2023

Mechanical MNIST – Unsupervised Learning Dataset

Mechanical MNIST – Unsupervised Learning Dataset

Quan Nguyen, Emma Lejeune (contact: elejeune@bu.edu)

Department of Mechanical Engineering, Boston University

Dataset Description

The data in this collection is stored in two ".zip" folders. The first ".zip" folder ("heterogeneous_pattern_mesh.zip") contains the heterogeneous patterns, and the second ".zip" folder ("random_displacement_markers.zip") contains the displacement data for randomly sampled markers. In this document, we outline data organization and provide instructions on how to easily import and use these data.

```
from dolfin import *
from fenics import *

# directories
input_ = 'input_folder/'
filename = 'circle_cl_0_03'

# import mesh ------------------------------------------------
meshfile = input_ + filename + '.xdmf'
mesh = Mesh(MPI.comm_world) # MPI.comm_world | MPI.comm_self
xdmf = XDMFFile(MPI.comm_world,input_ + filename+'.xdmf')
xdmf.read(mesh)
# 3D domains
mvc = MeshValueCollection("size_t", mesh, 3)
xdmf.read(mvc,'regions')
sub = cpp.mesh.MeshFunctionSizet(mesh, mvc)
dx = Measure('dx',subdomain_data=sub,domain=mesh)
# ------------------------------------------------------------
```

Figure 1. Code to import the mesh so that it is compatible with FEniCS.

1. Heterogeneous Patterns Mesh Folder

Data organization

The first .zip folder (heterogeneous_pattern_mesh.zip) contains the mesh files for our 6 heterogeneous patterns and our homogeneous pattern:

- Circle inclusion: "circle_cl_0_03.xdmf" and "circle_cl_0_03.h5"
- Ring inclusion: "ring_cl_0_03.xdmf" and "ring_cl_0_03.h5"
- Cross inclusion: "cross_cl_0_03.xdmf" and "cross_cl_0_03.h5"
- 4 circle inclusions: "4cirs_cl_0_03.xdmf" and "4cirs_cl_0_03.h5"
- Cahn-Hillard pattern: "cahn_hilliard_Image12_cl_0_02.xdmf" and "cahn_hilliard_Image12_cl_0_02.h5"
- Split domain: "split_cl_0_03.xdmf" and "split_cl_0_03.h5"

- Homogeneous sample: “homogeneous_cl_0_03.xdmf” and “homogeneous_cl_03.h5”

Using the data

With these mesh files, we can either view the files or use them in Finite Element simulations. To view the data, open the “.xdmf” file in Paraview (note: the “.xdmf” and the “.h5” files with the same name must be in the same folder). To import the mesh file into python and use alongside the FEniCS package, follow the python script shown in **Figure 1**.

```python
import numpy as np
from cluster_project import kinematics as kn
from cluster_project import cluster, plotting

# user inputs for size of sample
length_samp = 1

# circle inclusion / neo-hookean
disp_path = 'files/example_data/circle_inclusion_NH'
pt_loc_files = np.array(['pt_sssc_equi_disp0.4.txt','pt_sssc_uni_y_disp0.4.txt',\
                         'pt_sssc_uni_x_disp0.4.txt','pt_sssc_shear_0.4.txt',\
                         'pt_sssc_comp_0.2.txt'])
u_mat_files = np.array(['disp_sssc_equi_disp0.4.txt','disp_sssc_uni_y_disp0.4.txt',\
                         'disp_sssc_uni_x_disp0.4.txt','disp_sssc_shear_0.4.txt',\
                         'disp_sssc_comp_0.2.txt'])
pt_loc_all,u_mat_all = kn.load_multiple(disp_path,pt_loc_files,u_mat_files)
disp_type = np.array(['equibiaxial','uniaxial y','uniaxial x','shear','confined compression'])
domain_type = 'circle_inclusion' # necessary for obtaining ground truth / ARI

# generate grid markers
pt_len = 8000
points_sel = kn.sample_points(pt_len,L=length_samp)

# obtain kinematics at grid markers for each file
num_neigh=40 # number of neighbors to interpolate and obtain grid markers
u_mat_list,grad_u_list,strain_list,I_strain_list,F_list,I_F_list,C_list,I_C_list,b_list,I_b_list =
kn.get_kinematics_multiple(pt_loc_all,u_mat_all,points_sel,num_neigh)
```

Figure 2. Code to import the mesh so it is compatible with FEniCS. The “cluster_project” package provides the “kinematics” module to calculate kinematic features given displacements from randomly sampled markers. Before calculating kinematics, we need to prepare pairs of marker locations (files starting with “pt_”) and corresponding marker displacements (files starting with “disp_”). We can load all marker locations and displacements with the function “kn.load_multiple()”. Then, we can create a regularly spaced grid of markers for interpolation using “kn.sample_points()”, where the outputs contain the position of the grid markers. Finally, with the positions and displacements of the randomly sampled markers alongside the positions of the grid markers, we can compute the kinematic features for the grid markers with the function “kn.get_kinematics()”. More detailed information is available in the GitHub repository for this project.

2. Random Displacement Markers Folder

Data organization

The second .zip folder (random_displacement_markers.zip) contains the displacement values for approximately 1500 uniformly random sampled markers per simulation. Here, we organize these sets of

displacement data into 16 folders. Each set of displacement data has a “pt” file with the location of the randomly sampled markers, and a “disp” file with the displacement values correspond to the markers. The naming convention for the folders is “*pattern*” (e.g., “circle_inclusion”) + “*constitutive model*” (e.g., “NH” for neo-Hookean samples, and “HO” for Holzapfel-Ogden samples). If the name of a folder contains no other information aside from the default, then that folder contains 5 sets of controlled boundary conditions (e.g., “circle_inclusion_NH” contains 5 sets of displacement data for the circle inclusion pattern with the neo-Hookean model). If the name of a folder contains “_random”, then that folder contains 6 sets of displacement data for 6 random boundary conditions (e.g., “circle_inclusion_NH_random” contains 6 sets of displacement data for 6 random boundary conditions for the circle inclusion pattern with the neo-Hookean model). Finally, for the Holzapfel-Ogden samples, the folders with “_diff_mat” represent samples where the ground substance storing the fibers also has different properties (e.g., “halfhalf_HO_diff_mat_random“ contains 6 sets of displacement data for 6 random boundary conditions for the split domain, where the sub-domains have different fiber orientation and ground substance material properties).

Using the data

Using the python package accompanying this dataset (“clustering_project”), we can interpolate the displacements from the randomly sampled markers on a grid. Then, from the interpolated displacements, we can compute the kinematic features of regularly spaced grid markers. The process of importing the files and computing the gridded kinematic features is shown in **Figure 2**.