IMPRESS Documentation

Release 1.0

Artur Castiel

Nov 04, 2019

Contents

The **IMPRESS** is a preprocessor for general purpose capable of handling 2D and 3D meshes. It uses the robustness of PyMoab allied to a simple interface to generate all required mesh entities in a simulation by receiving only coarsening parameters (if necessary) and a mesh file supported by MOAB (see which files are supported in the Supported Files Formats session of MOAB description).

IMPRESS is on development phase, but if you'd like to use it, get instalation informations. Next, you can learn how to run it here.

Documentation coming soon!

CHAPTER 1

Contents:

1.1 Installing

1.1.1 Via Repo

The only way to use IMPRESS is to explicitly download it from our repo or simply fork it.

1.1.2 Dependencies

The simplest way to run IMPRESS is to use Docker as a container software with a image developed by our team that provides all the dependecies of IMPRESS. The image file is stored in a directory called `docker-image`. In that case, the procedure to use IMPRESS is:

- Install Docker correctly;
- Build the image stored in the directory called `docker-image`;
    1. Open your terminal and navigate to `docker-image` directory
    2. Build the `Dockerfile`

```
docker build -t image_name .
```

- Run Docker using the image you've just built through your terminal with the command:

```
sudo docker run -t -it -v path/to/repo/directory:/root image_name bash -c "cd /root;␣
↪ipython"
```

If all steps were completed properly, you're running iPython inside your recently created container and all files from IMPRESS repo must be available to be accessed.

1.2 Tutorials

1.2.1 Getting Started with IMPRESS

1.2.1.1 Mesh file

To use the preprocessor, open the *preprocessor.py* file in the root directory and insert the name of the mesh file to be preprocessed and its dimension. For example, with the line below, IMPRESS would look for '20.h5m' file and assume that this mesh is tridimensional:

```
M = msh('20.h5m', dim = 3)
```

1.2.1.2 Coarsening Settings

If you are interested in using a coarse mesh, it will be necessary to inform the partitioner scheme and coarsening ratio desired in the file msCoarse.yml. So far, IMPRESS handles only the Geometric Simple Partitioner Cube Based.

1.2.1.3 Variable Settings

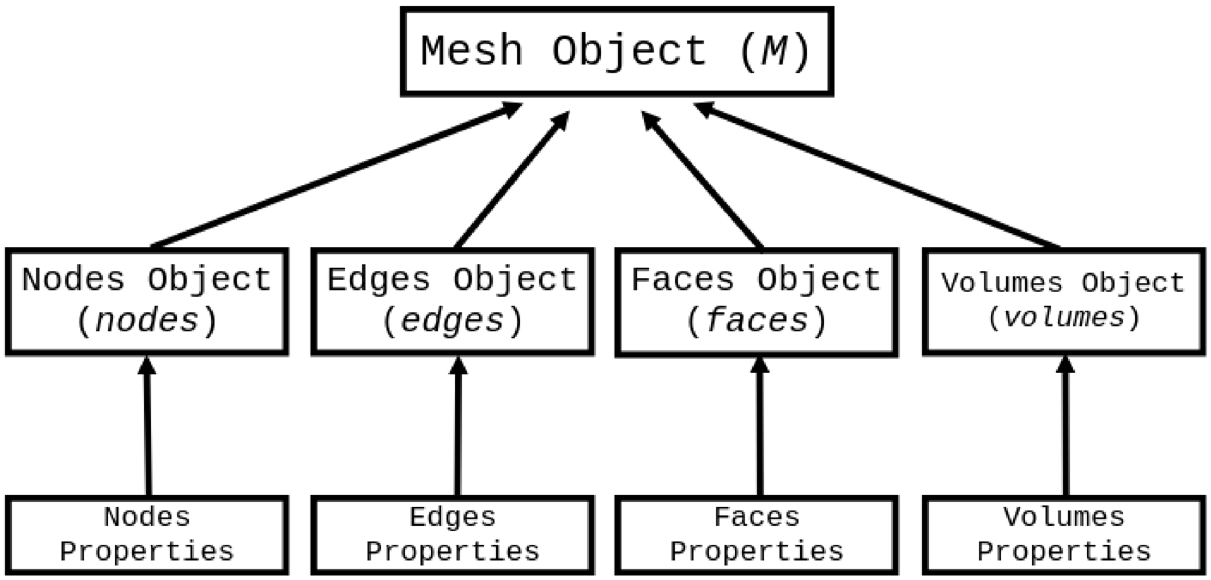
1.2.1.4 Run IMPRESS

Now, save preprocessor.py file and run it!

1.2.2 How to Access Mesh Entities Properties

To perform any kind of simulation, it will be necessary to access informations about the mesh entities such as coordinates from a node or from the center of an edge, for example. Probably, the user will also need informations about the adjacents elements or even the internal or boundary elements as well. We call these informations of **properties**.

IMPRESS was developed to automatically generated these properties for any kind of mesh entity (nodes, edges, faces or volumes, in a 3D mesh). Every time IMPRESS is executed, several objects are created to represent the mesh entities and inherited by a main class (that represents the mesh itself). A ilustrative image of the inheritance scheme follows below:



The IMPRESS' execution script instatiates this class creating an object called **M** through which it's possible to access all mesh entities. Furthermore, the user can obtain the mesh entity properties through its objects (which are literally called *nodes*, *edges*, *faces* and *volumes*).

1.2.2.1 Properties

The **properties** that IMPRESS provides are described below:

- **Coordinates**: returns the coordinates of an array of elements;
- **Center**: returns the coordinates of the center of an array of elements;
- **Boundary Elements**: returns the global id from all elements located in the mesh boundaries;
- **Internal Elements**: returns the global id from all elements that do not belong to the mesh boundaries;
- **Adjacencies**: returns the global id from all elements of the same dimension that are immediately connected to the element;
- **Connectivities**:
- **Flags**: returns a dictionary with all flags identified in the mesh.
- **Flagged Elements**: returns the global id from all flagged elements.
- **Global ID**: every type of mesh entity receives during the preprocessing a global ID, ranging from 0 to n-1 elements from the specific entity type. It is useful to perform many vectorized operations and identify elements that may receive a especial boundary condition as well, for example. It is especially important when a multiscale simulation is being perfomed.

These informations are stores in python objects:

Properties	Object	Input	Output
Coordinates	coords		
Boundary Elements	boundary		
Internal Elements	internal		
Adjacencies	adjacencies		
Connectivities	connectivies		
Flags	all_flags		
Flagged Elements	all_flagged_elements		
Global ID	global_id		

1.2.2.2 Consulting Properties

1.2.2.3 Coarse Scale Mesh Entities Properties

Coming soon!

1.2.3 Setting Boundary Conditions

1.3 Examples

1.3.1 Case 1

1.4 Packages

1.4.1 Core Module

1.4.2 Fine Scale Module

1.4.3 Mesh Components Module

CHAPTER 2

Indices and tables

- genindex
- modindex
- search