Hardness: A Python toolbox for indentation simulation

Release

Jan 15, 2018

Contents

CHAPTER 1

Install

```
pip install hardness
```

CHAPTER 2

Tutorials:

2.1 Notebooks

2.1.1 Indentation

Note: This notebook can be downloaded here: `indentation_2D.ipynb`

2D indentation using Argiope & Hardness

```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

```
!cat local_settings.py.example
```

```
# LOCAL SETTINGS
# Modify as needed and rename local_settings.py
GMSH_PATH    = "gmsh"
ABAQUS_PATH  = "abaqus"
```

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import hardness as hd
import argiope as ag
import pandas as pd
import numpy as np
import os, subprocess, time, local_settings, time
```

```
%matplotlib nbagg

mpl.rcParams['grid.color'] = 'k'
mpl.rcParams['grid.linestyle'] = ':'
mpl.rcParams['grid.linewidth'] = 0.5
mpl.rcParams['contour.negative_linestyle'] = 'solid'

# USEFUL FUNCTIONS
def create_dir(path):
  try:
    os.mkdir(path)
  except:
    pass
```

Settings

```
# SETTINGS
workdir   = "_workdir/"
outputdir = "outputs/"
label    = "indentation_2D"

create_dir(workdir)
create_dir(workdir + outputdir)
```

Model definition

```
#--------------------------------------------------------------------------------
# MESH DEFINITIONS
def element_map(mesh):
    mesh.elements.loc[mesh.elements.type.argiope == "tri3", ("type", "solver", "")] =
↪"CAX3"
    mesh.elements.loc[mesh.elements.type.argiope == "quad4", ("type", "solver", "")]␣
↪= "CAX4R"
    return mesh


def sample_material_map(mesh):
    mesh.elements["materials"] = "SAMPLE_MAT"
    return mesh

def indenter_material_map(mesh):
    mesh.elements["materials"] = "INDENTER_MAT"
    return mesh


parts = {
    "sample" : hd.models.Sample2D(lx = 1., ly = 1.,
                                  r1 = 2., r2 = 3.,
                                  Nx = 32, Ny = 16,
                                  Nr = 2, Nt = 64,
                                  gmsh_path = "gmsh",
                                  file_name = "dummy",
                                  workdir = workdir,
                                  gmsh_space = 2,
```

```
                                    gmsh_options = "-algo 'delquad'",
                                    element_map = element_map,
                                    material_map = sample_material_map),

    "indenter" : hd.models.SpheroconicalIndenter2D(
                                    R = 1.,
                                    psi= 70.3,
                                    r1 = 1.,
                                    r2 = 3.,
                                    r3 = 4.,
                                    lc1 = .05,
                                    lc2 = .5,
                                    rigid = False,
                                    gmsh_path = "gmsh",
                                    file_name = "dummy",
                                    workdir = workdir,
                                    gmsh_space = 2,
                                    gmsh_options = "-algo 'delquad'",
                                    element_map = element_map,
                                    material_map = indenter_material_map)}

materials = [ag.materials.Hollomon(label = "SAMPLE_MAT", strain_data_points = 100),
             ag.materials.Hollomon(label = "INDENTER_MAT", strain_data_points = 100)]

#-------------------------------------------------------------------------------
# STEP DEFINTIONS
steps = [
        hd.models.Step2D(name = "LOADING1",
                         control_type = "disp",
                         duration = 1.,
                         kind = "adaptative",
                         nframes = 50,
                         controlled_value = -0.2,
                         field_output_frequency = 99999),
        hd.models.Step2D(name = "UNLOADING1",
                         control_type = "force",
                         duration = 1.,
                         kind = "adaptative",
                         nframes = 50,
                         controlled_value = 0.,
                         field_output_frequency = 99999),
        hd.models.Step2D(name = "RELOADING1",
                         control_type = "disp",
                         duration = 1.,
                         kind = "adaptative",
                         nframes = 50,
                         controlled_value = -0.2,
                         field_output_frequency = 99999),
        hd.models.Step2D(name = "LOADING2",
                         control_type = "disp",
                         duration = 1.,
                         kind = "adaptative",
                         nframes = 50,
                         controlled_value = -0.4,
                         field_output_frequency = 99999),
        hd.models.Step2D(name = "UNLOADING2",
                         control_type = "force",
                         kind = "adaptative",
```

```
                          duration = 1.,
                          nframes = 50,
                          controlled_value = 0.,
                          field_output_frequency = 99999)
        ]

model0 = hd.models.Indentation2D(label = label,
                        parts = parts,
                        steps = steps,
                        materials = materials,
                        solver = "abaqus",
                        solver_path = local_settings.ABAQUS_PATH,
                        workdir = workdir,
                        verbose = True)
```

```
print("1: Preprocessing ---------------------------------")
%time model0.write_input()
print("2: Processing ------------------------------------")
%time model0.run_simulation()
print("3: Postprocessing --------------------------------")
%time model0.postproc()
print("4: Saving model ----------------------------------")
%time model0.save(workdir + "model.pcklz")
```

```
1: Preprocessing ---------------------------------
CPU times: user 1.51 s, sys: 32 ms, total: 1.54 s
Wall time: 2.79 s
2: Processing ------------------------------------
<Running "indentation_2D" using abaqus>
(b'Abaqus JOB indentation_2DnAbaqus 6.13-1nBegin Analysis Input File␣
 ↪ProcessornTue Jan  9 17:10:52 2018nRun prenTue Jan  9 17:10:55 2018nEnd␣
 ↪Analysis Input File ProcessornBegin Abaqus/Standard AnalysisnTue Jan  9␣
 ↪17:10:55 2018nRun standardnTue Jan  9 17:12:33 2018nEnd Abaqus/Standard␣
 ↪AnalysisnAbaqus JOB indentation_2D COMPLETEDn', None)
<Ran indentation_2D: duration 104.45s>
CPU times: user 4 ms, sys: 12 ms, total: 16 ms
Wall time: 1min 44s
3: Postprocessing --------------------------------
<Post-Processing"indentation_2D" using abaqus>
/opt/abaqus/scratch/Commands/abaqus viewer noGUI=indentation_2D_abqpp.py
(b'', None)
<Post-Processed indentation_2D: duration 5.33s>
CPU times: user 324 ms, sys: 52 ms, total: 376 ms
Wall time: 5.67 s
4: Saving model ----------------------------------
CPU times: user 1.2 s, sys: 0 ns, total: 1.2 s
Wall time: 1.2 s
```

```
model = ag.utils.load(workdir + "model.pcklz")
```

Model checking

Mesh building and quality checking.

```
parts["indenter"].mesh.elements.head()
```

```
parts["sample"].mesh.elements.head()
```

```
i = 1
fig = plt.figure()
parts_names = parts.keys()
for name, part in parts.items():
    mesh = part.mesh
    patches = mesh.to_polycollection(edgecolor = "black", linewidth = .5, alpha = 1.)
    stats = mesh.stats()
    patches.set_array( stats.stats.max_abs_angular_deviation )
    patches.set_cmap(mpl.cm.YlOrRd)
    ax = fig.add_subplot(1, 2, i)
    ax.set_aspect("equal")
    ax.set_xlim(mesh.nodes.coords.x.min(), mesh.nodes.coords.x.max())
    ax.set_ylim(mesh.nodes.coords.y.min(), mesh.nodes.coords.y.max())
    ax.add_collection(patches)
    cbar = plt.colorbar(patches, orientation = "horizontal")
    cbar.set_label("Max Abs. Angular Deviation [$^o$]")
    plt.xlabel("$x$")
    plt.ylabel("$y$")
    plt.grid()
    plt.title(name.title())
    i+= 1
plt.show()
```

```
<IPython.core.display.Javascript object>
```

Simulation

Post-Processing

Time data

```
hist = model.data["history"]
hist.head()
```

```
plt.figure()
for step, group in hist.groupby("step"):
  plt.plot(-group.dtot, -group.F, label = "Step {0}".format(step))
plt.grid()
plt.legend(loc = "best")
plt.ylabel("Total force $F$, []")
plt.xlabel("Displacement, $\delta$ []")
plt.show()
```

```
<IPython.core.display.Javascript object>
```

Fields

```
model.parts["sample"].mesh.fields_metadata()
```

```
model.parts["sample"].mesh.fields_metadata()
```

```
parts = {k:part.mesh.copy() for k, part in model.parts.items() }

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.set_aspect("equal")
ax.set_xlim(0., 4.)
ax.set_ylim(-2., 2.)

field_num = 14
disp_num = 15
levels = np.linspace(-1.e-2, 1.e-2, 11)

for k, mesh in parts.items():
    field = mesh.fields[field_num].data.v12
    disp = mesh.fields[disp_num].data
    mesh.nodes[("coords", "x")] += disp.v1
    mesh.nodes[("coords", "y")] += disp.v2
    tri = mesh.to_triangulation()
    patches = mesh.to_polycollection(facecolor = "none",
                                     edgecolor = "black",
                                     linewidth = .5)

    grad = ax.tricontourf(tri, field, levels, cmap = mpl.cm.jet, alpha = 1.)
    ax.tricontour(tri, field, levels, colors = "white", linewidths = 1.)
    ax.add_collection(patches)
cbar = plt.colorbar(grad)
cbar.set_label("Cauchy Stress, $\sigma_{12}$")
ax.axis("off")
plt.xlabel("$x$")
plt.ylabel("$y$")
#plt.grid()
```

```
<IPython.core.display.Javascript object>
```

```
<matplotlib.text.Text at 0x7fda4246bb00>
```

2.1.2 Fibre Indentation

Workdir

Reports

Note: This notebook can be downloaded here: `indentation_2D_Fibre.ipynb`

2D indentation of a fibre in a matrix

```
%load_ext autoreload
%autoreload 2
```

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import hardness as hd
import argiope as ag
import pandas as pd
import numpy as np
import os, subprocess, time, local_settings, time
%matplotlib nbagg

mpl.rcParams['grid.color'] = 'k'
mpl.rcParams['grid.linestyle'] = ':'
mpl.rcParams['grid.linewidth'] = 0.5
mpl.rcParams['contour.negative_linestyle'] = 'solid'

# USEFUL FUNCTIONS
def create_dir(path):
  try:
    os.mkdir(path)
  except:
    pass
```

Settings

```
# SETTINGS
workdir   = "workdir/"
outputdir = "outputs/"
label     = "indentation_2D"

create_dir(workdir)
create_dir(workdir + outputdir)
```

Model definition

```
#-------------------------------------------------------------------------------
# MESH DEFINITIONS
def element_map(mesh):
    mesh.elements.loc[mesh.elements.type.argiope == "tri3", ("type", "solver", "")] =
↪"CAX3"
    mesh.elements.loc[mesh.elements.type.argiope == "quad4", ("type", "solver", "")]␣
↪= "CAX4"
    return mesh


def sample_material_map(mesh):
    mesh.elements.loc[mesh.elements.sets.FIBRE, "materials"] = "FIBRE_MAT"
    mesh.elements.loc[mesh.elements.sets.MATRIX, "materials"] = "MATRIX_MAT"
    return mesh
```

```
def indenter_material_map(mesh):
    mesh.elements["materials"] = "INDENTER_MAT"
    return mesh


parts = {
    "sample" : hd.models.SampleFibre2D(Rf = 1.,
                                       ly1 = 1., ly2 = 10.,
                                       Nx = 16, Ny = 8,
                                       Nr = 8, Nt = None,
                                       gmsh_path = "gmsh",
                                       file_name = "dummy",
                                       workdir = workdir,
                                       gmsh_space = 2,
                                       gmsh_options = "-algo 'delquad'",
                                       element_map = element_map,
                                       material_map = sample_material_map),

    "indenter" : hd.models.SpheroconicalIndenter2D(
                                       R = 1.,
                                       psi= 70.3,
                                       r1 = 1.,
                                       r2 = 3.,
                                       r3 = 3.,
                                       lc1 = .1,
                                       lc2 = .5,
                                       rigid = False,
                                       gmsh_path = "gmsh",
                                       file_name = "dummy",
                                       workdir = workdir,
                                       gmsh_space = 2,
                                       gmsh_options = "-algo 'delquad'",
                                       element_map = element_map,
                                       material_map = indenter_material_map)}

materials = [ag.materials.ElasticPerfectlyPlastic(
                                      label = "MATRIX_MAT",
                                      young_modulus = .1,
                                      poisson_ratio = .3,
                                      yield_stress = .001),
             ag.materials.Elastic(label = "INDENTER_MAT",
                                      young_modulus = 1.,
                                      poisson_ratio = .3),
             ag.materials.Elastic(label = "FIBRE_MAT",
                                      young_modulus = .4,
                                      poisson_ratio = .3)]

#--------------------------------------------------------------------------------
# STEP DEFINTIONS
steps = [
        hd.models.Step2D(name = "LOADING1",
                         control_type = "disp",
                         duration = 1.,
                         kind = "adaptative",
                         nframes = 100,
                         controlled_value = -0.2,
                         field_output_frequency = 99999),
        hd.models.Step2D(name = "UNLOADING1",
```

```
                        control_type = "force",
                        duration = 1.,
                        kind = "adaptative",
                        nframes = 100,
                        controlled_value = 0.,
                        field_output_frequency = 99999),
        hd.models.Step2D(name = "RELOADING1",
                        control_type = "disp",
                        duration = 1.,
                        kind = "adaptative",
                        nframes = 100,
                        controlled_value = -0.2,
                        field_output_frequency = 99999),
        hd.models.Step2D(name = "LOADING2",
                        control_type = "disp",
                        duration = 1.,
                        kind = "adaptative",
                        nframes = 100,
                        controlled_value = -0.4,
                        field_output_frequency = 99999),
        hd.models.Step2D(name = "UNLOADING2",
                        control_type = "force",
                        kind = "adaptative",
                        duration = 1.,
                        nframes = 50,
                        controlled_value = 0.,
                        field_output_frequency = 99999)
        ]

model0 = hd.models.Indentation2D(label = label,
                        parts = parts,
                        steps = steps,
                        materials = materials,
                        solver = "abaqus",
                        solver_path = local_settings.ABAQUS_PATH,
                        workdir = workdir,
                        verbose = True)
```

```
print("1: Preprocessing ---------------------------------")
%time model0.write_input()
print("2: Processing ------------------------------------")
%time model0.run_simulation()
print("3: Postprocessing --------------------------------")
%time model0.postproc()
print("4: Saving model ----------------------------------")
%time model0.save(workdir + "model.pcklz")
```

```
1: Preprocessing ---------------------------------
CPU times: user 1.48 s, sys: 20 ms, total: 1.5 s
Wall time: 2.46 s
2: Processing ------------------------------------
<Running "indentation_2D" using abaqus>
(b'Abaqus JOB indentation_2DnAbaqus 6.13-1nBegin Analysis Input File␣
 ↪ProcessornSun Jan 14 22:02:21 2018nRun prenSun Jan 14 22:02:26 2018nEnd␣
 ↪Analysis Input File ProcessornBegin Abaqus/Standard AnalysisnSun Jan 14␣
 ↪22:02:26 2018nRun standardnSun Jan 14 22:03:11 2018nEnd Abaqus/Standard␣
 ↪AnalysisnAbaqus JOB indentation_2D COMPLETEDn', None)
```

```
<Ran indentation_2D: duration 54.94s>
CPU times: user 4 ms, sys: 8 ms, total: 12 ms
Wall time: 54.9 s
3: Postprocessing ---------------------------------
<Post-Processing"indentation_2D" using abaqus>
/opt/abaqus/scratch/Commands/abaqus viewer noGUI=indentation_2D_abqpp.py
(b'', None)
<Post-Processed indentation_2D: duration 11.13s>
CPU times: user 296 ms, sys: 16 ms, total: 312 ms
Wall time: 11.4 s
4: Saving model ----------------------------------
CPU times: user 88 ms, sys: 0 ns, total: 88 ms
Wall time: 89.5 ms
```

```
model = ag.utils.load(workdir + "model.pcklz")
```

Model checking

Mesh building and quality checking.

```
model.parts["indenter"].mesh.elements.head()
```

```
model.parts["sample"].mesh.elements.head()
```

```
parts = model.parts
i = 1
fig = plt.figure()
parts_names = parts.keys()
for name, part in parts.items():
    mesh = part.mesh
    patches = mesh.to_polycollection(edgecolor = "black", linewidth = .5, alpha = 1.)
    stats = mesh.stats()
    patches.set_array( stats.stats.max_abs_angular_deviation )
    patches.set_cmap(mpl.cm.YlOrRd)
    ax = fig.add_subplot(1, 2, i)
    ax.set_aspect("equal")
    ax.set_xlim(mesh.nodes.coords.x.min(), mesh.nodes.coords.x.max())
    ax.set_ylim(mesh.nodes.coords.y.min(), mesh.nodes.coords.y.max())
    ax.add_collection(patches)
    cbar = plt.colorbar(patches, orientation = "horizontal")
    cbar.set_label("Max Abs. Angular Deviation [$^o$]")
    plt.xlabel("$x$")
    plt.ylabel("$y$")
    plt.grid()
    plt.title(name.title())
    i+= 1
plt.show()
```

```
<IPython.core.display.Javascript object>
```

Simulation

Post-Processing

Time data

```
hist = model.data["history"]
hist.head()
```

```
plt.figure()
for step, group in hist.groupby("step"):
  plt.plot(-group.dtot, -group.F, label = "Step {0}".format(step))
plt.grid()
plt.legend(loc = "best")
plt.ylabel("Total force $F$, []")
plt.xlabel("Displacement, $\delta$ []")
plt.show()
```

```
<IPython.core.display.Javascript object>
```

Fields

```
model.parts["sample"].mesh.fields_metadata()
```

```
model.parts["sample"].mesh.fields_metadata()
```

```
parts = {k:part.mesh.copy() for k, part in model.parts.items() }

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.set_aspect("equal")
ax.set_xlim(0., 3.)
ax.set_ylim(-2., 2.)

field_num = 14
disp_num = 15
levels = np.linspace(-1.e-1, 1.e-1, 11)

for k, mesh in parts.items():
    field = mesh.fields[field_num].data.v22
    disp = mesh.fields[disp_num].data
    mesh.nodes[("coords", "x")] += disp.v1
    mesh.nodes[("coords", "y")] += disp.v2
    tri = mesh.to_triangulation()
    patches = mesh.to_polycollection(facecolor = "none",
                                     edgecolor = "black",
                                     linewidth = .5)
    grad = ax.tricontourf(tri, field, levels, cmap = mpl.cm.terrain, alpha = 1)
    ax.tricontour(tri, field, levels, colors = "white", linewidths = 1.)
```

```
    ax.add_collection(patches)
cbar = plt.colorbar(grad)
cbar.set_label("Cauchy Stress, $\sigma_{12}$")
plt.xlabel("$x$")
plt.ylabel("$y$")
#plt.grid()
```

```
<IPython.core.display.Javascript object>
```

```
<matplotlib.text.Text at 0x7fd9740942e8>
```

```
parts["indenter"].fields
```

```
[<Tensor4Field S at node; step=0 ('LOADING1'); frame=0 >,
 <Vector2Field U at node; step=0 ('LOADING1'); frame=0 >,
 <Tensor4Field S at node; step=0 ('LOADING1'); frame=1 >,
 <Vector2Field U at node; step=0 ('LOADING1'); frame=1 >,
 <Tensor4Field S at node; step=1 ('UNLOADING1'); frame=0 >,
 <Vector2Field U at node; step=1 ('UNLOADING1'); frame=0 >,
 <Tensor4Field S at node; step=1 ('UNLOADING1'); frame=1 >,
 <Vector2Field U at node; step=1 ('UNLOADING1'); frame=1 >,
 <Tensor4Field S at node; step=2 ('RELOADING1'); frame=0 >,
 <Vector2Field U at node; step=2 ('RELOADING1'); frame=0 >,
 <Tensor4Field S at node; step=2 ('RELOADING1'); frame=1 >,
 <Vector2Field U at node; step=2 ('RELOADING1'); frame=1 >,
 <Tensor4Field S at node; step=3 ('LOADING2'); frame=0 >,
 <Vector2Field U at node; step=3 ('LOADING2'); frame=0 >,
 <Tensor4Field S at node; step=3 ('LOADING2'); frame=1 >,
 <Vector2Field U at node; step=3 ('LOADING2'); frame=1 >,
 <Tensor4Field S at node; step=4 ('UNLOADING2'); frame=0 >,
 <Vector2Field U at node; step=4 ('UNLOADING2'); frame=0 >,
 <Tensor4Field S at node; step=4 ('UNLOADING2'); frame=1 >,
 <Vector2Field U at node; step=4 ('UNLOADING2'); frame=1 >]
```

CHAPTER 3

Package documentation:

3.1 Models

Indentation dedicated classes and functions.

3.1.1 Meta classes

3.1.2 2D Indentation

3.1.3 Parts

Samples

Indenters

Steps

Inputs file generators

3.2 Post Processing

Post processing tools.