python.org/), the X11 window system,

and a small number of additional free, open-source libraries. What follows is a brief description of using `OOF2` to analyse a particular microstructure.

2 Using OOF2

`OOF2` starts with a micrograph, such as the one shown in Figure 1. While this particular micrograph happens to be of a silicon nitride ceramic, Si_3N_4, the topology of the images that `OOF2` can analyse is arbitrary, and is only limited by the resolution and quality of the input. By using simple-point-and-click operations, `OOF2` can assign any desired (real or hypothetical) single-crystal material properties to each of the components of the microstructure, and rapidly examine the effects of those properties on the local microstructural fields or the macrosopic response.

Figure 1 A Si_3N_4 microstructure, courtesy of Paul et al. (1998). The dark regions are a crystalline phase, and light regions are glassy amorphous interfacial regions



3 Microstructures

The fundamental abstract entity used by `OOF2` is a *`Microstructure`* (with a capital M), just as the fundamental object manipulated by a word processor is a document. A *`Microstructure`* is a rectangular array of pixels, where each pixel carries information about the local material properties. A *`Microstructure`* may (but does not have to) contain one or more *`Images`*. An *`Image`* (an experimental micrograph or the output of a simulation) is the usual starting point for an `OOF2` calculation, and `OOF2` can create a *`Microstructure`* object directly from an *`Image`* file. Most common image file formats are accepted. If different image processing techniques are needed to bring out different features of a (small m) microstructure, the (capital M) *`Microstructure`* may contain more than one *`Image`*, as long as they all have the same pixel size.

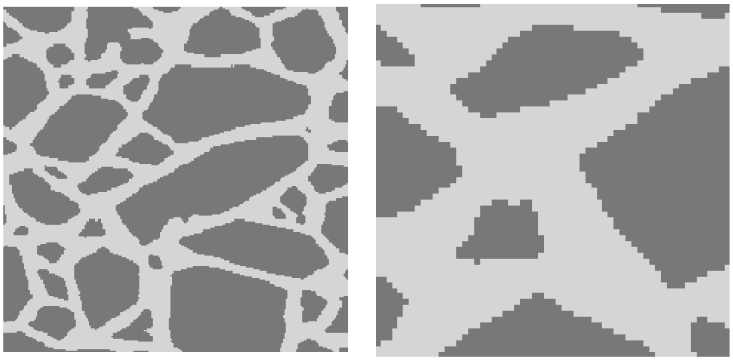
When an *`Image`* is loaded into `OOF2`, its physical width and height can be specified, in any type of units. `OOF2` contains no fixed system of units. The only requirement is that all numerical quantities must be entered using a consistent set of units. The width and height do *not* have to be commensurate with the *`Image`*'s size in pixels; that is, the pixels do not have to be physically square.

Microstructures can also be created without using an *Image*, by simply specifying their sizes both in physical units and in pixels.

4 Materials, pixel selections, and pixel groups

After creating a *Microstructure*, the next required task for the OOF2 user is to assign *Materials* to each pixel in the *Microstructure*. *Materials* are defined as sets of *Properties*. Most *Properties* are coefficients in physical constitutive relations, such as elastic modulus, thermal conductivity, piezoelectric coupling tensor, or mass density. Other *Properties*, such as *Orientation*, indirectly affect the constitutive equations, while others, such as *Colour*, are purely decorative. Figure 2 shows a possible 'material map' of the *Microstructure* from Figure 1, in which each pixel has been coloured by the *Colour Property* assigned to its *Material*.

Figure 2 A 'Material map' constructed from the *Microstructure* shown in Figure 1. The colour of the pixels indicates what *Material* has been assigned to them: (a) shows the entire *Microstructure* and (b) is a detail demonstrating unphysical jagged edges



(a) (b)

When assembling a *Material* from *Properties*, the *Properties* are chosen from a hierarchical list. The *Properties* can be parameterised, copied, named, and saved for future reference. New *Properties* can be added to the hierarchy. At the current time, the hierarchy includes mechanical properties (linear elasticity, stress-free strain, and force density), thermal properties (thermal conductivity and heat sources), electric properties (dielectric permittivity and space charge density), couplings (thermal expansion, piezoelectricity, and pyroelectricity), and the non-constitutive properties orientation and colour. All of the moduli and coupling constants have both isotropic and anisotropic variants, appropriate for any crystal symmetry. Although OOF2 only solves two dimensional problems, crystal symmetries and orientations have their full three-dimensional form.

The pixels to which the *Materials* are assigned correspond to microstructural constituents (grains, grain boundaries, precipitate phases, etc.). These sets of pixels are identified and grouped by the user using OOF2's pixel selection tools. This operation in general requires some judgement on the part of the user to physically interpret the image

data, although there may be special cases where fully automatic pixel grouping is possible.

The OOF2 pixel selection tools include both graphical (operated by mouse clicks on an *Image*) and non-graphical operations, and tools to create and modify named groups of pixels, as well as tools for modifying the underlying *Images*. The graphical pixel selection tools include methods for selecting all pixels within a given colour range of a clicked pixel, and various methods for choosing a contiguous set of pixels. Non-graphical selection tools include methods for expanding or contracting the currently selected set, or selecting all pixels within a given absolute colour range.

Selected pixels can be added to or subtracted from named pixel groups. Creating pixel groups is not required, but is a convenient way to recover a previous set of selected pixels. Furthermore, *Materials* can be assigned to a named pixel group, as well as to whichever pixels are currently selected. Figure 2 was constructed in one of the simplest possible ways. First, the *Image* was thresholded; i.e., each pixel was set to either black or white, depending on whether its original grey level was below or above a certain predetermined value. Then two pixel groups named 'grains' and 'matrix' were created. The black pixels were selected and added to the 'grains' group, and the white pixels were selected and added to the 'matrix' group. Two *Materials*, also called 'grains' and 'matrix' were created, and assigned to the pixels in the corresponding pixel groups. Finally, *Colour Properties* were added to the two *Materials*.

5 Skeletons

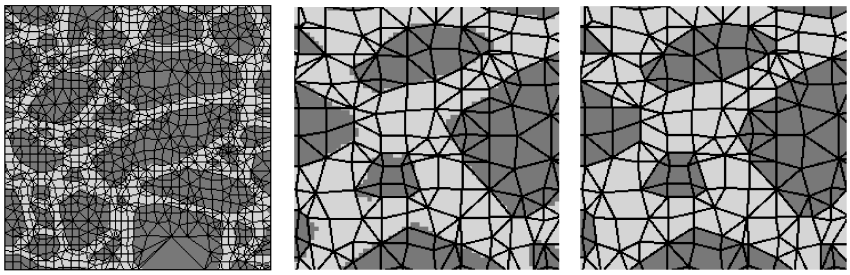
After creating a *Microstructure* and assigning *Materials* to its pixels, the next step in using OOF2 is to create a finite element mesh *Skeleton*. A *Skeleton* is not quite a complete finite element mesh – it specifies the positions and shapes of the elements, but contains no information about finite element interpolation functions, which fields are defined, or which equations are being solved. Like *Images* and *Materials*, *Skeletons* are components of an OOF2 *Microstructure* object. One *Microstructure* can be home to many *Skeletons*. After a *Skeleton* has been created, it can be adapted via node motion and element refinement so that it provides a good geometrical representation of its associated microstructure. A *Skeleton* is a good representation if each of its elements is *homogeneous*, meaning that its area consists of only one type of *Material* in the *Microstructure*. In the end, the *Material* assigned to each *Skeleton* element, whether or not the element is homogeneous, will be the *Material* assigned to the majority of pixels within the element (pixels at element edges can make fractional contributions to this calculation).

There are three ways to create a *Skeleton* from a *Microstructure* in OOF2. The simplest creates a single quadrilateral element or two triangular elements from each pixel. While this process is straightforward and requires no user intervention, in most cases it almost certainly will create an unnecessary number of elements, which will slow down further computations. Additionally, it will introduce internal jagged edges, which are unphysical and do not represent the topology of the analysed image. For example, the jagged edges of the grains in Figure 2(b) are just a result of the pixelisation of the original image, and it makes no sense for a mesh to follow the stairstep boundaries pixel by pixel.

A better way to create a *Skeleton* is to use OOF2's 'Auto' feature, which requires the user to specify the minimum and maximum sizes of the features of interest in the *Microstructure*. OOF2 will then create an initial *Skeleton* just fine enough to resolve the maximum sized features, and will locally refine and adapt the *Skeleton* so that it can resolve the minimum sized features. Such a *Skeleton* is shown in Figure 3. Note that most, but not all, of the elements have good shapes (containing no highly obtuse or acute angles, and having aspect ratios near 1.0) and that most, but not all, of the boundaries between *Materials* coincide with element edges.

Figure 3 (a) A *Skeleton* created from the *Microstructure* shown in Figure 2(a), using the Auto Skeleton feature. In this and the following figures, the jagged edges in the element boundaries are display artifacts; (b) of the *Skeleton*, corresponding to the region shown in Figure 2(b) and (c) with elements coloured according to their dominant *Material*

Author: Please check if Figure 3 and 6 captions are ok.

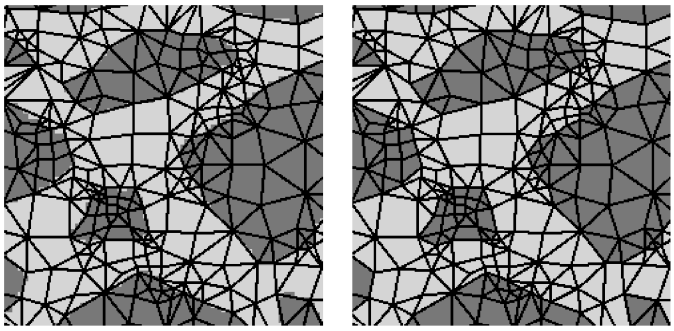


(a) (b) (c)

The third way to create a *Skeleton* is to do it by hand, first creating a regular array of large elements, and then using various *Skeleton* modification tools to adapt the *Skeleton* to the *Microstructure*. It is also possible to use the same tools to improve a *Skeleton* that was initially created automatically. The tools include ways of refining, or subdividing, inhomogeneous elements, various methods for moving nodes to improve element homogeneity or shape, and methods for merging elements and removing extremely badly shaped elements. It is possible to select a portion of the *Microstructure* and to apply the tools only to elements in that area, or even to select an individual *Skeleton* node and specify its desired position exactly. Figure 4 shows the result of using some of these tools on a portion of the *Skeleton* shown in Figure 3. The new *Skeleton* follows the *Material* boundaries better, but in places comes perilously close to resolving the individual pixels.

When creating a *Skeleton*, it is important to remember that the original micrograph is an approximation to the actual microstructure, the material map is an approximation to the micrograph, and the *Skeleton* is an approximation to the material map. Furthermore, the finite element solution that will be obtained is yet another approximation. In this context, there is nothing to be gained by over refining a *Skeleton*.

Figure 4 A detail (a) of the `Skeleton` from Figure 3, with additional refinement and (b) the same detail, with elements coloured according to their dominant `Material`



(a) (b)

6 Meshes, fields, equations, boundary conditions and solutions

Once a satisfactory representation is obtained, an actual finite element `Mesh` can be created from a `Skeleton`. The `Mesh` adds physics and math to the `Skeleton` geometry to create a fully functional finite element mesh. Many `Meshes` may be made from a single `Skeleton`. In each `Mesh`, one `Mesh` element is created for each `Skeleton` element, and one `Mesh` node is created for each `Skeleton` vertex. Additional nodes are created on `Mesh` edges and within elements if required by the finite element interpolation order specified during `Mesh` construction. `OOF2` currently supports three- and six-noded triangular elements and four-, eight-, and nine-noded quadrilateral elements.

After creating a `Mesh`, the next step is to define and activate `Fields`. In `OOF2`, `Fields` are the spatially distributed quantities which determine the local physical state of a system. `Fields` in this sense are distinct from the usual fields (with a lowercase *f*). For example, stress is a field in the sense that it has values everywhere in the continuum, but it is not a `Field`, because in `OOF2` it is a computed secondary characteristic and not a fundamental quantity.

`OOF2` currently contains displacement, temperature, and voltage `Fields`. A `Field` is said to be *defined* on a `Mesh` if the values of its components are stored in the `Mesh` nodes. A defined `Field` can be evaluated at any point within a `Mesh` by using the finite element interpolation functions. `Field` values can be initialised to fixed values, to Python functions of x and y, or by copying from another `Mesh` (and interpolating if necessary). `Fields` are said to be *active* if their values are to be obtained from the finite element solution. Finally, `Fields` are *in-plane* if they are constrained to have no out-of-plane derivatives. This generalises the concept of plane-strain.

`OOF2` can solve two classes of `Equations`: divergence equations and plane-flux (generalised plane-stress) equations. Divergence equations specify that the divergence of some flux equals the applied forces. The built-in divergence equations are the heat equation (Fick's second law), the force balance (mechanical equilibrium) equation, and Coulomb's equation in its differential form. The exact form of the equations depends on the `Properties` defined in the `Microstructure`, and on which `Fields` are defined and active. The moduli and body forces at any point will be zero unless the local

Material contains the corresponding *Properties*. For example, an elastic and thermal problem may have both temperature and displacement *Fields*, and elasticity and thermal-conductivity properties. This is by itself a well-posed problem, but there will be no contribution to the stress from changes in temperature unless a thermal-expansion property, with associated modulus, is also included in the local *Material*.

Plane-flux equations specify that the out-of-plane component of a *Flux* (stress, heat flux, etc.) is zero. For mechanical problems, plane stress is often imposed by constructing effective in-plane moduli which do not contribute to out-of-plane stresses, and then solving an effective two-dimensional finite element problem (Zienkiewicz and Taylor, 2000). OOF2 does not do this, because the form of the effective moduli depends upon which couplings are present, and these are not known until run-time. Instead, OOF2 solves plane-flux equations along with the divergence equations within the finite-element scheme, which achieves the same effect in a much more general context. This means that the out-of-plane *Fluxes* will in general not be pointwise identically zero, but they will tend towards zero as the size of the *Mesh* elements goes to zero and the finite element approximation improves.

OOF2 supports five kinds of boundary conditions: Dirichlet, Neumann, floating, generalised force, and periodic. It also supports two kinds of boundaries: point and edge. Point boundaries are named collections of element nodes, and edge boundaries are named collections of ordered contiguous element edges. Boundaries are defined in *Skeletons* and implicitly defined in *Meshes* derived from the *Skeletons*. Each *Skeleton* has eight predefined boundaries: edge boundaries at the top, left, bottom, and right edges, and point boundaries each consisting of a single node at each of the four corners of the *Microstructure*. New boundaries can be defined, and need not lie along the actual external edges of the *Microstructure*.

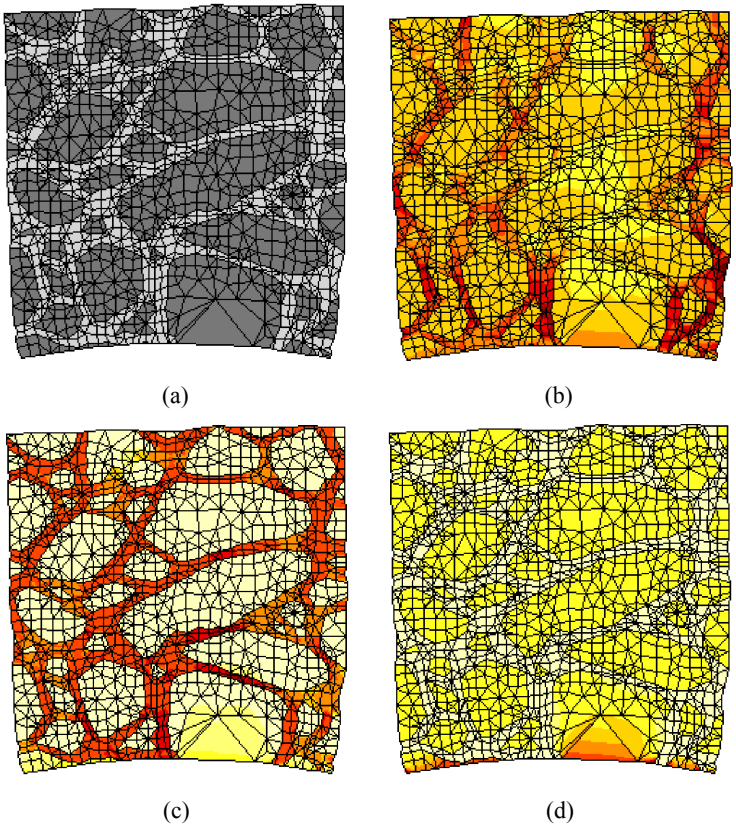
Dirichlet boundary conditions specify the values of *Field* components at boundaries. Neumann conditions specify the normal components of *Fluxes*. Floating boundary conditions specify *Field* values, but only up to an unspecified offset, which will be determined as part of the solution. Generalised force conditions specify the divergence of a *Flux* at each node of a point boundary. Periodic boundary conditions, as expected, enforce the equality of *Field* values at paired boundaries. They can only be applied to *Meshes* constructed from *Skeletons* that were declared to be periodic when they were built.

All of the boundary conditions, except periodic, can be specified in terms of an arbitrary 'profile' function, which determines how the *Field* or *Flux* value varies along the boundary. Profiles can be specified in terms of arbitrary Python functions of position, absolute arc length, or fractional arc length along the boundary.

Figure 5 shows the Si_3N_4 microstructure from Figure 1 after assigning fictional material parameters and equilibrating. The dark grains are isotropically elastic with Young's modulus $E = 10$ and Poisson's ratio $\nu = 0.3$, and have an isotropic thermal expansion coefficient alpha = 1.0. (The units are arbitrary, as mentioned above. In these units, the size of the whole undistorted microstructure is 200×200.) The matrix between the grains is softer than the grains, with $E = 0.1$ and $\nu = 0.45$, and has no thermal expansion. Dirichlet boundary conditions were applied to the bottom boundary, setting the x component of displacement to zero and constraining y to follow a parabolic profile. All other boundaries were free. The temperature was defined and initialised to 0.1 everywhere, and the displacement was computed. OOF2 solved an 8966×8966

matrix equation, which converged to a relative tolerance of 10^{-13} in 2191 iterations of a conjugate gradient solver. The results are shown in Figure 5.

Figure 5 (a) A `Microstructure` deformed by Dirichlet boundary conditions on the bottom edge, and thermal expansion of the grains; (b) the *xx* component of the total strain. Values range from black (–0.423, tension) through red and yellow to white (0.368, compression); (c) the trace of the geometric strain. Black = –0.239 and white = 0.31 and (d) the trace of the stress tensor. Black = –5.51 and white = 0.31



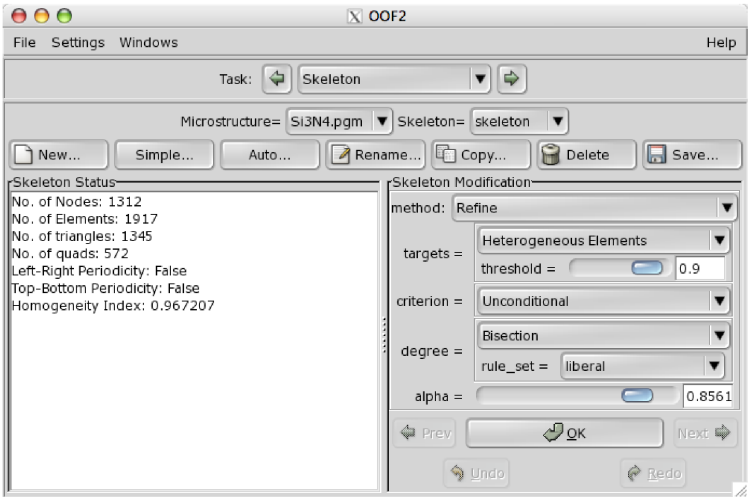
7 Output and analysis

`OOF2` contains a number of methods for examining the results of a virtual experiment. Figure 5 displays a few of the ways of visualising an equilibrated `Mesh`. `OOF2` can plot the components and invariants of all `Fields` and `Fluxes`, as well as energy densities and strains, using either original, actual deformed, or enhanced coordinates. (Enhanced coordinates exaggerate displacements.) It can plot cross sections along arbitrary straight lines drawn on the `Microstructure`, and can compute averages and integrals over `Microstructures`, cross sections, subregions, and boundaries of a `Mesh`. This allows it to be used to compute effective macroscopic moduli.

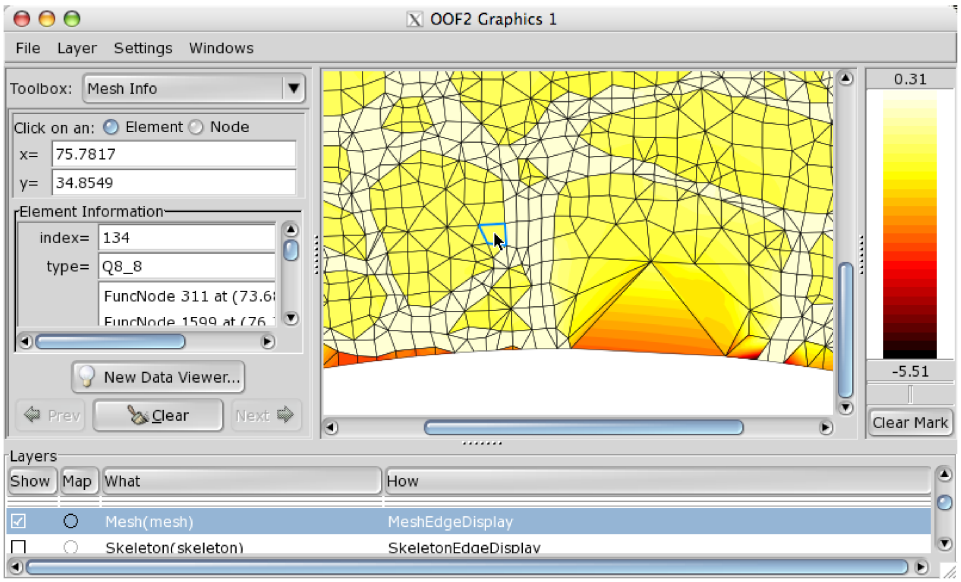
8 GUI and scripting

OOF2 can be driven either by a GUI, or via scripts. Figure 6 illustrates some features of the user interface.

Figure 6 (a) The main OOF2 control window. The window has many separate pages, for performing the various required tasks. The page shown creates and modifies *Skeletons* and (b) the OOF2 graphics window, which displays microstructures and handles interactive mouse-based input. Different tasks are handled by different toolboxes on the left hand side of the window. The toolbox shown is showing the attributes of the Mesh element under the mouse in the display area



(a)



(b)

Every action in the GUI that has a consequential outcome is recorded as a text command, which can be saved in a log file. The file can be read back into the program to repeat a calculation or part of one. Log files are actually Python scripts, so they can be edited to include arbitrary Python variables, functions, and control statements. For example, a script can be saved for doing a calculation on one image, and then modified to run the calculation on a whole series of images, or with a range of material parameters. Scripts can be loaded either with or without the GUI running, allowing background batch computation.

9 OOF2 extensions

Many of the features of `OOF2` are designed to be easily extendible by end users. It's possible to add new *`Fields`*, *`Fluxes`*, or *`Equations`* with a few lines of code. New *`Material Properties`*, output functions, and finite element types can be added with a bit more effort. The details, including sample code, are given in the on-line `OOF2` manual. Extensions can either be compiled into the main `OOF2` code base, or built as separate modules that can be loaded at run-time if and when they are needed.

10 Current and future development

At the current time, `OOF2` development is concentrated on adding time dependent and non-linear properties, such as plasticity. `OOF2` will soon be able to directly importorientation imaging microscopy (OIM) data. Finally, `OOF2` is being extended to work on three dimensional micrographs. The long-term goal is to make `OOF` a general purpose platform for the computation of physical properties of two and three dimensional microstructures.

Acknowledgements

The authors would like to thank Seung-Ill Haan for his contributions to the `OOF2` code. Rhonald Lua is supported by the Penn State University MatCASE project, which is funded by NSF ITR research grant DMR-0205232.

Disclaimer

Certain commercial equipment, instruments, or materials are identified in this paper to specify adequately the experimental procedures and conditions. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

References

Langer, S.A., Fuller Jr., E.R. and Carter, W.C. (2001) 'OOF: an image-based finite-element analysis of material microstructures', *Computing in Science and Engineering*, Vol. 3, No. 3, p.15.

Paul, F.B., Sun, E.Y., Plucknett, K.P., Alexander, K.B., Hsueh, C-H., Lin, H-T., Waters, S.B., Westmoreland, C.G., Kang, E-S., Hirao, K. and Brito, M.E. (1998) *Journal of the American Ceramic Society*, Vol. 81, No. 11, pp.2821–2830. AUTHOR PLEASE SUPPLY ARTICLE TITLE.

Zienkiewicz, O.C. and Taylor, R.L. (2000) *The Finite Element Method*, 5th ed., Vol. 1, Butterworth Hienemann, Oxford, p.91.

Websites

Cygwin may be found at http://www.cygwin.com/

Python may be found at http://www.python.org/

The `OOF` website is located at http://www.ctcms.nist.gov/oof/