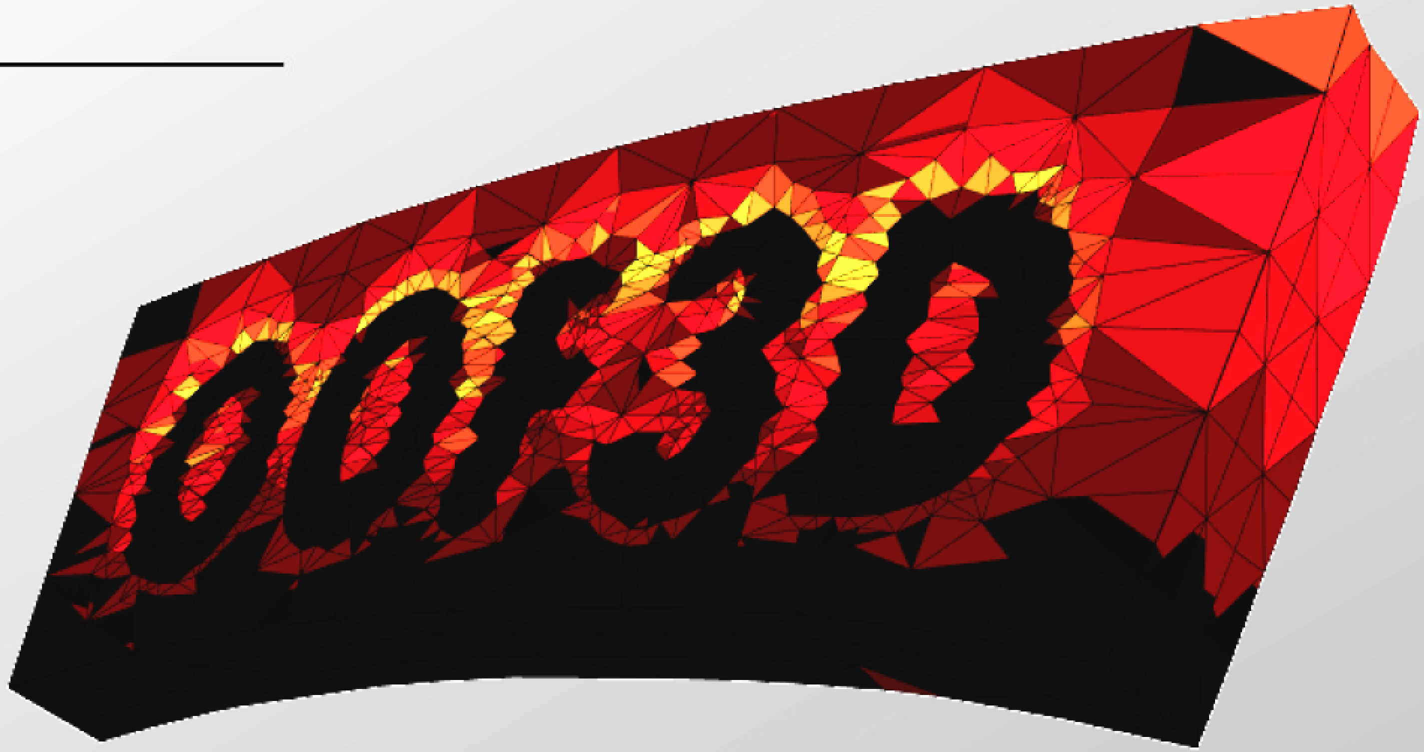


The OOF Finite Element Tool Finite For Materials Science

Andrew Reid, NIST

SciPy 2017

July 12, 2017

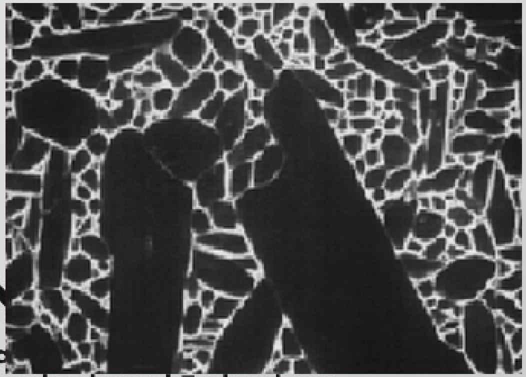
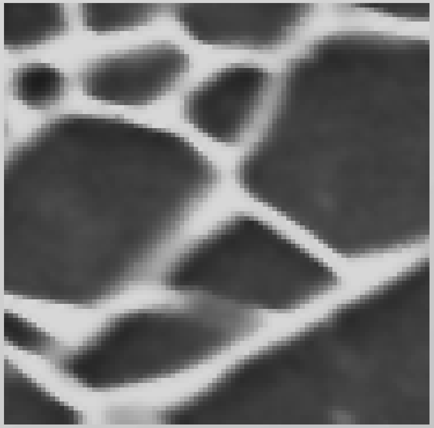
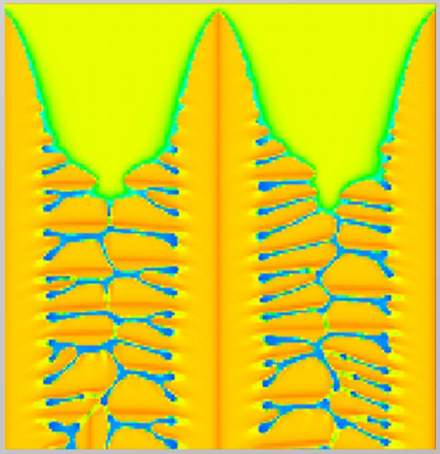
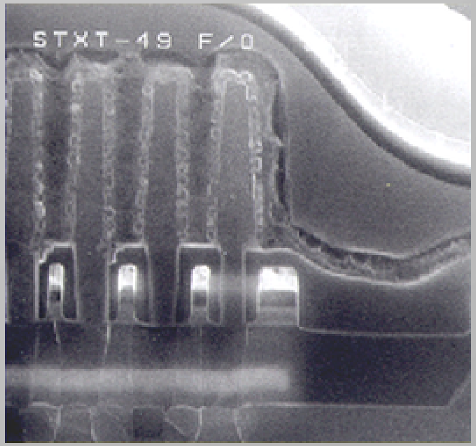
NIST
National Institute of Standards and Technology
U.S. Department of Commerce

FEM on Microstructures

Many years ago, an attempt was made to do FEM modeling of material microstructures.

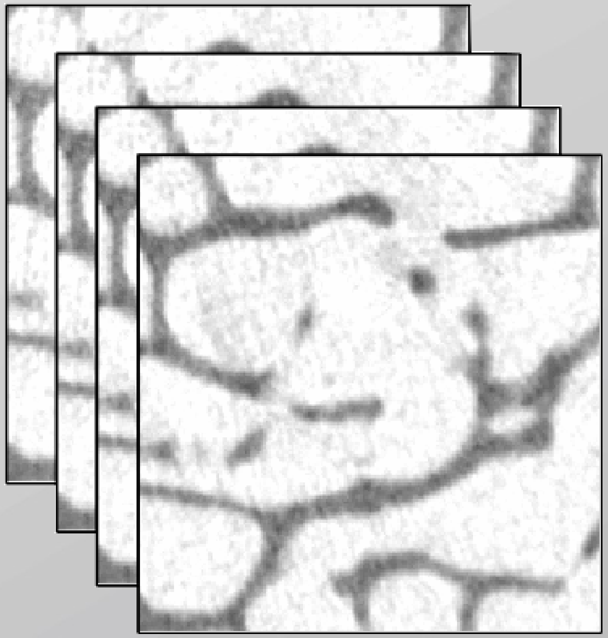
It proved to be hard, available tools were poorly adapted.

Work began on a materials-focused FEM tool, which evolved to fill this empty niche in the tool space, including image segmentation tools, meshing of microstructures, and extensibility to custom constitutive models.

Scope: Materials Science

- Meso-scale samples (microns)
- Interdisciplinary (chemistry, physics, more)
- Encapsulates math for Materials Science users
- Focused on structure-property relations
  - For a given structure, how do constituent properties control aggregate properties?
  - For given constituents, what structures give good properties?



National Institute of Standards and Technology
U.S. Department of Commerce

The Math of Materials Science

For some flux, σ dependent on a field ϕ :

$$M \cdot \ddot{\phi} + C \cdot \dot{\phi} + \nabla \cdot \sigma + f = 0$$

$$\sigma = \sum_i \gamma_i \nabla \dot{\phi} + \sum_i k_i \nabla \phi$$

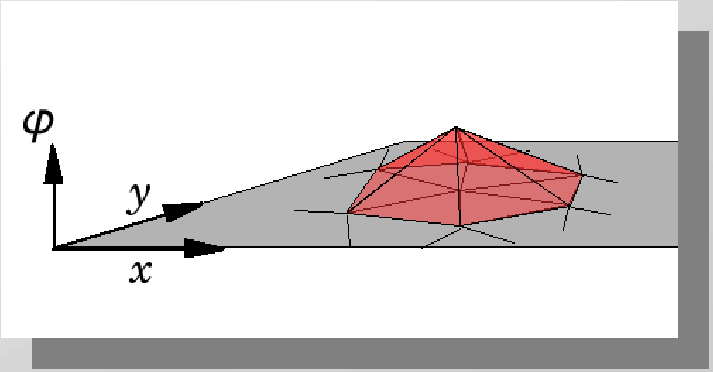
Constitutive rule – Materials Science domain experts know this relation

Typical fields: Displacement, temperature, concentration
Typical fluxes: Stress, heat flux, chemical flux

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

From Math to FE

$$\tilde{\phi} = \sum_i \phi_i N_i(\vec{x})$$



Continuum equilibrium equation: $\nabla \cdot \sigma + f = 0$

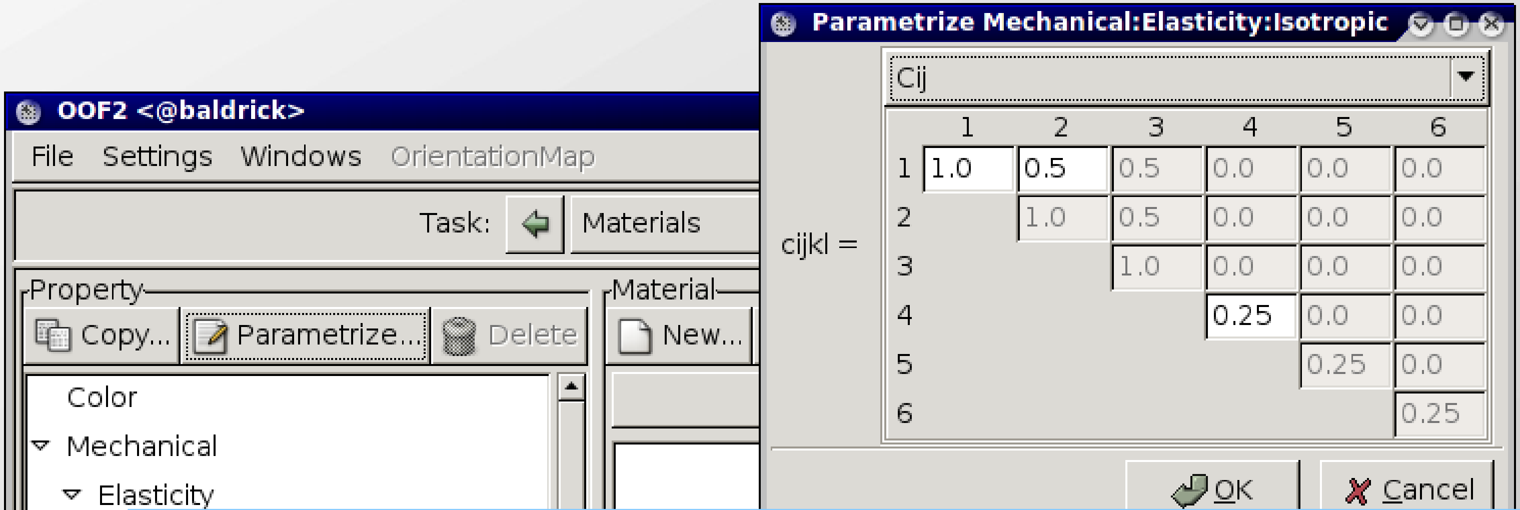
...becomes, in "weak form"

$$\sum_i \int N_j(\vec{x}) \cdot L[N_i(\vec{x})] d\vec{x} \cdot \phi_i + \int N_j(\vec{x}) f(\vec{x}) d\vec{x} = 0$$

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

Use this to put together a highly general, extensible solver, with arbitrary couplings from unknown constitutive rules

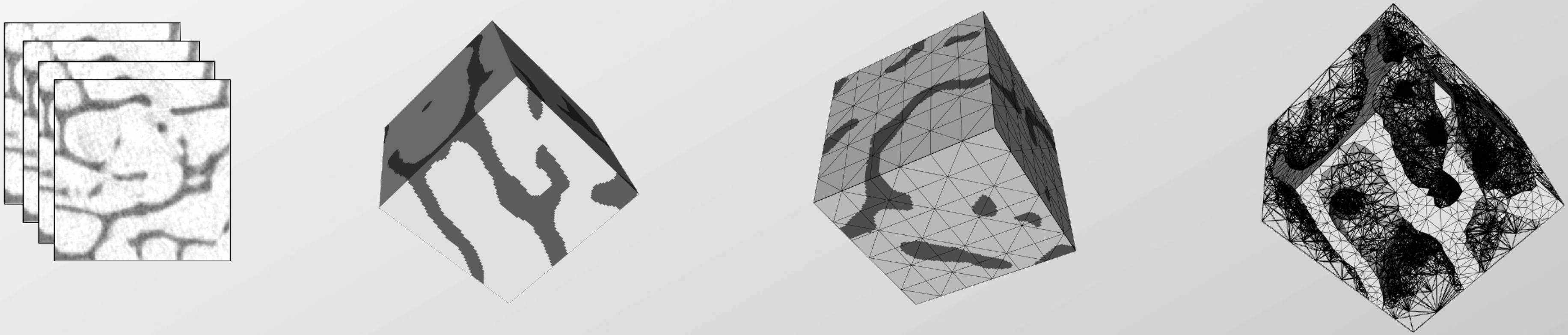
Constitutive Rules



- Numerous built-in properties (elasticity, etc.)
  - Includes couplings (thermal expansion, etc.)
  - Standard materials science nomenclature
- Ability to save, share properties
- Extension framework, stable API
- Open-source for the ultimate in extensibility
  - Python conspicuously useful here

The OOF user is an expert in the constitutive rules, but may not want to write code

Workflow: Image, Segments, Skeleton, Mesh

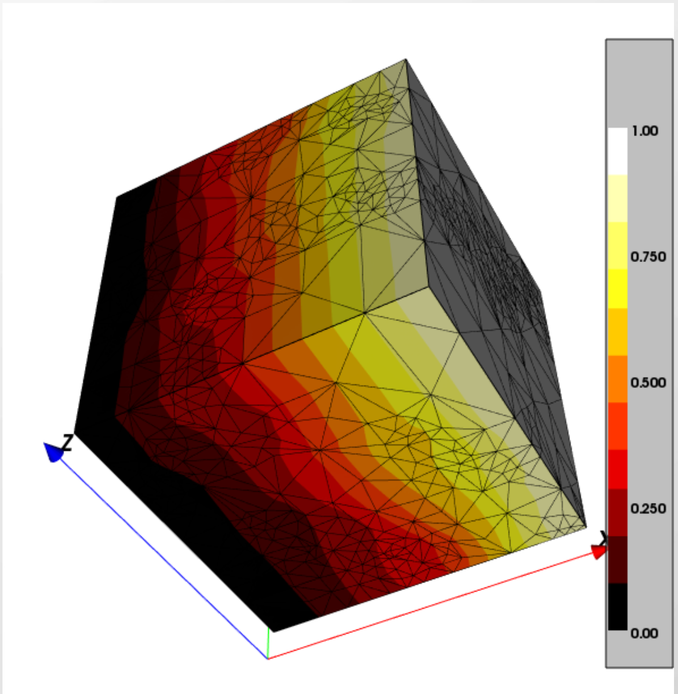


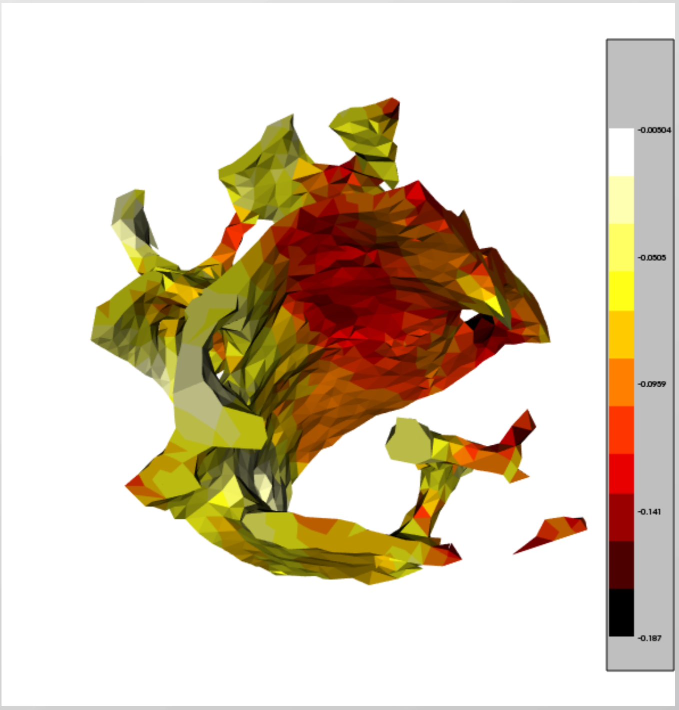
- Stack images into a 3D voxel set
- Modify the image – blur, despeckle, edge-detect
- Segment – selection tools (burn, brush) identify constituent phases
- Overlay a regular, space-filling grid
- Manipulate the mesh to match the segmentation boundaries
    - Bisection, node movement, simulated annealing
    - Tools preserve sanity, space-filling features of the mesh
    - Energy function (homogeneity, shape) measures quality

The OOF user is an expert in interpreting the results, and can rapidly assess correctness of the steps.

National Institute of Standards and Technology
U.S. Department of Commerce

Workflow: Analysis of Results





Solve the system – nonlinear, time-dependent, sparse solvers

Local graphics window provides interactive assessment of data
PDF export capability, publication-quality
Statistical tools – average, standard deviation, min/max
Integration of fluxes over boundaries
Extensible – new output methods easy to add

Direct data export capability for more sophisticated analysis
OOF developers don't attempt to anticipate all analyses

NIS
National
Standard
U.S. Dep

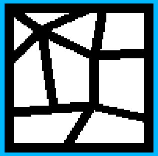
Conceptual Framework

Material – collection of properties

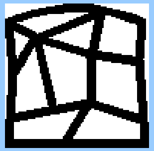
Image – maps pixels to colors

Microstructure -- “document” object

Skeleton – FE geometry

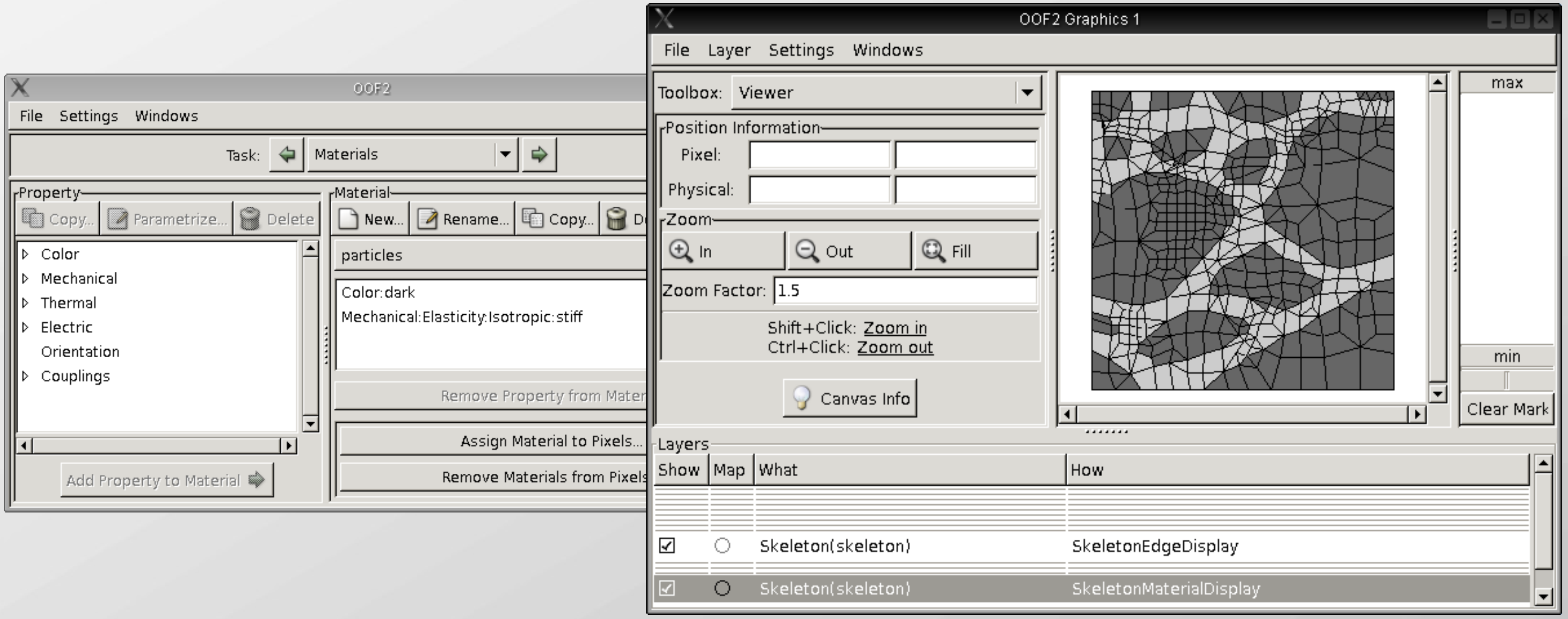


Mesh – physics of the problem



Using OOF

GUI mode or Menu-based command mode




```
# oof2 --text
>>> OOF.File.Load.Script("my_lifes_work.oof")
>>> ...
```

Architecture

- Written in a combination of Python and C++

Python and C++:
- Free, multiplatform
- Object-oriented
- Mostly feature-stable

Python:
- Flexible
- Dynamic (“duck typing”)
- Many available libraries

SWIG

C++:
- Many standard tools
- Fast executables
- Even more libraries

Architecture

Runs on Unix-like systems (Mac OS/X, Linux)

Dependent packages/libraries:

- SparseLib++ (customized, provided)
- Distutils (included with Python)
- SWIG (local fork, provided, build-time only)
- PyGTK
  - GTK2
- GnomeCanvas
- ImageMagick
- Blas/lapack
- VTK (for 3D)
- Docbook (to build the manual)
- OpenMP (for parallelization)

Not required to run current release

National Institute of Standards and Technology
U.S. Department of Commerce

Architecture

SWIG Advantage:
We have the ability to move the Python/C++ language boundary up or down the object stack in response to changes in performance requirements, maintainability, or other changes.

Seriously OO:
Throughout the development process, careful attention was paid to making the object structure reflect the problem structure, for comprehensibility, Python/C++ barrier-crossing, future maintainability, and future expansion.

Automation

A number of users have successfully used OOF for parametric studies – build a mesh in the GUI, write a command-line script to iterate over a parameter of interest, and perform multiple virtual experiments.

The menu system makes it easy to manipulate OOF objects in Python.

(e.g. K. Hazeli, C. El Mir, S. Papanikolaou, M. Delbo, KT Ramesh, “The Origins of Asteroidal Rock Disaggregation: Interplay of Thermal Fatigue and Microstructure”, Icarus, 2017, in press. Arxiv:1701.03510)

```
. . .
OOF.Material.Assign(material='Chondrule',microstructure='image3.png', pixels='Chondrule')

for T in Ts:
  Pname='T='+str(T).rjust(5,'0')
  OOF.Mesh.Set_Field_Initializer(mesh='image3.png:skeleton:mesh',field=Temperature...
  OOF.Mesh.Apply_Field_Initializers(mesh='image3.png:skeleton:mesh')
  . . .
```

Automation

The OOF team is also interested in exploring opportunities for data extraction from online databases, and integration into emerging workflow systems where it might add considerable value.

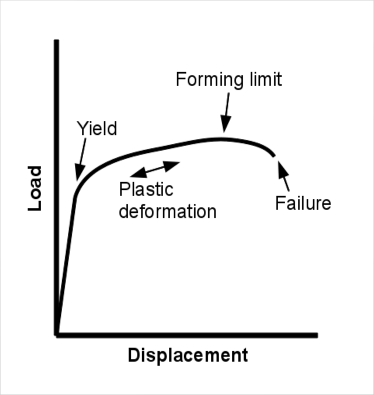
- Push-button extraction of property data from external databases
  - Materials Project
  - MDCS instances
  - Materials Data Facility?

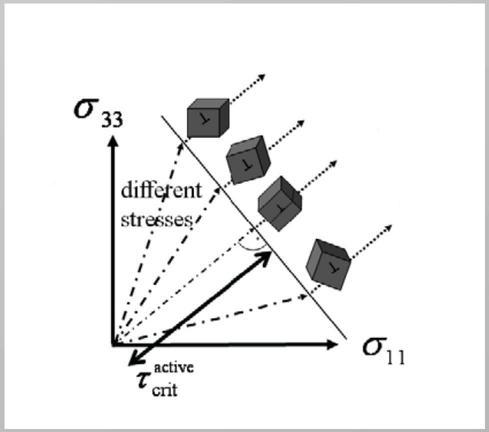
- Ingestion and generation of standard data formats for existing multiscale or multitool workflows
  - Dream3D, HDF5
  - PRISMS, ICE, others?

Development Focus: Crystal Plasticity

The crystal plasticity problem spans different scientific and engineering communities, and multiple length scales, and would benefit from a materials-focused real-space tool.

- Mechanical properties are fundamental to materials behavior
- Plasticity is fundamental to mechanical properties of metals
- Crystal plasticity couples crystallography to macro behavior
- Path from plasticity to forming traverses many length scales
- Input data comes from many diverse communities

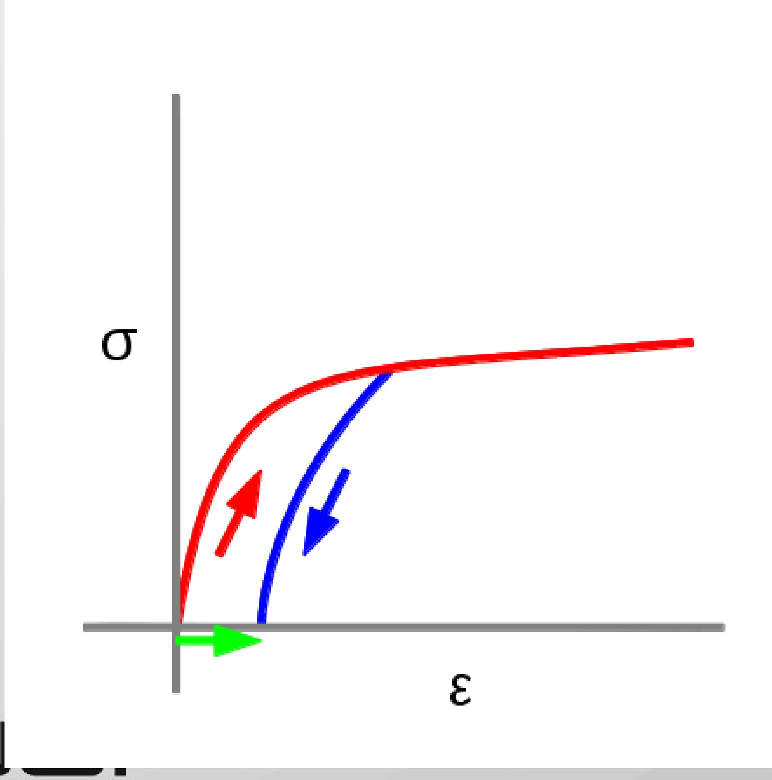




National Institute of Standards and Technology
U.S. Department of Commerce

Development Focus: Crystal Plasticity

"Classical" Plasticity



- Profoundly nonlinear
  - Inequality constraints
- History-dependent
  - Path-dependent state
  - Many possible variables
  - Possible rate-dependence

National Institute of Standards and Technology
U.S. Department of Commerce

Development Focus: Crystal Plasticity

$$\sigma = \sum_i \gamma_i \nabla \dot{\phi} + \sum_i k_i \nabla \phi$$

+ field-dependent and non-analytic features

Plasticity is not a straightforward PDE, has history-dependent info, and inequality constraints

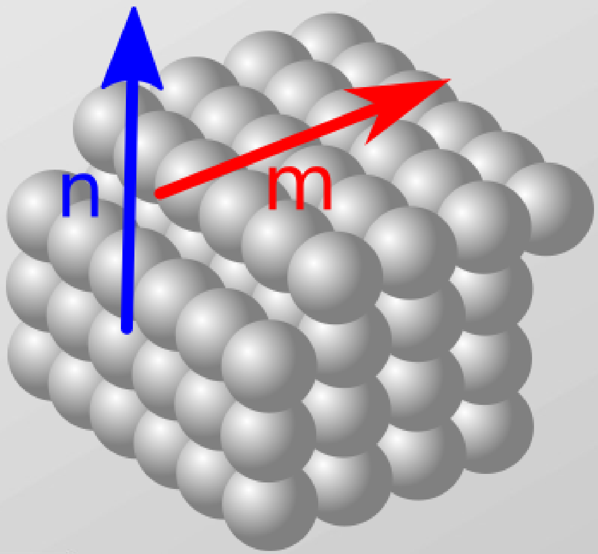
Approach:
Make contact with plasticity experts from the experimental and computational mechanics community. (NIST NCAL, CMU, Johns Hopkins)

Adopt the best existing models, build from there.

Development Focus: Crystal Plasticity

“Crystal” Plasticity

Has the same phenomenology, but the plastic response is made up of contributions from individual slip systems, dependent on the crystallography.



Accumulated slip of the plane with normal **n** in the direction **m** contributes to the plastic strain rate by the outer product of m and n.

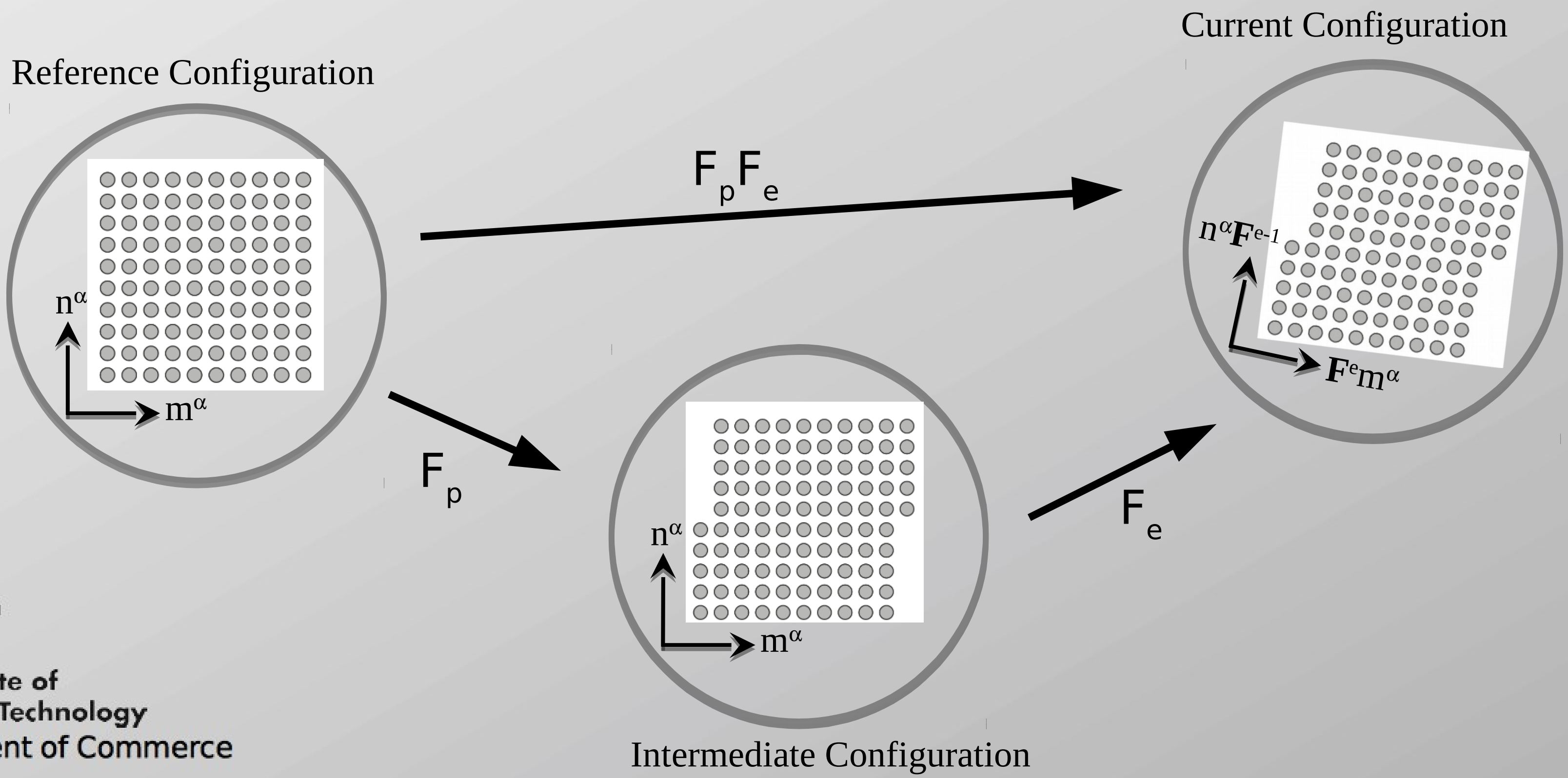
The total plastic strain rate is the sum of all moving slip systems.

$$L_p = \sum_{\alpha} \dot{\gamma}^{\alpha} (\mathbf{m}^{\alpha} \otimes \mathbf{n}^{\alpha})$$

Development Focus: Crystal Plasticity

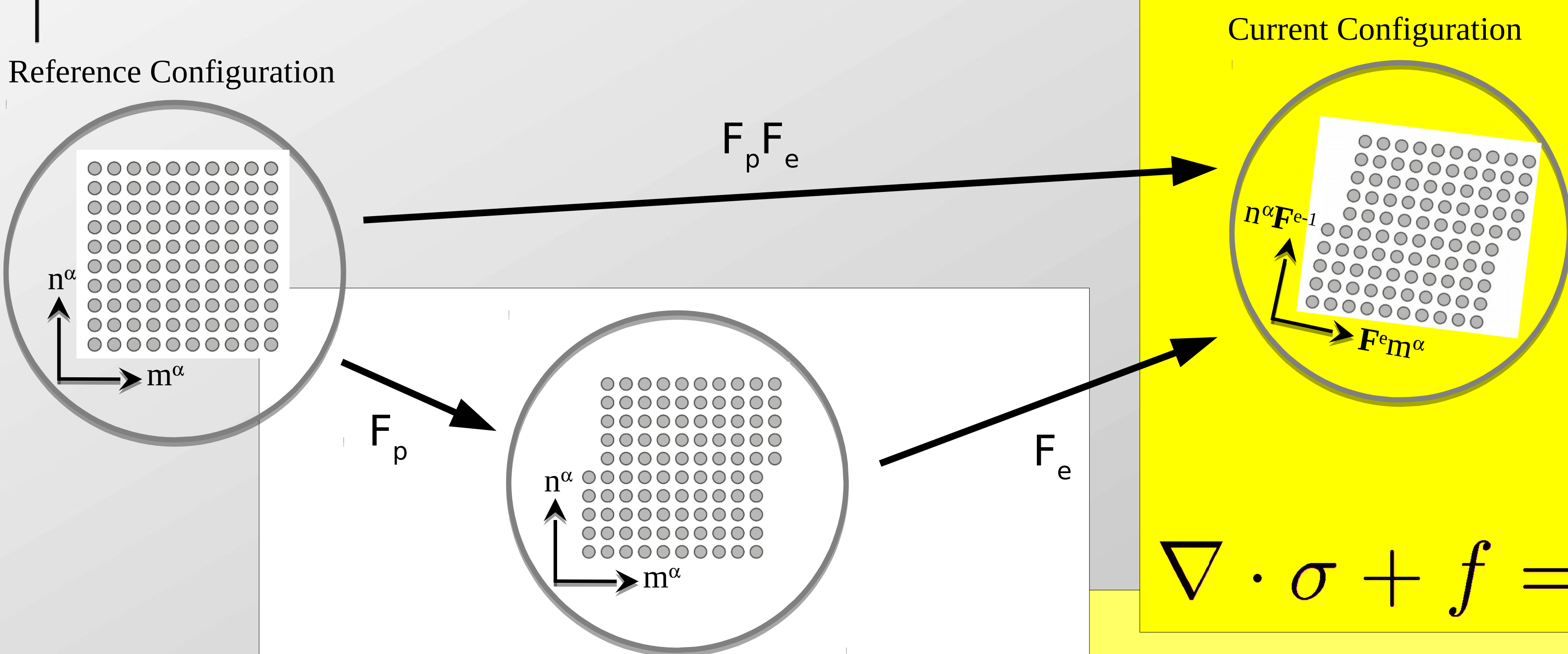
Challenge:

"Impedance mismatch" – computational mechanics practitioners are accustomed to codes tailored to the mechanics problem specifically, and have a mature, well-tested, non-extensible algorithm!



National Institute of Standards and Technology
U.S. Department of Commerce

Development Focus: Crystal Plasticity



$$\nabla \cdot \sigma + f = 0$$

$$L_p = \sum_\alpha \dot{\gamma}^\alpha (\mathbf{m}^\alpha \otimes \mathbf{n}^\alpha)$$

$$\sigma = \frac{F \cdot S \cdot F^T}{\det(F)}$$

$$\dot{\gamma}^\alpha = f(S_{ij}) \quad L_p \equiv \dot{F}_p F_p^{-1}$$

National Institute of
Standards and Technolo
U.S. Department of Co

Development Focus: Crystal Plasticity

Having learned this, our challenge is to incorporate these effects while retaining the generality of scope of the original code, **and** allowing for easily-pluggable plastic constitutive rules, packaged for Materials Science expert users.

Strategy: Prototype codes to explore the software-architecture issues which arise here.

- Need to store inter-time-step data at integration points
- Matrix construction process interacts with time-step size
- Need to do integrals (and derivatives) in the right space

National Institute of Standards and Technology
U.S. Department of Commerce

Development Focus: Crystal Plasticity

Status:

• First prototype (Python) completed, object structure settled.
• Second prototype (C++) completed, data management issues settled.
• Integral issues undertaken on a branch in the main repo.

$$L_p = \sum_\alpha \dot{\gamma}^\alpha (\mathbf{m}^\alpha \otimes \mathbf{n}^\alpha)$$

$$\dot{\gamma}^\alpha = f(S_{ij}) \quad L^p \equiv \dot{F}_p {F_p}^{-1}$$

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

Current Releases

- Newest 2D: OOF 2.1.13, December 2016
- Newest 3D: OOF 3.0.1, December 2016
- Full first- and second-order time-dependence (since 2.1.0/3.0)
- Sophisticated nonlinear solvers
- EBSD orientation-map capability (2D only, since 2.0)
- Nonlinearity-friendly property extension API
- Sophisticated meshing and image segmentation tools (since 2.0/3.0)
- Wide selection of built-in constitutive rules

In development

- Parallelization
  - Required for large data sets
- History-dependent properties
  - Viscosity, CPFEM
- Inequality constraints
  - Isotropic plasticity
  - Surface interactions

The OOF Team

Development:
Steve Langer, NIST/ITL
Andrew Reid, NIST/MML
Shahriyar Keshavarz, NIST/Theiss
Günay Doğan, NIST/Theiss

David Feraud, NIST/U. Blaise Pascal
Lizhong Zhang (NIST/U. Blaise Pascal)
Yannick Congo (NIST/U. Blaise Pascal)
Valerie Coffman(formerly NIST/ITL, currently Xometry)
Rhonald Lua (formerly NIST/PSU, currently Baylor)
Edwin García (formerly NIST/PSU, currently Purdue)
Seung-Ill Haan (formerly UMBC, currently Samsung)
Andrew Roosen (formerly NIST/MSEL, currently U Delaware)

Testing and feedback:
Craig Carter, MIT
Edwin Fuller, NIST ret'd

http://www.ctcms.nist.gov/oof
https://github.com/usnistgov/OOF2
https://github.com/usnistgov/OOF3D

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