The PyFEM Mesh Conversion Software Library
Dr Gordon W. H. German
30 October 2012

Computational Geoscience, Minerals Downunder, CSIRO
The PyFEM Library: Talk Overview

• What is it?
• Why is it?
• Design overview

• FEMStats – statistical add-on module

• Coding process: Agile scrums and 'bots called Jenkins

• Adding to the library, future collaborations and work
The PyFEM Library: What is it?

The PyFEM library is an open-software project for the conversion of 2D and 3D mesh formats.

- Primarily designed for conversion of Finite Element meshes as used in numerical simulations in the geosciences.

- Typically, meshes are constructed in one package and then solved in another, as solver codes generally don't provide intensive mesh editing facilities.
The PyFEM Library: What is it?

In general, PyFEM provides utility to the numerical modeller where:

- a numerical solver requires an input mesh data in one particular format, but the mesh is more easily created in another;
The PyFEM Library: What is it?

In general, PyFEM provides utility to the numerical modeller where:

- the output from a solver is better visualised or analysed in a different mesh format
The PyFEM Library: What is it?

In general, PyFEM provides utility to the numerical modeller where:

- continuous conversion is required between mesh formats to couple one solver to another, perhaps in different phenomenological domains.
The PyFEM Library: What is it?

PyFEM is open-source:

- Written in the Python scripting language for portability (some intensive sections written in C++).
- Object-oriented, highly commented code.
- Multiple programmers and reviewers.
- Auto-build, tested for Windows and Linux
- Single file or batch conversions
- Standalone scripting or run within the Desktop Modelling Toolkit software.
The PyFEM Library: What is it?

PyFEM currently supports meshes in the following formats:

- goCad
- Shemat
- Patran
- Finley
- XMML4FEM
- Generic ASCII
- FastFlo
- Flac3D
- Gmsh
- TecPlot
The PyFEM Library: Why is it?

Numerical modellers using proprietary codes needed a better way to build complex models.
The PyFEM Library: Why is it?

Building the mesh outside of the solver was (circa 1999) a novel approach:

- Needed a platform independent approach – computational resource could be quite different architecture to the modeller's desktop.
- Needed an extensible design model.
- Open-source meant that others could contribute to addition of various reader/writer modules.
- Interpretive code (vs compiled) made writing/testing within disparate environments simpler.
The PyFEM Library: Why is it?

Multiple personnel from various research groups became involved (circa 2002):

- Gordon German, Peter Hornby, Rob Woodcock from SCOM, CSIRO
- Simon Cox, Thomas Poulet from Computational Geosciences, CSIRO
- Lutz Gross from University of Queensland.

Distributed contributors meant a “team model” of development was required.

XMML4FEM developed as an xml representation of a multi-dimensional finite element mesh.
The PyFEM Library: Design Overview

Uses central (core) class to represent mesh.

Converter classes for each mesh format.

Core class must hold ALL known representations.

Ideally require lossless translation.
Central (core) class to represent mesh: PyFEM.py

Class: PyFEM

Public Methods:
- MakeHomogeneous()
- AddElement()
- AddNode()
- AddSurface()
- AddSubSurface()
- AddRegion()
- AddSampleValue()
- GetSampleValue()
- NodeOrder()
- BoundingBox()
The PyFEM Library: Design Overview

Conversion classes comprise of:

- `ReadFile(file_obj)`
- `Read(stream)` – main routines
- `WriteFile(mesh_obj)`
- `Write(stream)` – main routines
- Test routines:
  - `isHomogeneous()`
  - `isRegular()`
  - `isOrtho()`
  - `elemTypes()`
Multi-mesh “Post-processing” module that enables:

- User-defined functions based on *a priori* data existing in the mesh across multiple models, e.g. create new variable “average volumetric strain rate” from existing strain rate data for 300 results meshes.
- Find value of $x$ at given xyz coord for all meshes, e.g. useful for particle history in a time sequence.
- Sorting nodes by xyz coordinates.
- Writing out multiple mesh data as vtk or xmml4fem files for visualisation.
- “Clipping” each mesh and returning a specific portion,
- Re-sampling a set of given meshes.

Pre-processing functions, eg ancillary module “cleans” surfaces created in GoCad for correct mesh generation in Patran.
A “Rules” script is developed that contains the user functions and rules on how to apply them.

FEMStats runs the rules script over all applicable mesh files.

Rules files can be extended, contain multiple rules (that can be turned on or off or chained together).

Rules files are shared between modellers.
The PyFEM Library: FEMStats module

DMT provides a GUI for driving FEMStat data conversions
The PyFEM Library: The Coding Process

Distributed team approach to coding requires some discipline!

Based on “Agile” principles - some are “bent” :) 
- 6 month goal planning
- 3 week iteration planning
- Weekly 30min review meetings for whole team
- Daily 5min “scrum meetings” for local groups
- SVN repositories and Wiki collaborative pages
- Trunk **must** build
- Unit-tests on all major classes and methods
- Buildbot (Jenkins) auto-builds on trunk
- “Monthly” updates released
The PyFEM Library: The Coding Process

Distributed team approach to coding requires some discipline!

Using Jira for issue tracking (now CSIRO-wide):
- Issues added at iteration planning
- Issue assigned to relevant personnel (can be unassigned, reassigned)
- Issue resolved, left open for another iteration or destroyed
- Resolved issues reassigned for peer-review
- Reviewed issue closed.
The PyFEM Library: The Coding Process

It's not that bad!

And a lot better than this!
The PyFEM Library: Extending and additions

We encourage scientists working with volumetric data that have a specific format or translation problem to contact us:

Put in a request to the developers, or join the team and code your own, based on our templates

- Add it to the library
- Extend the FEMStats functionality
- Become a meshing guru for your own research team!

Read more on our Wiki:


Contact Gordon German, Minerals DownUnder CSIRO or email cg-dev@csiro.au

Thank You!