planarfibers

Release 0.0.2

Julian Karl Bauer

May 05, 2022

CONTENTS

1 Example Notebooks 1

CHAPTER

ONE

EXAMPLE NOTEBOOKS

1.1 Get points witin admissible parameter space

[1]:
```
import planarfibers
import pandas as pd
pd.set_option('display.max_columns', 100)
pd.set_option('display.width', 1000)
```

[2]:
```
df = planarfibers.discretization.get_points_on_slices(
    radii=["0", "1/2", "9/10"],
    la1s=["1/2", "4/6", "5/6", "1"],
    numeric=False,
)
```

[3]:
```
print(df)
```

```
                    la1 radius_factor   beta      r                          d_1            ␣
↪ d_8
v00-upper-0         1/2           9/10   pi/2   9/80                       69/560            ␣
↪    0
v00-upper-1         2/3           9/10   pi/2   1/10                       61/630            ␣
↪    0
v00-upper-2         5/6           9/10   pi/2   1/16                      89/5040            ␣
↪    0
v00-mid-0           1/2              0      0      0                        3/280            ␣
↪    0
v00-mid-1           2/3              0      0      0                       -1/315            ␣
↪    0
v00-mid-2           5/6              0      0      0                    -113/2520            ␣
↪    0
v00-lower-0         1/2           9/10  -pi/2   9/80                      -57/560            ␣
↪    0
v00-lower-1         2/3           9/10  -pi/2   1/10                      -13/126            ␣
↪    0
v00-lower-2         5/6           9/10  -pi/2   1/16                    -541/5040            ␣
↪    0
v00-mid-3             1              0      0      0                        -4/35            ␣
↪    0
vshc-central-la1-0  1/2              0      0      0                        3/280            ␣
↪    0
```

(continues on next page)

(continued from previous page)

```
vshc-m90-0-la1-0     1/2          1/2  -pi/2   1/16                 -29/560      ␣
↪    0
vshc-m90-1-la1-0     1/2         9/10  -pi/2   9/80                 -57/560      ␣
↪    0
vshc-m45-0-la1-0     1/2          1/2  -pi/4   1/16      3/280 - sqrt(2)/32    ␣
↪sqrt(2)/32
vshc-m45-1-la1-0     1/2         9/10  -pi/4   9/80    3/280 - 9*sqrt(2)/160 ␣
↪9*sqrt(2)/160
vshc-0-0-la1-0       1/2          1/2      0   1/16                   3/280      ␣
↪1/16
vshc-0-1-la1-0       1/2         9/10      0   9/80                   3/280      ␣
↪9/80
vshc-45-0-la1-0      1/2          1/2   pi/4   1/16      3/280 + sqrt(2)/32    ␣
↪sqrt(2)/32
vshc-45-1-la1-0      1/2         9/10   pi/4   9/80    3/280 + 9*sqrt(2)/160 ␣
↪9*sqrt(2)/160
vshc-90-0-la1-0      1/2          1/2   pi/2   1/16                  41/560      ␣
↪    0
vshc-90-1-la1-0      1/2         9/10   pi/2   9/80                  69/560      ␣
↪    0
vshc-central-la1-1   2/3            0      0      0                  -1/315      ␣
↪    0
vshc-m90-0-la1-1     2/3          1/2  -pi/2   1/18                 -37/630      ␣
↪    0
vshc-m90-1-la1-1     2/3         9/10  -pi/2   1/10                 -13/126      ␣
↪    0
vshc-m45-0-la1-1     2/3          1/2  -pi/4   1/18       -sqrt(2)/36 - 1/315    ␣
↪sqrt(2)/36
vshc-m45-1-la1-1     2/3         9/10  -pi/4   1/10       -sqrt(2)/20 - 1/315    ␣
↪sqrt(2)/20
vshc-0-0-la1-1       2/3          1/2      0   1/18                  -1/315      ␣
↪1/18
vshc-0-1-la1-1       2/3         9/10      0   1/10                  -1/315      ␣
↪1/10
vshc-45-0-la1-1      2/3          1/2   pi/4   1/18       -1/315 + sqrt(2)/36    ␣
↪sqrt(2)/36
vshc-45-1-la1-1      2/3         9/10   pi/4   1/10       -1/315 + sqrt(2)/20    ␣
↪sqrt(2)/20
vshc-90-0-la1-1      2/3          1/2   pi/2   1/18                  11/210      ␣
↪    0
vshc-90-1-la1-1      2/3         9/10   pi/2   1/10                  61/630      ␣
↪    0
vshc-central-la1-2   5/6            0      0      0               -113/2520      ␣
↪    0
vshc-m90-0-la1-2     5/6          1/2  -pi/2  5/144               -401/5040      ␣
↪    0
vshc-m90-1-la1-2     5/6         9/10  -pi/2   1/16               -541/5040      ␣
↪    0
vshc-m45-0-la1-2     5/6          1/2  -pi/4  5/144  -113/2520 - 5*sqrt(2)/288 ␣
↪5*sqrt(2)/288
vshc-m45-1-la1-2     5/6         9/10  -pi/4   1/16     -113/2520 - sqrt(2)/32    ␣
↪sqrt(2)/32
```

(continues on next page)

(continued from previous page)

```
vshc-0-0-la1-2      5/6          1/2      0  5/144                     -113/2520     ␣
↪5/144
vshc-0-1-la1-2      5/6         9/10      0   1/16                     -113/2520      ␣
↪1/16
vshc-45-0-la1-2     5/6          1/2   pi/4  5/144  -113/2520 + 5*sqrt(2)/288 ␣
↪5*sqrt(2)/288
vshc-45-1-la1-2     5/6         9/10   pi/4   1/16     -113/2520 + sqrt(2)/32    ␣
↪sqrt(2)/32
vshc-90-0-la1-2     5/6          1/2   pi/2  5/144                     -17/1680      ␣
↪   0
vshc-90-1-la1-2     5/6         9/10   pi/2   1/16                      89/5040      ␣
↪   0
```

1.2 Homogenize for representative N4 in slices of parameter space

[1]:
```
import planarfibers
import matplotlib.pyplot as plt
import pandas as pd
import vofotensors
from vofotensors.abc import la1, d1, d8
import sympy as sp
import numpy as np
```

[2]:
```
pd.set_option("display.max_columns", 100)
pd.set_option("display.width", 1000)
```

[3]:
```
# User input
SCALE_HOMOGENEOUS = False
key_quantity = "E_modulus"
homogenization_function = planarfibers.approximation.calc_MTOA
la1_values = ["3 / 6", "4 / 6", "5 / 6"]
```

[4]:
```
# Get points on views
df = planarfibers.discretization.get_points_on_slices(
    radii=["0", "1/2", "9/10"],
    la1s=list(map(eval, la1_values)) + ["1"],
    numeric=True,
)
```

[5]:
```
# Get fiber orientation tensors
parameterizations = vofotensors.fabric_tensors.N4s_parametric
parameterization = parameterizations["planar"]["la1_d1_d8"]
N4_func = sp.lambdify([la1, d1, d8], parameterization)
df["N4"] = df.apply(
    lambda row: N4_func(la1=row["la1"], d1=row["d_1"], d8=row["d_8"]), axis=1
)
```

[6]:
```
# Define angle discretization
angles = np.linspace(0, 2 * np.pi, 120)
```

[7]:
```
# Homogenize
df["stiffness"] = df.apply(
    lambda row: homogenization_function(N4=row["N4"], inp=None),
    axis=1,
)
```

[8]:
```
# Define helper func to explicitly select either Youngs or generalized compression mod.
def get_Youngs_modulus(stiffness, angles):
    projector = planarfibers.utils.PlanarStiffnesProjector()
    E, K = projector.get_planar_E_K(stiffness=stiffness, angles=angles)
    if key_quantity == "E_modulus":
        return E
    elif key_quantity == "K_modulus":
        return K
```

[9]:
```
# Get Youngs-Modulus for direction in plane
df[key_quantity] = df.apply(
    lambda row: get_Youngs_modulus(stiffness=row["stiffness"], angles=angles),
    axis=1,
)
```

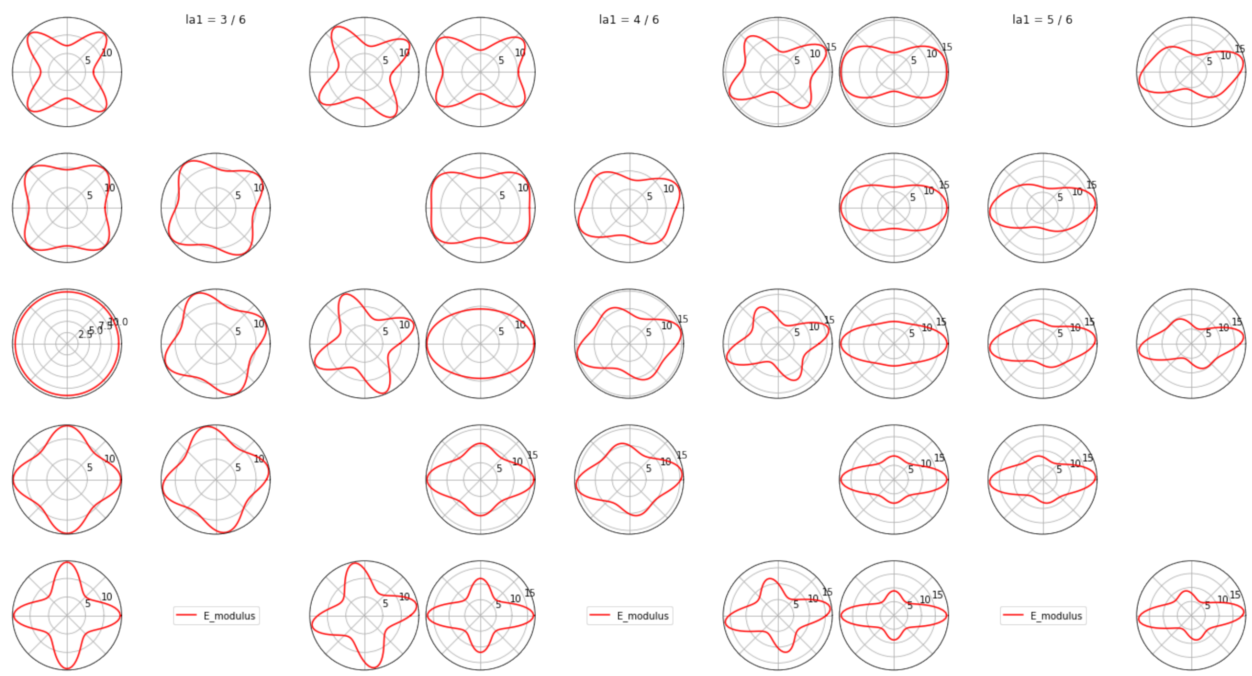
[10]:
```
# Define layout
la1_key_extensions = {f"-la1-{index}": value for index, value in enumerate(la1_values)}
# la1_key_extensions = {
# "-la1-0": "3 / 6",
# "-la1-1": "4 / 6",
# "-la1-2": "5 / 6",
# }
grid_indices = {
    "vshc-central": (2, 0),
    "vshc-m90-0": (3, 0),
    "vshc-m90-1": (4, 0),
    "vshc-m45-0": (3, 1),
    "vshc-m45-1": (4, 2),
    "vshc-0-0": (2, 1),
    "vshc-0-1": (2, 2),
    "vshc-45-0": (1, 1),
    "vshc-45-1": (0, 2),
    "vshc-90-0": (1, 0),
    "vshc-90-1": (0, 0),
}
legend_axis_indices = (4, 1)
empty_axes_indices = [(0, 1), (1, 2), (3, 2), legend_axis_indices]
```

[11]:
```
# Plot first view
nbr_slices = len(la1_key_extensions)
fig = plt.figure(figsize=(6 * nbr_slices, 10))
subfigs = fig.subfigures(1, nbr_slices, wspace=0.0)
for index, (key_extension, la1val) in enumerate(la1_key_extensions.items()):
    subfig = subfigs[index]
    subfig.suptitle(f"la1 = {la1val}")
    axs = subfig.subplots(ncols=3, nrows=5, subplot_kw={"projection": "polar"})
```

(continues on next page)

(continued from previous page)

```
    lines = []  # Initialize legend lines
    for key_N4_start, grid_index in grid_indices.items():
        ax = axs[grid_index]
        key = key_N4_start + key_extension

        ax.plot(angles, df.loc[key][key_quantity], label=key_quantity, color="red")

        # Update legend if something has been plotted
        lines_tmp, labels_tmp = ax.get_legend_handles_labels()
        if len(lines_tmp) > len(lines):
            lines, labels = lines_tmp, labels_tmp

    for ax in axs.flatten():
        ax.set_xticklabels([])
        if SCALE_HOMOGENEOUS:
            ax.set_ylim(
                bottom=0,
                top=1.2,
            )
            ax.set_yticks([0, 0.5, 1])

    legend_axis = axs[legend_axis_indices]
    legend_axis.legend(lines, labels, loc="center")

    for indice in empty_axes_indices:
        ax = axs[indice]
        ax.axis("off")
fig.tight_layout()
```



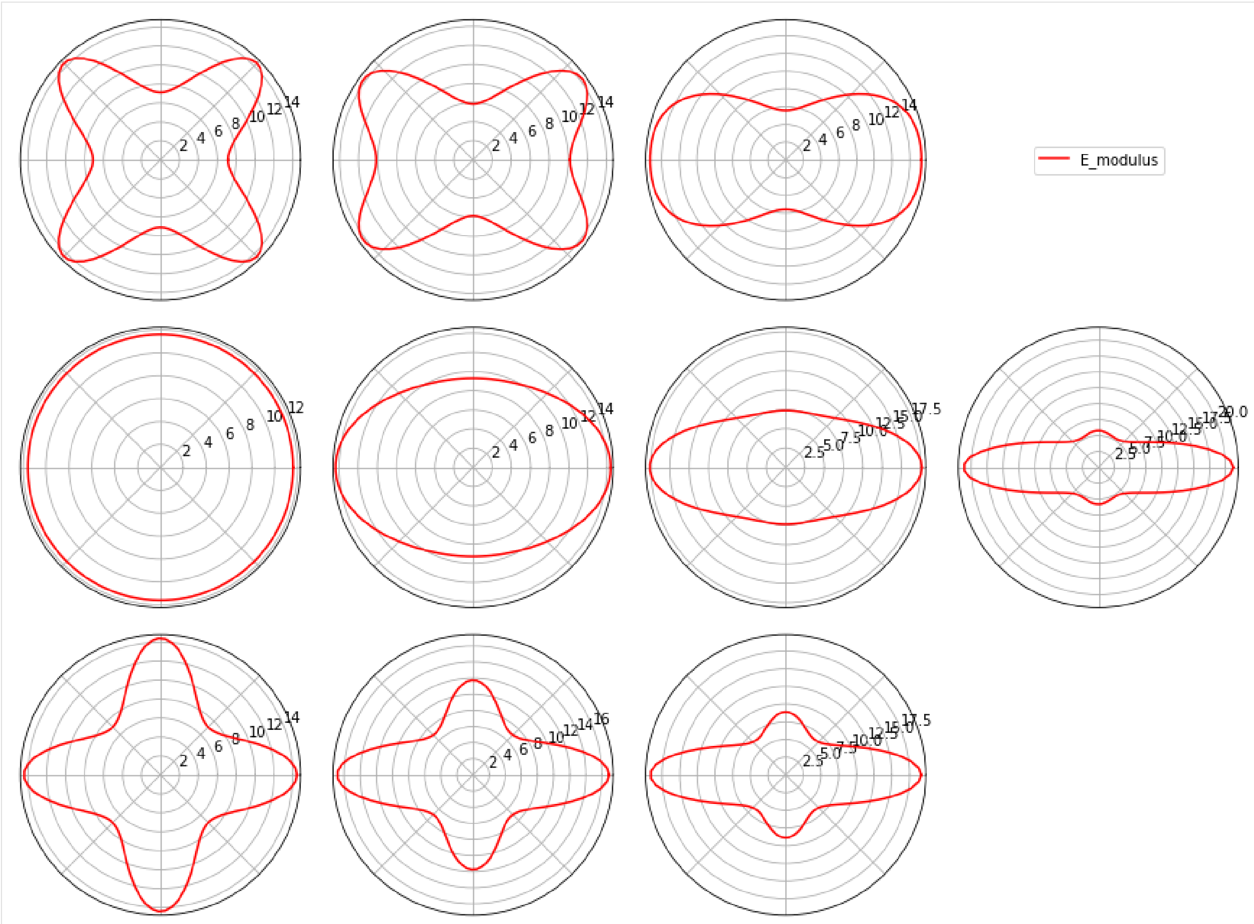
[12]:
```
# Define layout second view
```

(continues on next page)

(continued from previous page)

```
grid_indices = {
    "v00-upper-0": (0, 0),
    "v00-upper-1": (0, 1),
    "v00-upper-2": (0, 2),
    #
    "v00-mid-0": (1, 0),
    "v00-mid-1": (1, 1),
    "v00-mid-2": (1, 2),
    #
    "v00-lower-0": (2, 0),
    "v00-lower-1": (2, 1),
    "v00-lower-2": (2, 2),
    #
    "v00-mid-3": (1, 3),
    #
}
legend_axis_indices = (0, 3)
empty_axes_indices = [(2, 3), legend_axis_indices]
```

[13]:
```
# Plot second dview
fig, axs = plt.subplots(
    figsize=(12, 9), ncols=4, nrows=3, subplot_kw={"projection": "polar"}
)
lines = []  # Initialize legend lines
for key_N4_start, grid_index in grid_indices.items():
    ax = axs[grid_index]
    key = key_N4_start

    ax.plot(angles, df.loc[key][key_quantity], label=key_quantity, color="red")

    # Update legend if something has been plotted
    lines_tmp, labels_tmp = ax.get_legend_handles_labels()
    if len(lines_tmp) > len(lines):
        lines, labels = lines_tmp, labels_tmp
for ax in axs.flatten():
    ax.set_xticklabels([])
    if SCALE_HOMOGENEOUS:
        ax.set_ylim(
            bottom=0,
            top=1.2,
        )
        ax.set_yticks([0, 0.5, 1])
legend_axis = axs[legend_axis_indices]
legend_axis.legend(lines, labels, loc="center")
for indice in empty_axes_indices:
    ax = axs[indice]
    ax.axis("off")
fig.tight_layout()
```



plt.show()

1.3 Integrate random planar FODF with numerical averager

[1]:
```
import planarfibers
import pandas as pd
import numpy as np
```

[2]:
```
pd.set_option("display.max_columns", 100)
pd.set_option("display.width", 1000)
np.set_printoptions(linewidth=400)
```

[3]:
```
# Select random FODF: choose exact closre FODF, see equation (77) in "Fiber orientation␣
↪distributions based on planar fiber orientation tensors of fourth order. Math. Mech.␣
↪Solids (to appear 2022)"".
la1 = 5 / 6
odf_func = lambda phi: ((1.0 - la1) * la1) / (
    2.0 * np.pi * (la1**2 + (1.0 - 2.0 * la1) * np.cos(phi) ** 2)
)
```

[4]:
```
# Select quantity which is to be averaged
quantity = planarfibers.approximation.calc_MT_UD()
print(quantity)
```

```
[[24.52696446  4.87908133  4.87908133  0.          0.          0.        ]
 [ 4.87908133  9.12135685  5.36895671  0.          0.          0.        ]
 [ 4.87908133  5.36895671  9.12135685  0.          0.          0.        ]
 [ 0.          0.          0.          3.75240015  0.          0.        ]
 [ 0.          0.          0.          0.          3.97000178  0.        ]
 [ 0.          0.          0.          0.          0.          3.97000178]]
```

[5]:
```
# Average
averager = planarfibers.averager.AveragerPlanar(odf_planar=odf_func)
average = averager.average(quantity)
print(average)
```

```
[[ 2.08517135e+01  5.98673099e+00  4.96072722e+00  0.00000000e+00  0.00000000e+00  8.
↪54110020e-16]
 [ 5.98673099e+00  1.05813085e+01  5.28731081e+00  0.00000000e+00  0.00000000e+00 -6.
↪43671067e-17]
 [ 4.96072722e+00  5.28731081e+00  9.12135685e+00  0.00000000e+00  0.00000000e+00 -8.
↪74005141e-18]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  3.78866709e+00  1.53209539e-17  0.
↪00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  5.77997479e-18  3.93373484e+00  0.
↪00000000e+00]
 [ 1.01914811e-15 -1.51996602e-16 -1.44620409e-17  0.00000000e+00  0.00000000e+00  6.
↪18530110e+00]]
```