Comptes Rendus
Mécanique
Supplementary material to the article available at:
https://doi.org/10.5802/crmeca.228

Supplementary material: Finite strain formulation of the discrete equilibrium gap principle: application to mechanically consistent regularization for large motion tracking

Document complémentaire : Formulation en transformation finie du principe d'écart d'équilibre discret : application à la régularisation mécanique pour le suivi de mouvement

Martin Genet [a, b]

[a] Laboratoire de Mécanique des Solides, École Polytechnique/IPP/CNRS, Palaiseau, France

[b] Équipe MΞDISIM, INRIA, Palaiseau, France

E-mail: martin.genet@polytechnique.edu

A. Code for Figures 1, 2, 3 and 4

Imports

[ ]:

```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_Struct import generate_images_and_meshes_from_Struct
from plot_disp_error_vs_regul_strength      import plot_disp_error_vs_regul_strength
from lib_viewer                              import Viewer
```

ISSN (electronic) : 1873-7234

Parameters

[ ]:

```
n_dim = 2

images_folder = "generate_images"

n_voxels = 100

structure_deformation_type_lst  = [                         ]
structure_deformation_type_lst += [["square", "translation"]]
structure_deformation_type_lst += [["square", "rotation"   ]]
structure_deformation_type_lst += [["square", "compression"]]
structure_deformation_type_lst += [["square", "shear"      ]]

texture_type_lst  = [         ]
texture_type_lst += ["tagging"]

noise_level_lst  = [   ]
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [   ]
mesh_size_lst += [0.1]

regul_type_lst  = [                                                          ]
regul_type_lst += ["continuous-linear-elastic"                               ]
regul_type_lst += ["continuous-linear-equilibrated"                          ]
regul_type_lst += ["continuous-elastic"                                      ]
regul_type_lst += ["continuous-equilibrated"                                 ]
regul_type_lst += ["discrete-simple-elastic"                                 ]
regul_type_lst += ["discrete-simple-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"           ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential"       ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                                   ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"                  ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"              ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential"       ]

regul_level_lst  = [        ]
regul_level_lst += [0.99    ]
regul_level_lst += [0.1*2**3]
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.0     ]

do_generate_images                   = 1
do_generate_meshes                   = 1
do_run_warp                          = 1
do_plot_disp_error_vs_regul_strength = 1
```

Synthetic images

[ ]:

```
if (do_generate_images):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for texture_type                     in texture_type_lst              :
   for noise_level                      in noise_level_lst               :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run in range(1, n_runs+1):

        print("*** generate_images ***"            )
        print("structure_type:"  , structure_type  )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )

        generate_images_and_meshes_from_Struct(
            n_dim            = n_dim                          ,
            n_voxels         = n_voxels                       ,
            structure_type   = structure_type                 ,
            deformation_type = deformation_type               ,
            texture_type     = texture_type                   ,
            noise_level      = noise_level                    ,
            k_run            = k_run if (n_runs > 1) else None,
            generate_images  = 1                              ,
            compute_meshes   = 0                              )
```

Ground truth motion

[ ]:

```
if (do_generate_meshes):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for mesh_size                        in mesh_size_lst                 :

    print("*** generate_meshes ***"            )
    print("structure_type:"  , structure_type  )
    print("deformation_type:", deformation_type)
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_Struct(
        n_dim            = n_dim           ,
        n_voxels         = n_voxels        ,
        structure_type   = structure_type  ,
        deformation_type = deformation_type,
        texture_type     = "no"             ,
        noise_level      = 0                ,
        mesh_size        = mesh_size        ,
        generate_images  = 0                ,
        compute_meshes   = 1                )
```

Tracking

[ ]:

```
if (do_run_warp):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for texture_type                     in texture_type_lst              :
   for noise_level                      in noise_level_lst              :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for mesh_size   in mesh_size_lst     :
      for regul_type  in regul_type_lst    :
       for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.0

        print("*** run_warp ***"                   )
        print("structure_type:"  , structure_type  )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("mesh_size:"       , mesh_size       )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = structure_type
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = images_folder

        mesh_basename  = structure_type
        mesh_basename += "-"+deformation_type
        mesh_basename += "-h="+str(mesh_size)
        if (structure_type == "heart"):
            mesh_basename += "-mesh"
        working_basename  = images_basename
        working_basename += "-h="+str(mesh_size)
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp(
            working_folder                              = working_folder  ,
            working_basename                            = working_basename,
            images_folder                               = images_folder   ,
            images_basename                             = images_basename ,
            mesh_folder                                 = mesh_folder     ,
            mesh_basename                               = mesh_basename   ,
            regul_type                                  = regul_type      ,
            regul_model                                 = regul_model     ,
            regul_level                                 = regul_level     ,
            regul_poisson                               = regul_poisson   ,
            relax_type                                  = "backtracking"  ,
            normalize_energies                          = 1               ,
            tol_dU                                      = 1e-2            ,
            n_iter_max                                  = 100             ,
            continue_after_fail                         = 1               ,
            write_VTU_files                             = 1               ,
            write_VTU_files_with_preserved_connectivity = 1               )
```

Visualization

[ ]:

```
structure_type = "square"

deformation_type = "translation"
# deformation_type = "rotation"
# deformation_type = "compression"
# deformation_type = "shear"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
# regul_level = 0.0

images_basename  = structure_type
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

[ ]:

```
if (do_plot_disp_error_vs_regul_strength):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for texture_type                     in texture_type_lst               :
   for regul_type                       in regul_type_lst                 :

    print("*** plot_disp_error_vs_regul_strength ***")
    print("structure_type:"  , structure_type  )
    print("deformation_type:", deformation_type)
    print("texture_type:"    , texture_type    )
    print("regul_type:"      , regul_type      )

    plot_disp_error_vs_regul_strength(
        images_folder           = images_folder          ,
        sol_folder              = working_folder         ,
        structure_type          = structure_type         ,
        deformation_type        = deformation_type       ,
        texture_type            = texture_type           ,
        regul_type              = regul_type             ,
        noise_level_lst         = noise_level_lst        ,
        n_runs_for_noisy_images = n_runs_for_noisy_images,
        regul_level_lst         = regul_level_lst        ,
        regul_level_for_zero    = 1e-3                   ,
        generate_datafile       = 1                      ,
        generate_plotfile       = 1                      ,
        generate_plot           = 1                      )

    plotfile_basename  = "plot_disp_error_vs_regul_strength"
    plotfile_basename += "/"+structure_type
    plotfile_basename += "-"+deformation_type
    plotfile_basename += "-"+texture_type
    plotfile_basename += "-"+regul_type
    IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```

B. Code for Figure 5

Imports

[ ]:

```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_HeartSlice import generate_images_and_meshes_from_HeartSlice
from plot_disp_error_vs_regul_strength          import plot_disp_error_vs_regul_strength
from lib_viewer                                 import Viewer
```

Parameters

[ ]:

```
images_folder = "generate_images"

n_voxels = 100

deformation_type_lst  = [                  ]
deformation_type_lst += ["contractandtwist"]

texture_type_lst  = [         ]
texture_type_lst += ["tagging"]

noise_level_lst  = [   ]
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [   ]
mesh_size_lst += [0.1]
```

```
regul_type_lst  = [                                                          ]
regul_type_lst += ["continuous-linear-elastic"                               ]
regul_type_lst += ["continuous-linear-equilibrated"                          ]
regul_type_lst += ["continuous-elastic"                                      ]
regul_type_lst += ["continuous-equilibrated"                                 ]
regul_type_lst += ["discrete-simple-elastic"                                 ]
regul_type_lst += ["discrete-simple-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"           ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential"       ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                                   ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"                  ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"              ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential"       ]

regul_level_lst  = [        ]
regul_level_lst += [0.99    ]
regul_level_lst += [0.1*2**3]
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.0     ]

do_generate_images                   = 1
do_generate_meshes                   = 1
do_run_warp                          = 1
do_plot_disp_error_vs_regul_strength = 1
```

Synthetic images

[ ]:

```
if (do_generate_images):
 for deformation_type in deformation_type_lst:

    print("*** running model ***"              )
    print("deformation_type:", deformation_type)

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               )

    for texture_type in texture_type_lst:
     for noise_level  in noise_level_lst :

        n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1
        for k_run in range(1, n_runs+1):

            print("*** generate_images ***"            )
            print("deformation_type:", deformation_type)
            print("texture_type:"    , texture_type    )
            print("noise_level:"     , noise_level     )
            print("k_run:"           , k_run           )

            generate_images_and_meshes_from_HeartSlice(
                n_voxels         = n_voxels                     ,
                deformation_type = deformation_type             ,
                texture_type     = texture_type                 ,
                noise_level      = noise_level                  ,
                k_run            = k_run if (n_runs > 1) else None,
                run_model        = 0                            ,
                generate_images  = 1                            )
```

Ground truth motion

[ ]:

```
if (do_generate_meshes):
 for deformation_type in deformation_type_lst:
  for mesh_size        in mesh_size_lst       :

    print("*** generate_meshes ***"            )
    print("deformation_type:", deformation_type)
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               ,
        mesh_size        = mesh_size       )
```

Tracking

[ ]:

```
if (do_run_warp):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for mesh_size   in mesh_size_lst     :
      for regul_type  in regul_type_lst    :
       for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp ***")
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("mesh_size:"       , mesh_size       )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = images_folder

        mesh_basename  = "heart"
        mesh_basename += "-"+deformation_type
        mesh_basename += "-h="+str(mesh_size)
        mesh_basename += "-mesh"

        working_basename = images_basename
        working_basename += "-h="+str(mesh_size)
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp(
            working_folder                              = working_folder  ,
            working_basename                            = working_basename,
            images_folder                               = images_folder   ,
            images_basename                             = images_basename ,
            mesh_folder                                 = mesh_folder     ,
            mesh_basename                               = mesh_basename   ,
            regul_type                                  = regul_type      ,
            regul_model                                 = regul_model     ,
            regul_level                                 = regul_level     ,
            regul_poisson                               = regul_poisson   ,
            relax_type                                  = "backtracking"  ,
            normalize_energies                          = 1               ,
            tol_dU                                      = 1e-2            ,
            n_iter_max                                  = 100             ,
            continue_after_fail                         = 1               ,
            write_VTU_files                             = 1               ,
            write_VTU_files_with_preserved_connectivity = 1               )
```

Visualization

[ ]:

```
deformation_type = "contractandtwist"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
# regul_level = 0.0

images_basename  = "heart"
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

[ ]:
```
if (do_plot_disp_error_vs_regul_strength):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for regul_type       in regul_type_lst      :

    print("*** plot_disp_error_vs_regul_strength ***")
    print("deformation_type:", deformation_type)
    print("texture_type:"    , texture_type    )
    print("regul_type:"      , regul_type      )

    plot_disp_error_vs_regul_strength(
        images_folder           = images_folder          ,
        sol_folder              = working_folder         ,
        structure_type          = "heart"                ,
        deformation_type        = deformation_type       ,
        texture_type            = texture_type           ,
        regul_type              = regul_type             ,
        noise_level_lst         = noise_level_lst        ,
        n_runs_for_noisy_images = n_runs_for_noisy_images,
        regul_level_lst         = regul_level_lst        ,
        regul_level_for_zero    = 1e-3                   ,
        generate_datafile       = 1                      ,
        generate_plotfile       = 1                      ,
        generate_plot           = 1                      )

    plotfile_basename  = "plot_disp_error_vs_regul_strength"
    plotfile_basename += "/"+"heart"
    plotfile_basename += "-"+deformation_type
    plotfile_basename += "-"+texture_type
    plotfile_basename += "-"+regul_type
    IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```

C. Code for Figure 6

Imports

[ ]:
```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_HeartSlice import generate_images_and_meshes_from_HeartSlice
from plot_disp_error_vs_mesh_size               import plot_disp_error_vs_mesh_size
from lib_viewer                                 import Viewer
```

Parameters

[ ]:
```
images_folder = "generate_images"

n_voxels = 100

deformation_type_lst  = [                  ]
deformation_type_lst += ["contractandtwist"]
```

```
texture_type_lst = []
texture_type_lst += ["tagging"]

noise_level_lst  = []
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [        ]
mesh_size_lst += [0.1     ]
mesh_size_lst += [0.1/2**1]
mesh_size_lst += [0.1/2**2]
mesh_size_lst += [0.1/2**3]
mesh_size_lst += [0.1/2**4]

regul_type_lst  = [                                                          ]
regul_type_lst += ["continuous-linear-elastic"                               ]
regul_type_lst += ["continuous-linear-equilibrated"                          ]
regul_type_lst += ["continuous-elastic"                                      ]
regul_type_lst += ["continuous-equilibrated"                                 ]
regul_type_lst += ["discrete-simple-elastic"                                 ]
regul_type_lst += ["discrete-simple-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated"                            ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"           ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential"       ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                                   ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"                  ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"              ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential"       ]

regul_level_lst  = [        ]
regul_level_lst += [0.0     ]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**3]
regul_level_lst += [0.99    ]

do_generate_images                          = 1
do_generate_meshes                          = 1
do_run_warp                                 = 1
do_run_warp_and_refine                      = 1
do_plot_disp_error_vs_mesh_size             = 1
do_plot_disp_error_vs_mesh_size_with_refine = 1
```

Synthetic images

[ ]:
```
if (do_generate_images):
 for deformation_type in deformation_type_lst:

    print("*** running model ***"             )
    print("deformation_type:", deformation_type)

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               )

    for texture_type in texture_type_lst:
     for noise_level  in noise_level_lst :

        n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

        for k_run in range(1, n_runs+1):

            print("*** generate_images ***"            )
            print("deformation_type:", deformation_type)
            print("texture_type:"    , texture_type    )
            print("noise_level:"     , noise_level     )
            print("k_run:"           , k_run           )

            generate_images_and_meshes_from_HeartSlice(
                n_voxels         = n_voxels                       ,
                deformation_type = deformation_type               ,
                texture_type     = texture_type                   ,
                noise_level      = noise_level                    ,
                k_run            = k_run if (n_runs > 1) else None,
                run_model        = 1                              ,
                generate_images  = 0                              )
```

Ground truth motion

[ ]:
```
if (do_generate_meshes):
 for deformation_type in deformation_type_lst:
  for mesh_size        in mesh_size_lst       :

    print("*** generate_meshes ***"            )
    print("deformation_type:", deformation_type)
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model       = 1              ,
        generate_images = 0              ,
        mesh_size       = mesh_size      )
```

Tracking (single-level)

[ ]:

```
if (do_run_warp):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for mesh_size   in mesh_size_lst     :
      for regul_type  in regul_type_lst    :
       for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp ***"                   )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("mesh_size:"       , mesh_size       )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = images_folder

        mesh_basename  = "heart"
        mesh_basename += "-"+deformation_type
        mesh_basename += "-h="+str(mesh_size)
        mesh_basename += "-mesh"

        working_basename = images_basename
        working_basename += "-h="+str(mesh_size)
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)
        dwarp.warp(
            working_folder                              = working_folder  ,
            working_basename                            = working_basename,
            images_folder                               = images_folder   ,
            images_basename                             = images_basename ,
            mesh_folder                                 = mesh_folder     ,
            mesh_basename                               = mesh_basename   ,
            regul_type                                  = regul_type      ,
            regul_model                                 = regul_model     ,
            regul_level                                 = regul_level     ,
            regul_poisson                               = regul_poisson   ,
            relax_type                                  = "backtracking"  ,
            normalize_energies                          = 1               ,
            tol_dU                                      = 1e-2            ,
            n_iter_max                                  = 100             ,
            continue_after_fail                         = 1               ,
            write_VTU_files                             = 1               ,
            write_VTU_files_with_preserved_connectivity = 1               ,
            print_iterations                            = 0               )
```

Tracking (multi-level)

[ ]:

```
if (do_run_warp_and_refine):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for regul_type  in regul_type_lst    :
      for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp_and_refine ***"        )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = "generate_images"

        mesh_basenames = []
        for mesh_size in mesh_size_lst:
            mesh_basename  = "heart"
            mesh_basename += "-"+deformation_type
            mesh_basename += "-h="+str(mesh_size)
            mesh_basename += "-mesh"

            mesh_basenames += [mesh_basename]

        working_basename = images_basename
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp_and_refine(
            working_folder      = working_folder     ,
            working_basename    = working_basename   ,
            images_folder       = images_folder      ,
            images_basename     = images_basename    ,
            mesh_folder         = mesh_folder        ,
            mesh_basenames      = mesh_basenames     ,
            regul_type          = regul_type         ,
            regul_model         = regul_model        ,
            regul_level         = regul_level        ,
            regul_poisson       = regul_poisson      ,
            relax_type          = "backtracking"     ,
            normalize_energies  = 1                  ,
            tol_dU              = 1e-2               ,
            n_iter_max          = 100                ,
            continue_after_fail = 1                  )
```

Visualization

[ ]:

```
deformation_type = "contractandtwist"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1       ; k_mesh_size = 0
# mesh_size = 0.1/2**1; k_mesh_size = 1
# mesh_size = 0.1/2**2; k_mesh_size = 2
# mesh_size = 0.1/2**3; k_mesh_size = 3
# mesh_size = 0.1/2**4; k_mesh_size = 4
```

```
with_refine = 1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
# regul_level = 0.0

images_basename  = "heart"
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
if not (with_refine):
    working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)
if (with_refine):
    working_basename += "-refine="+str(k_mesh_size)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

[ ]:

```
if (do_plot_disp_error_vs_mesh_size) or (do_plot_disp_error_vs_mesh_size_with_refine):

    with_refine_lst = []
    if (do_plot_disp_error_vs_mesh_size            ): with_refine_lst += [False]
    if (do_plot_disp_error_vs_mesh_size_with_refine): with_refine_lst += [True ]
```

```
for with_refine      in with_refine_lst     :
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for regul_type       in regul_type_lst      :

    plot_disp_error_vs_mesh_size(
        images_folder           = images_folder          ,
        sol_folder              = working_folder         ,
        structure_type          = "heart"         ,
        deformation_type        = deformation_type       ,
        texture_type            = texture_type           ,
        regul_type              = regul_type             ,
        noise_level_lst         = noise_level_lst        ,
        n_runs_for_noisy_images = n_runs_for_noisy_images,
        regul_level_lst         = regul_level_lst        ,
        mesh_size_lst           = mesh_size_lst          ,
        error_for_nan           = 10                     ,
        with_refine             = with_refine            ,
        generate_datafile       = 1                      ,
        generate_plotfile       = 1                      ,
        generate_plot           = 1                      )

    plotfile_basename  = "plot_disp_error_vs_mesh_size"
    if (with_refine):
        plotfile_basename += "-with_refine"
    plotfile_basename += "/"+"heart"
    plotfile_basename += "-"+deformation_type
    plotfile_basename += "-"+texture_type
    plotfile_basename += "-"+regul_type
    IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```