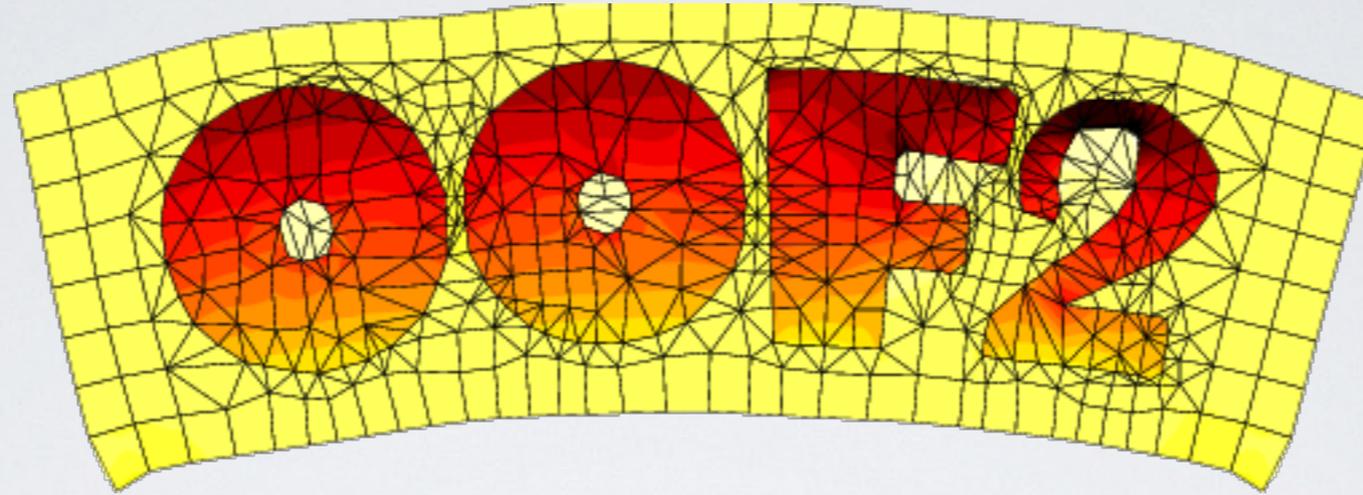


# MODULE 3: FINITE ELEMENT METHOD

Practice: OOF2

# **I. What is OOF2?**

# OOF2



OOF: Finite Element Analysis of Microstructures

**NIST** National Institute of Standards and Technology • U.S. Department of Commerce

<http://www.ctcms.nist.gov/oof/oof2/>

# OOF2

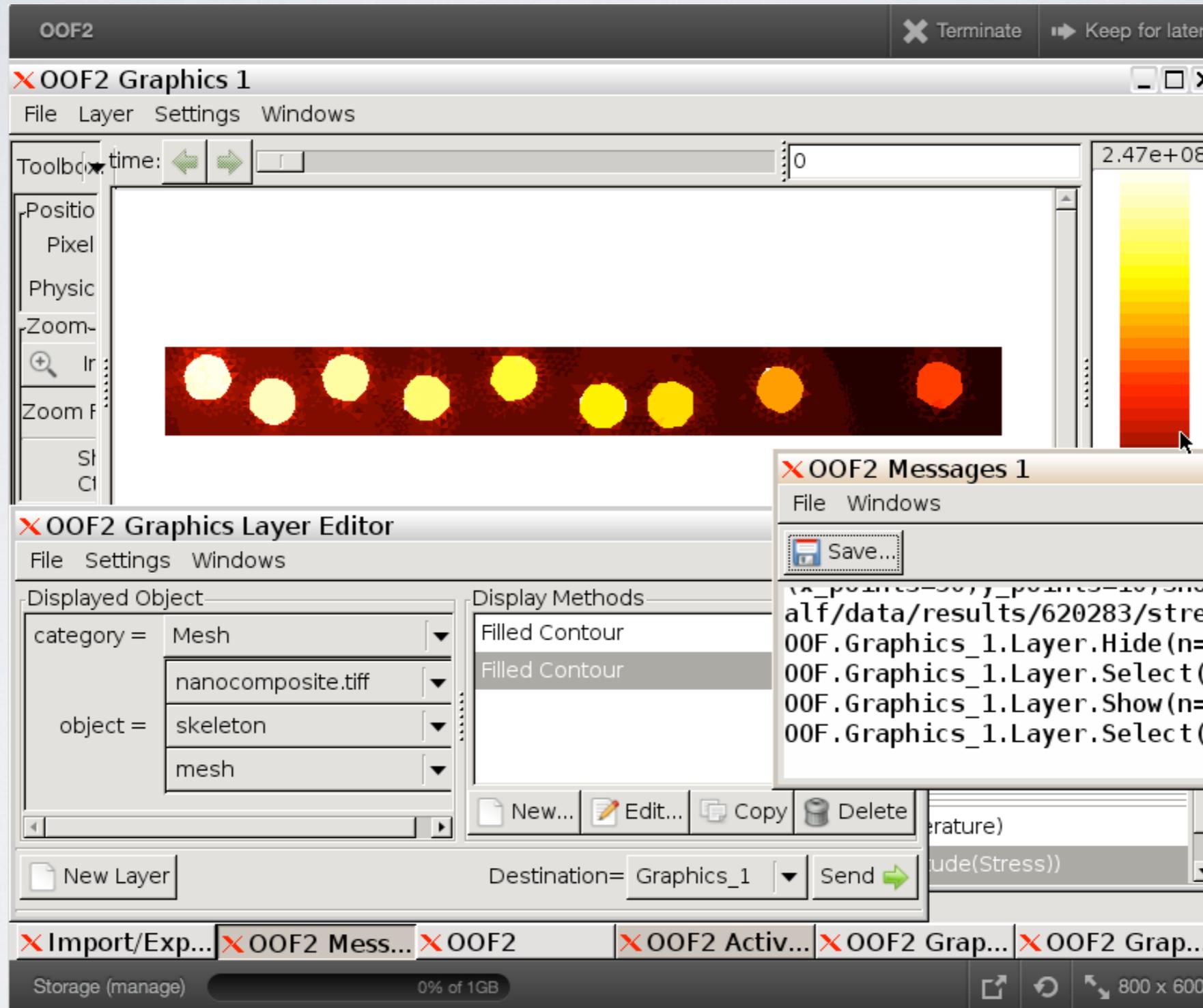
- Object Oriented Finite Element Analysis Tool
- Originally developed in the late 90's at NIST by Craig Carter, Ed Fuller, Andy Roosen, and Steve Langer
- Fundamentally a **two-dimensional** FEM tool

**OOF2** is public domain software created at the [National Institute of Standards and Technology \(NIST\)](#) to investigate the properties of microstructures. The microstructure of a material is the (usually) complex ensemble of polycrystalline grains, second phases, cracks, pores, and other features occurring on length scales large compared to atomic sizes.

At the simplest level, **OOF2** is designed to answer questions like, "I know what this material looks like and what it's made of, but I wonder what would happen if I pull on it in different ways?", or "I have a picture of this stuff and I know that different parts expand more than others as the temperature increases -- I wonder where the stresses are greatest?"

# OOF2

- Very user-friendly GUI



- CLI also available

\$\$\$

- **FREE** to the end user!



# Useability

- Very well documented

<http://www.ctcms.nist.gov/~langer/oof2man/>



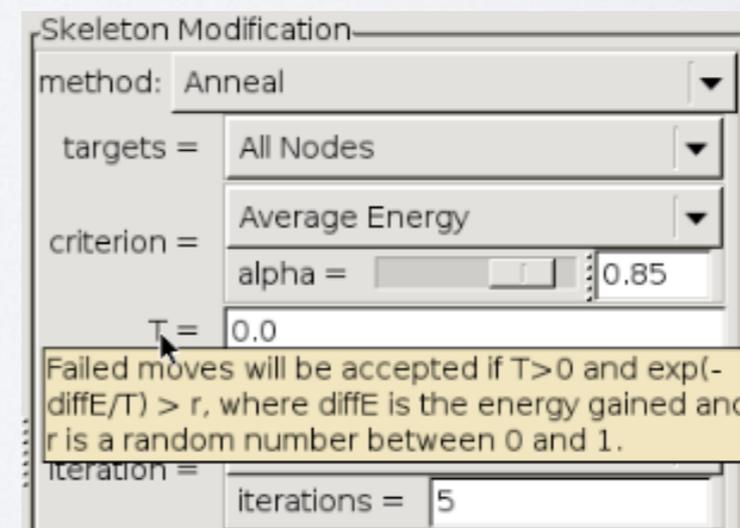
- Many tutorials and guides available

<http://www.ctcms.nist.gov/~rlua>

[www.ctcms.nist.gov/oof/talks/workshop06/SteveLanger/example.pdf](http://www.ctcms.nist.gov/oof/talks/workshop06/SteveLanger/example.pdf)

<http://nanohub.org/resources/4732>

- Rollover help within the GUI



# Availability

- Free download of source code from

<http://www.ctcms.nist.gov/oof/oof2/#download>

- Supported by Linux and Mac OS X, but many pre-reqs

<http://www.ctcms.nist.gov/oof/oof2/prerequisites.html>

Python (2.4 through 2.7)

<http://www.python.org>

Magick++

<http://www.imagemagick.org/www/Magick++/index.html>

gtk+-2.0 (2.6 or later)

<http://www.gtk.org/download/>

libgnomecanvas2

<http://directory.fsf.org/graphics/misc/libgnomecanvas.html>

pygtk2 (2.6 or later)

<http://www.pygtk.org>

swig 1.1 build 883

<http://www.swig.org/download.html>

- Local installation can be tricky

- EWS:** `module load OOF2`

# Availability

But...

## OOF: Finite Element Analysis of Microstructures

### OOF2

- OOF Home
  - Mailing List
  - Newsletters
  - Credits and Disclaimer
- OOF1
- OOF2
  - Download
  - OOF2 Manual
  - OOF2 Reference Manual
  - Bugs
  - Run OOF2 online**
- gtklogger
- OOF2 Working Group
  - Steve Langer
  - Andrew Reid
  - Günay Doğan
  - R. Edwin García
- NIST Links
  - Applied and Computational Mathematics Division
  - Information Technology Laboratory
  - Center for Theoretical and Computational Materials Science

[\[Introduction\]](#) [\[Features\]](#)  
[\[System Requirements\]](#) [\[Disclaimer and Copyright\]](#) [\[Download\]](#) [\[Installation\]](#) [\[Platform Specific Installation Notes\]](#)  
[\[Getting Started\]](#) [\[Manual\]](#) [\[FAQ\]](#)  
[\[Reporting Bugs\]](#) [\[Known Problems\]](#)

**Introduction** OOF2 version 2.1.7 is now **available**. The major differences between 2.1 and 2.0 are that 2.1 can solve time dependent problems, and has much improved nonlinear solvers. A detailed discussion of the differences and a summary of how to use the new features is included in the [What's New in 2.1](#) page.

OOF2 version 2.1.5 can now be run on the [nanoHUB](#). See the [announcement](#) for more details.

OOF2 retains (almost) all of the features of OOF1, although it does not read OOF1 data files. The latest versions of OOF1, however, can write OOF2 data files.

OOF2 is based on a new set of C++ classes for finite elements and material properties, tied together in a Python infrastructure. [Python](#) is an easy to use, high-level, object-oriented scripting language.

OOF2 is in an [Active Development](#) status.

# Availability

Online webtool available at nanoHUB

<https://nanohub.org/tools/oof2/>

The screenshot shows the nanoHUB.org website interface. At the top, there is a navigation bar with links for Home, Resources, Members, Explore, nanoHUB-U, Partners, About, and Support. A search bar is located in the top right corner. The main content area features the OOF2 tool page, which includes a 'Launch Tool' button, version information (Version 2.1.5-64 - published on 18 Jun 2012), and an abstract. The abstract describes OOF2 as public domain finite element analysis software created at the National Institute of Standards and Technology (NIST) to investigate the properties of microstructures. It also includes a 'Credits' section and a 'SEE ALSO' section with recommendations.

**OOF2**  
By Stephen Langer<sup>1</sup>, R. Edwin Garcia<sup>2</sup>, Andrew Reid<sup>1</sup>  
1. NIST 2. Purdue University  
Object oriented finite element analysis tool

**Launch Tool**  
Version 2.1.5-64 - published on 18 Jun 2012  
doi:10.4231/D37W67515 cite this  
Open source: license | download  
View All Supporting Documents

**10.0 RANKING**  
● Expert  
919 users, detailed usage  
302 users in 26 classes  
2 Citation(s)  
19 questions (Ask a question)  
2 review(s) (Review this)  
1 wish(es) (Add a new wish)  
Add to your favorites!

**SEE ALSO**  
No results found.

**RECOMMENDATIONS**  
OOF2 Tutorial  
OOF2 64 bit  
Meep  
1D Finite Different Method Conduction Heat Transfer Tool  
OOF1  
Floods/Floops  
Quasicontinuum Method  
NEMO5 Tutorials

**Category**  
Tools

**Published on**  
18 Jun 2012

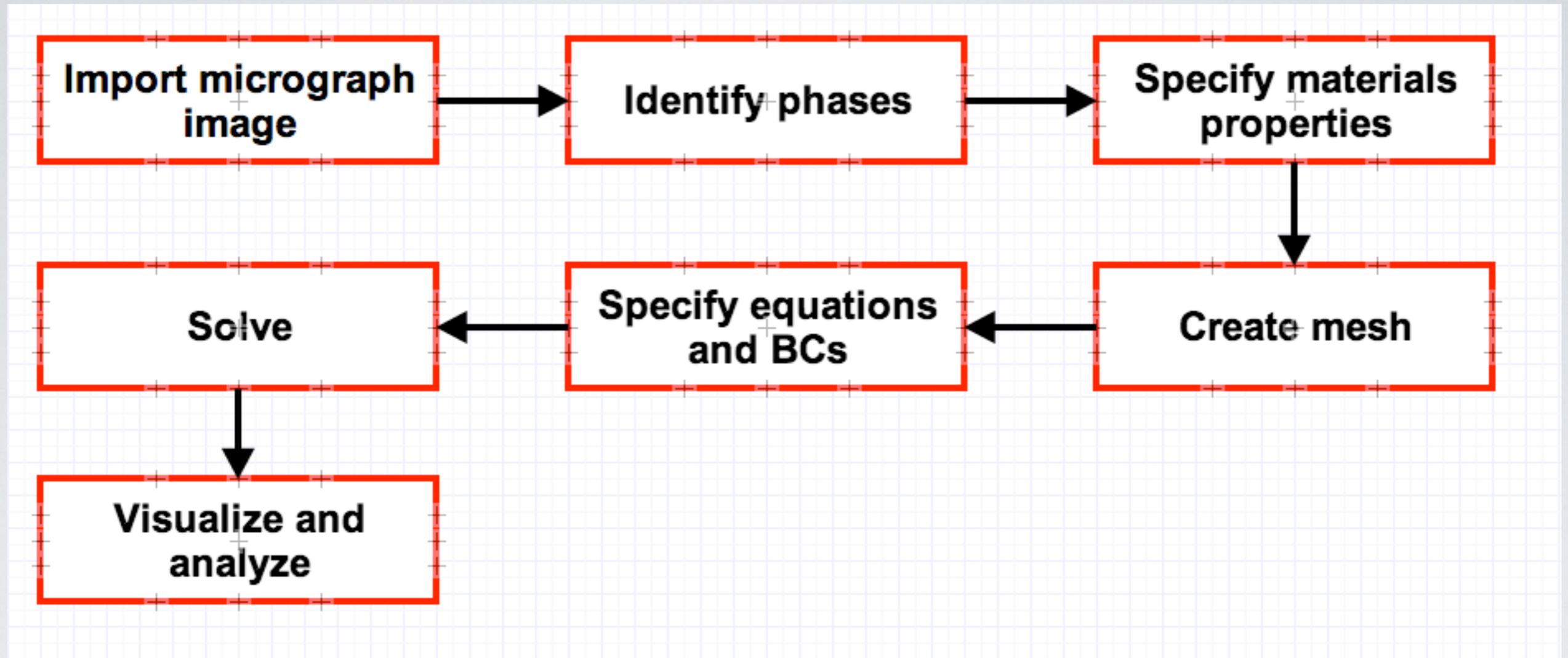
**Abstract**  
OOF2 is public domain finite element analysis software created at the National Institute of Standards and Technology (NIST) to investigate the properties of microstructures. At the simplest level, OOF2 is designed to understand the effects of far fields (boundary conditions) on the local microstructural fields, or to assess the mechanical, electrical, and thermal reliability of a material with a complex topology. OOF2 allows the user to study the thermal, electrical, and stress fields in a microstructure, along with couplings such as piezoelectricity, pyroelectricity, and thermal expansion. OOF2 can also perform crystallographic analyses of polycrystalline materials by using tensor form material properties. The inputs necessary to perform a simulation include: 1) a microstructure (real micrograph or computer generated), 2) material properties and 3) boundary conditions. The specified information enables OOF2 to simulate the multiphysical properties, thus allowing to analyze and engineer the effect of microstructure. The figures below is a lamellar directionally solidified eutectic of NiO and yttria-stabilized ZrO<sub>2</sub> which was used as the input microstructure. Next a zoomed in view of the mesh is shown and finally the stresses developed in the microstructure due to an increase in temperature are displayed. \*Images from: S.A. Langer, E.R. Fuller, Jr. and W.C. Carter, OOF: an image-based finite-element analysis of material microstructures. Comput Sci Eng 3 3 (2001), pp. 15-23.

**Credits**

Requires a **free** nanoHUB account for access  
(Please register for nanoHUB, Facebook/Google sign-ins can be buggy.)

## **II. OOF2 basics**

# Workflow



- All tasks performed in seamlessly and in sequence using OOF2 integrated GUI
- Windows based environment

# Units

- **OOF2 has no units!**

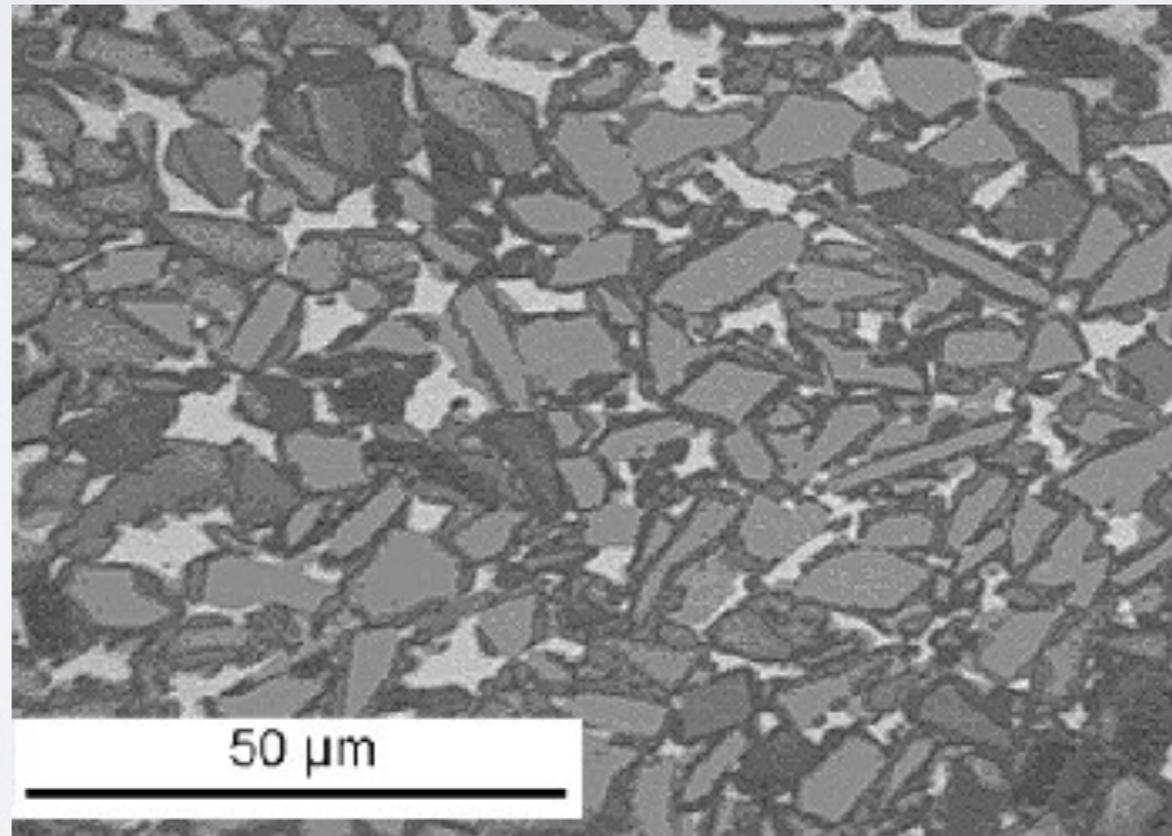


Use most convenient units  
No conversions required

No error/consistency checking  
GIGO  
One error multiplies...

# Microstructures

- Fundamental ethos of OOF2 is to process and simulate FEM systems based on experimental microstructures
- Establish microstructure in OOF2 by **image upload**

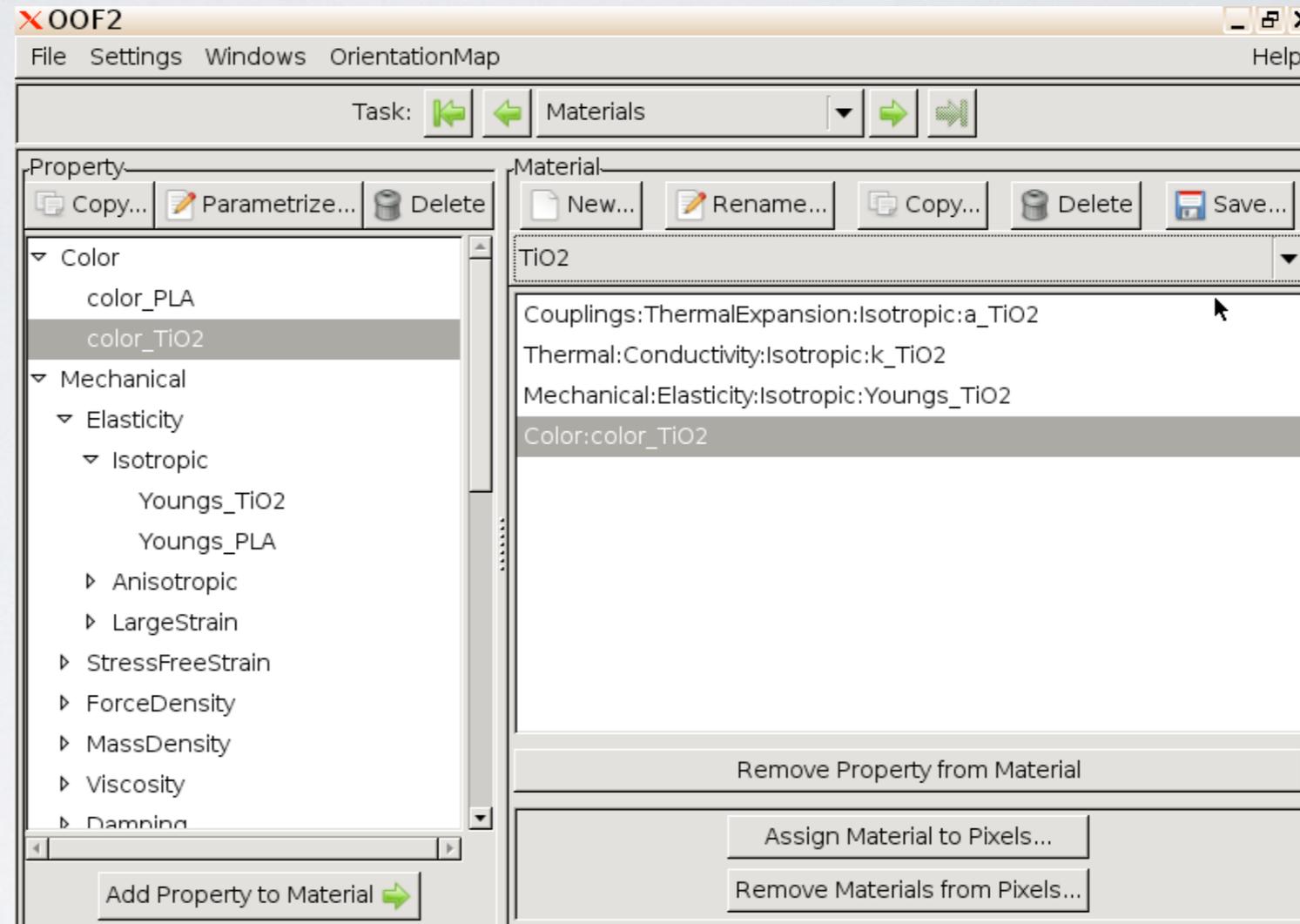


silicon carbide micrograph

- Different phases / materials identified by groups of pixels

# Materials

- Materials with defined properties created and assigned to pixel groups

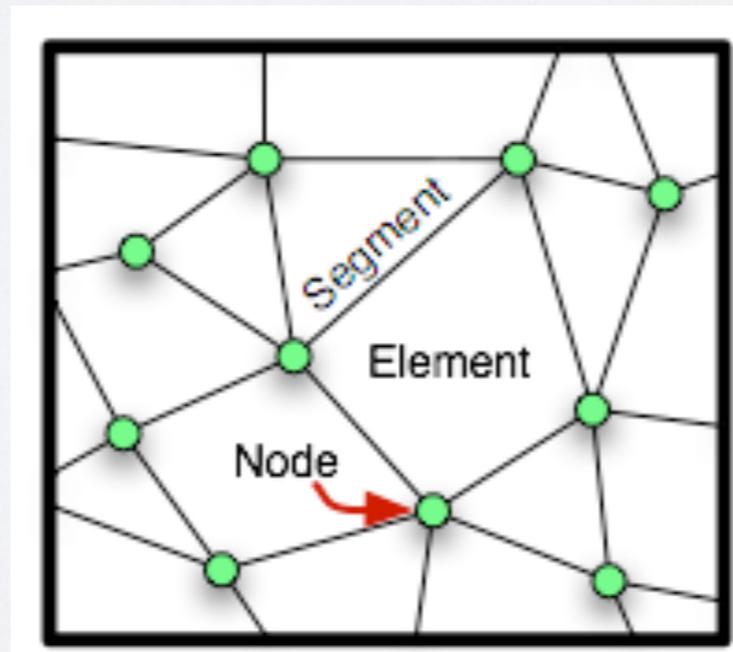


- Isotropic or anisotropic properties supported

Materials properties: Young's modulus Poisson's ratio thermal conductivity coefficient of thermal expansion dielectric permittivity viscosity density color ...

# Skeletons

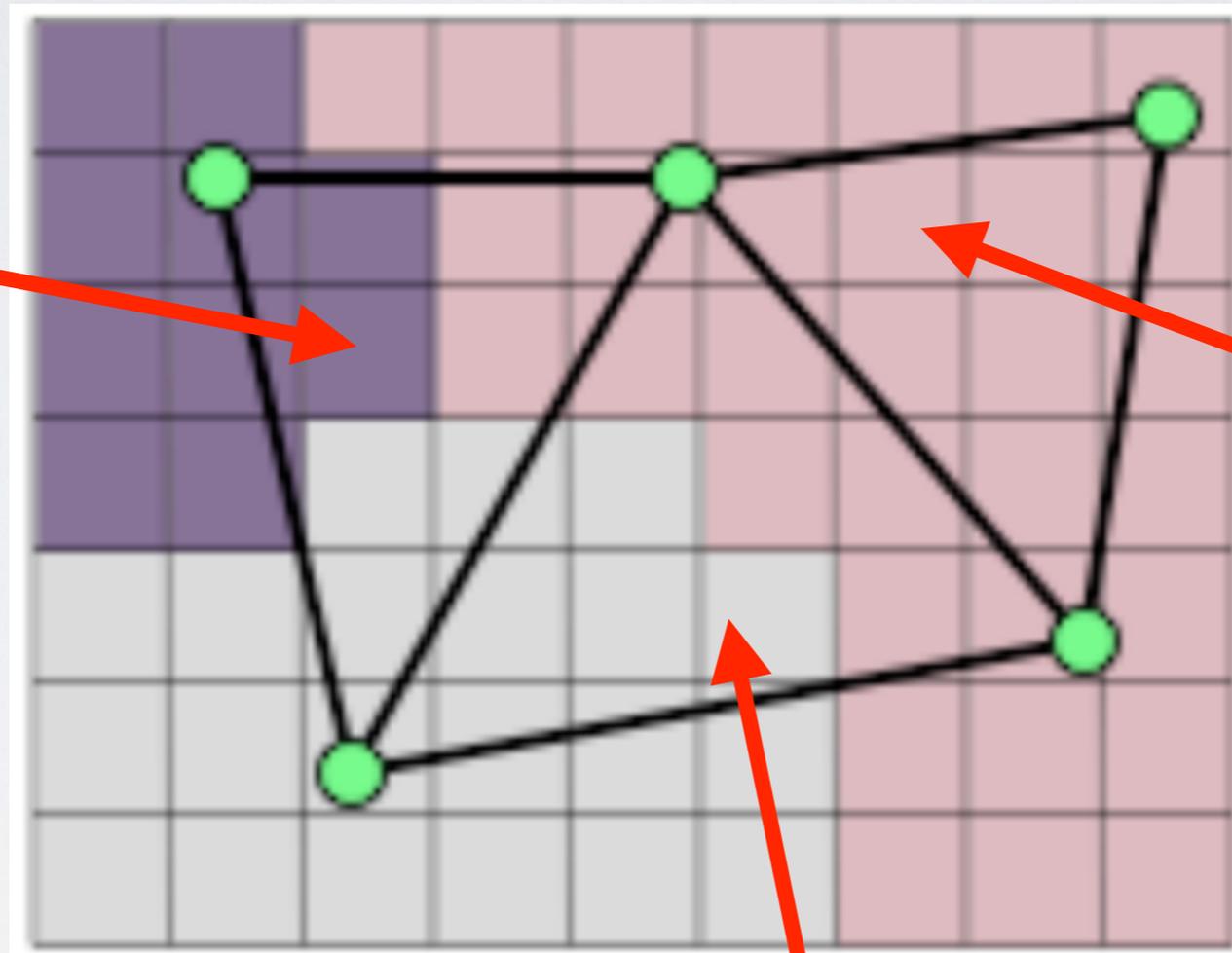
- A **skeleton** is a partitioning of the micrograph into finite elements, it contains **only geometric information**
- A **mesh** is derived from the skeleton, containing **geometry + equations, fields and boundary conditions**
- Modifying the boundary conditions, material properties, or equations to be solved requires rebuilding of the mesh but not the skeleton



# Element homogeneity

- A good skeleton is highly **homogeneous** (homogeneity index  $\sim 0.99$ ) - each element contains a single phase

heterogeneous  
(HI  $\sim 0.3$ )



homogeneous  
(HI = 1.0)

heterogeneous  
(HI  $\sim 0.5$ )

# Skeleton refinement

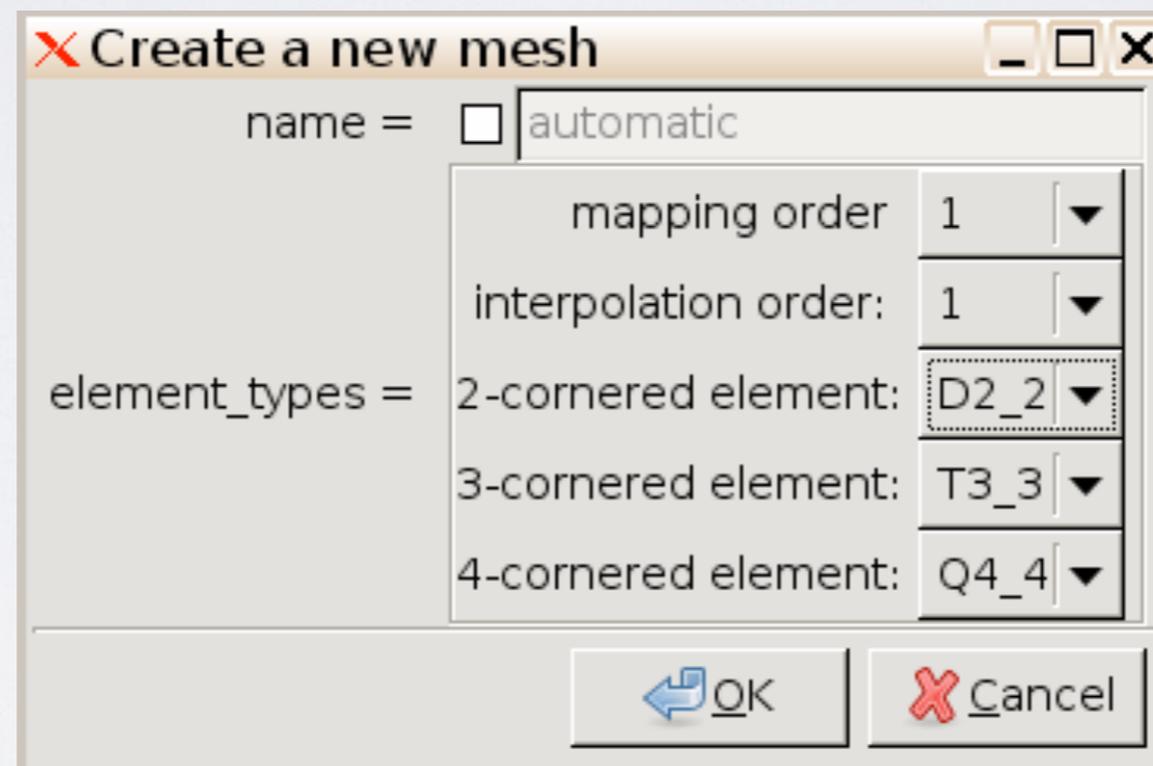
- A good skeleton also contains very few high-aspect ratio elements, yielding better FEM numerical solutions
- A number of skeleton refinement tools exist to minimize the **effective energy**

$$E = \alpha E_{homog} + (1 - \alpha) E_{shape}$$

- annealing
- edge swapping
- smoothing
- refinement (element partitioning)

# Meshing

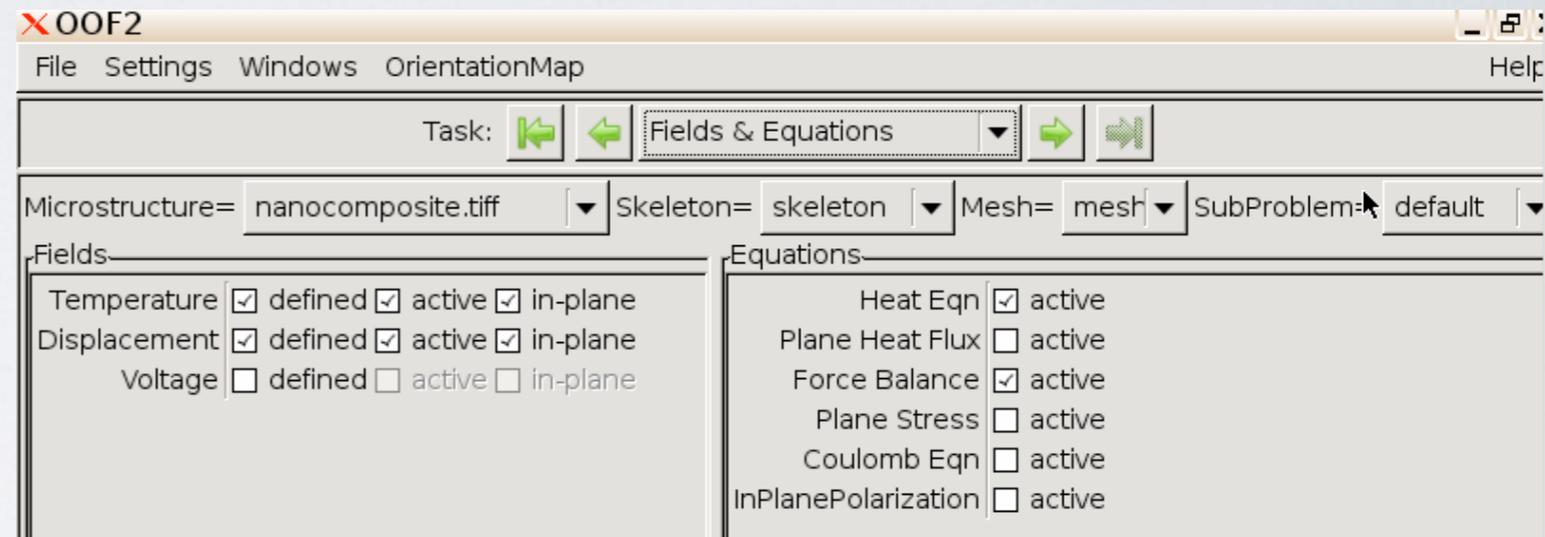
- The **mesh** is constructed from the **skeleton**
- The mesh elements in OOF2 support **linear** and **quadratic** interpolation functions



- These functions interpolate values from the mesh nodes to the element interiors

# Equations

- OOF2 supports solution of the following equations:
  - heat equation
  - force balance
  - Coulomb equation
  - plane stress
  - in-plane polarization
  - plane heat flux



- Complex boundary conditions supported (Dirichlet, Neumann, periodic, generalized force)
- Complex fields possible (T, displacement, voltage)

# Solvers

- A number of advanced numerical solvers are available
- User-specified tolerance and maximum iterations
- Problems are typically **large** and **sparse**

direct: slow  
high accuracy  
unsuitable for large problems (hi RAM reqmts)

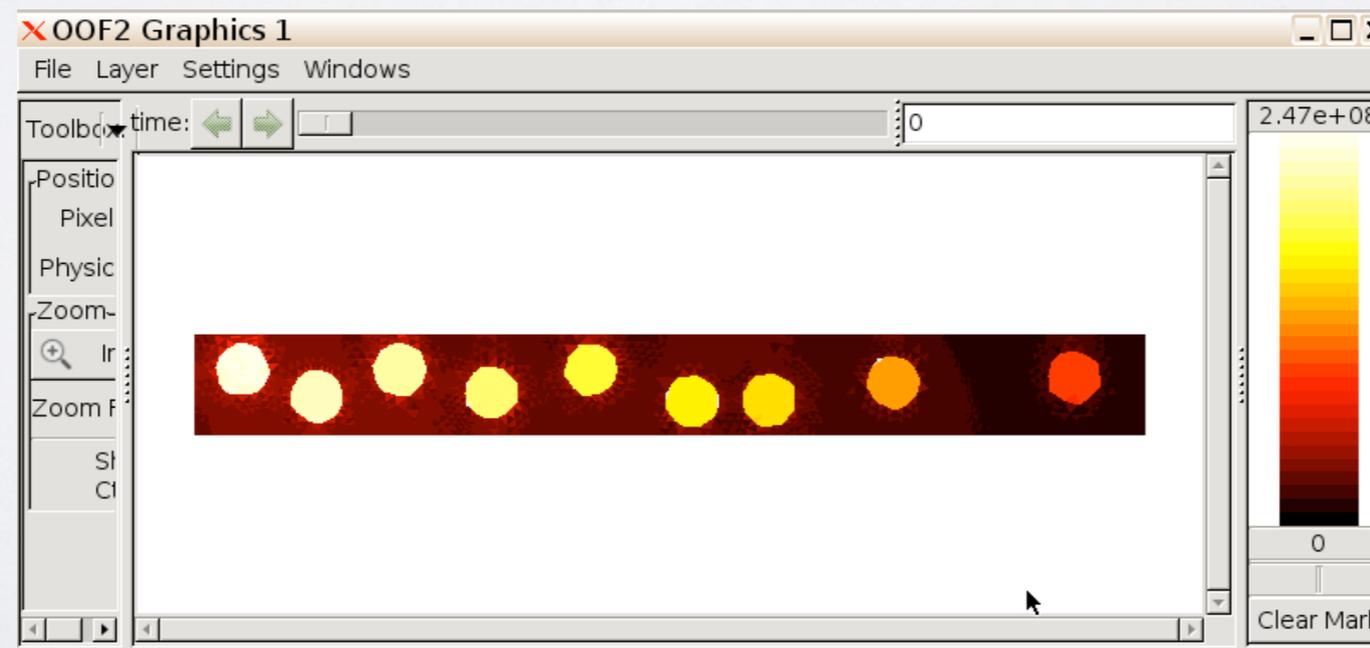
iterative: fast  
lower accuracy  
only choice for large problems

# Visualization

- Micrograph, skeleton, and mesh visualized in Graphics pane

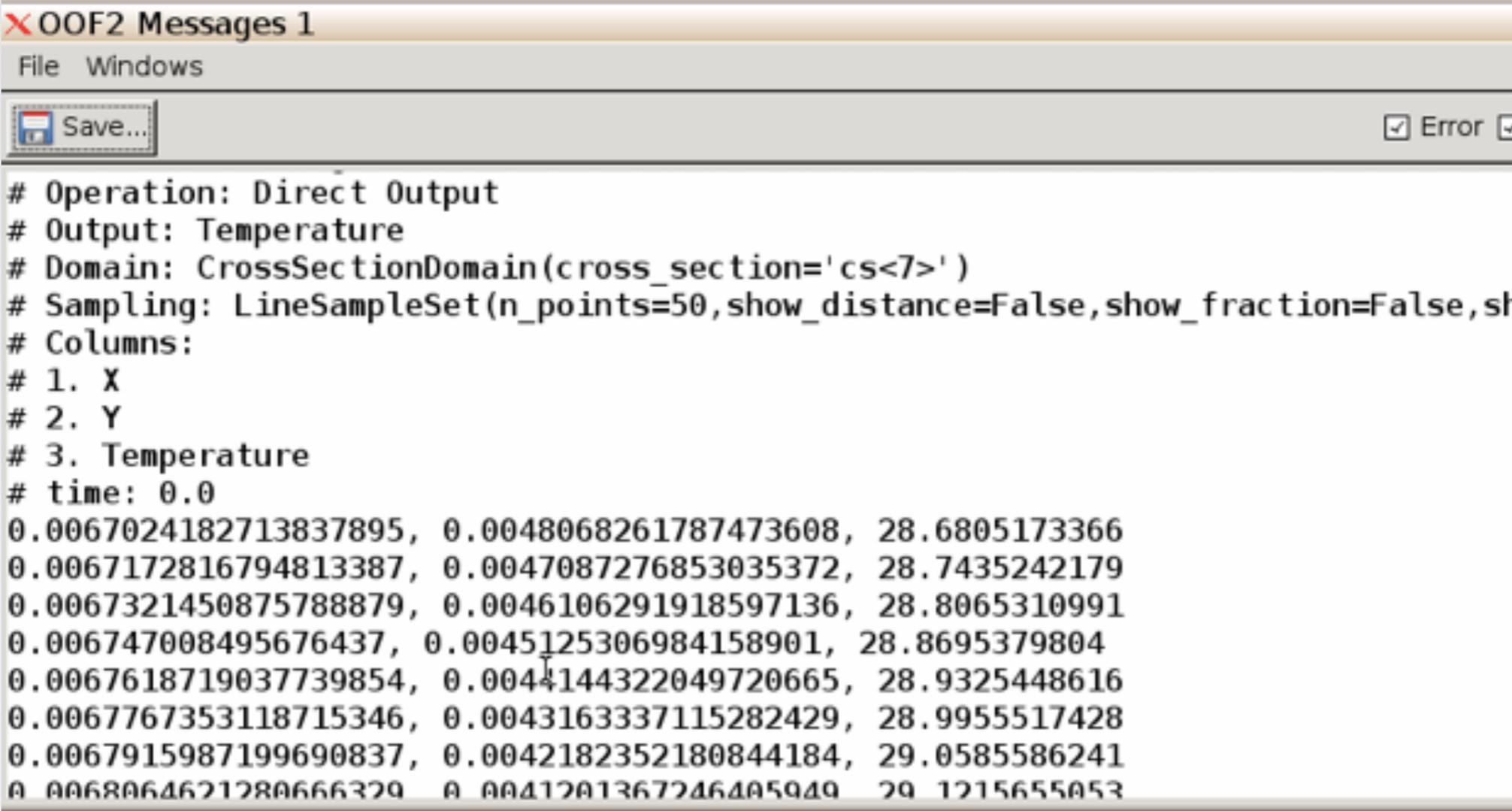


- Also used to visualize solutions (stresses, temperature) computed over the terminal mesh



# Analysis

- The solution of scalar and vector fields over the terminal mesh can be outputted using the Analysis pane



The screenshot shows a window titled "OOF2 Messages 1" with a menu bar containing "File" and "Windows". Below the menu bar is a toolbar with a "Save..." button and a checked "Error" checkbox. The main area contains the following text:

```
# Operation: Direct Output
# Output: Temperature
# Domain: CrossSectionDomain(cross_section='cs<7>')
# Sampling: LineSampleSet(n_points=50,show_distance=False,show_fraction=False,sh
# Columns:
# 1. X
# 2. Y
# 3. Temperature
# time: 0.0
0.0067024182713837895, 0.0048068261787473608, 28.6805173366
0.0067172816794813387, 0.0047087276853035372, 28.7435242179
0.0067321450875788879, 0.0046106291918597136, 28.8065310991
0.006747008495676437, 0.0045125306984158901, 28.8695379804
0.0067618719037739854, 0.0044144322049720665, 28.9325448616
0.0067767353118715346, 0.0043163337115282429, 28.9955517428
0.0067915987199690837, 0.0042182352180844184, 29.0585586241
0.0068064621280666320, 0.0041201367246405040, 29.1215655053
```

- Data can be dumped to file for offline analysis

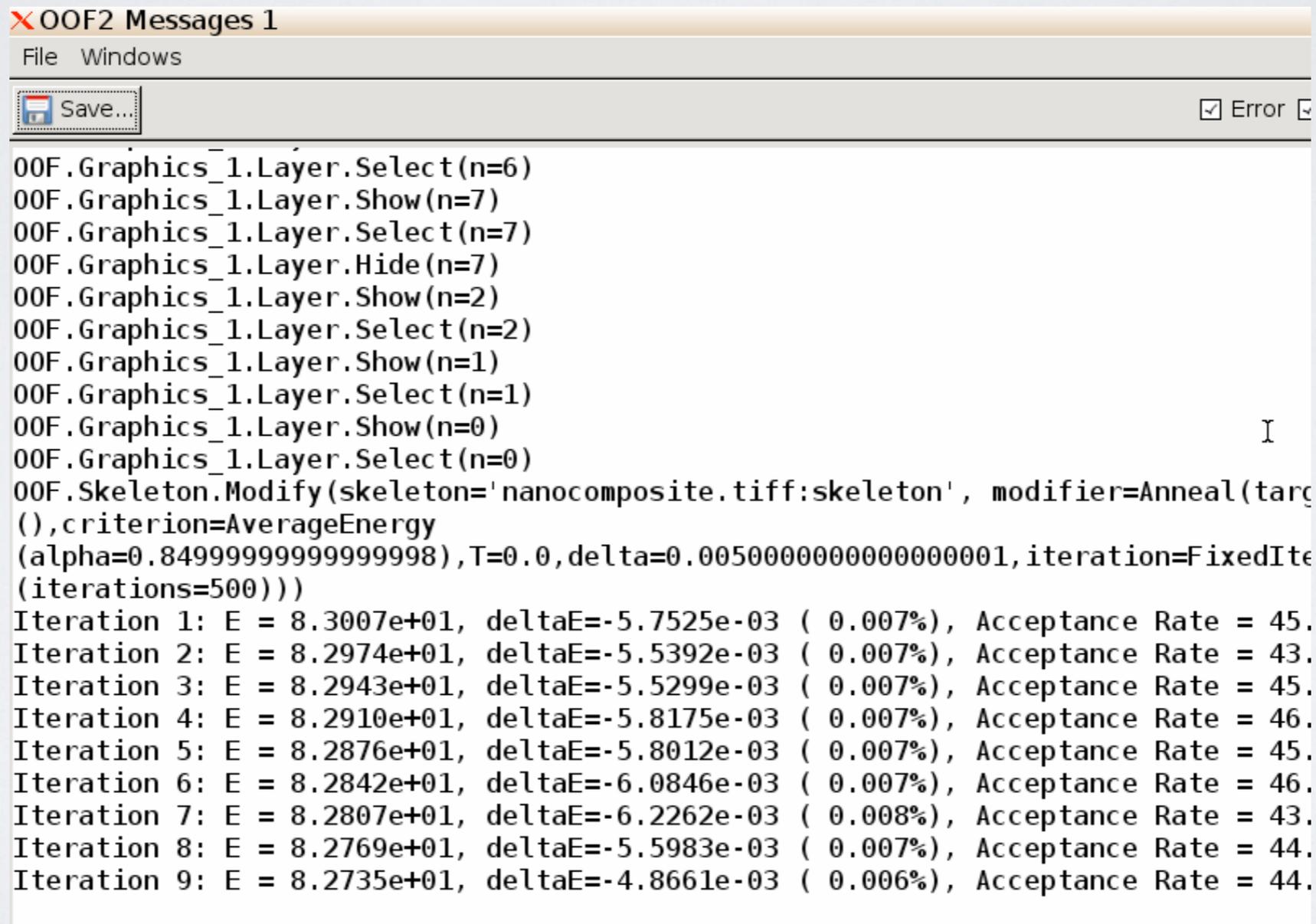
# Activity viewer

- The Activity window indicates the current tasks occupying OOF2
- Very useful for monitoring long processes (solving, skeleton refinement, visualization)



# Messages

- The Messages window provides detailed task information, and communications to the user



```
X OOF2 Messages 1
File Windows
Save... [Error]
00F.Graphics_1.Layer.Select(n=6)
00F.Graphics_1.Layer.Show(n=7)
00F.Graphics_1.Layer.Select(n=7)
00F.Graphics_1.Layer.Hide(n=7)
00F.Graphics_1.Layer.Show(n=2)
00F.Graphics_1.Layer.Select(n=2)
00F.Graphics_1.Layer.Show(n=1)
00F.Graphics_1.Layer.Select(n=1)
00F.Graphics_1.Layer.Show(n=0)
00F.Graphics_1.Layer.Select(n=0)
00F.Skeleton.Modify(skeleton='nanocomposite.tiff:skeleton', modifier=Anneal(target=(), criterion=AverageEnergy(alpha=0.84999999999999998), T=0.0, delta=0.005000000000000000001, iteration=FixedIteration(iterations=500)))
Iteration 1: E = 8.3007e+01, deltaE=-5.7525e-03 ( 0.007%), Acceptance Rate = 45.
Iteration 2: E = 8.2974e+01, deltaE=-5.5392e-03 ( 0.007%), Acceptance Rate = 43.
Iteration 3: E = 8.2943e+01, deltaE=-5.5299e-03 ( 0.007%), Acceptance Rate = 45.
Iteration 4: E = 8.2910e+01, deltaE=-5.8175e-03 ( 0.007%), Acceptance Rate = 46.
Iteration 5: E = 8.2876e+01, deltaE=-5.8012e-03 ( 0.007%), Acceptance Rate = 45.
Iteration 6: E = 8.2842e+01, deltaE=-6.0846e-03 ( 0.007%), Acceptance Rate = 46.
Iteration 7: E = 8.2807e+01, deltaE=-6.2262e-03 ( 0.008%), Acceptance Rate = 43.
Iteration 8: E = 8.2769e+01, deltaE=-5.5983e-03 ( 0.007%), Acceptance Rate = 44.
Iteration 9: E = 8.2735e+01, deltaE=-4.8661e-03 ( 0.006%), Acceptance Rate = 44.
```