— Computer Aided Design Optimizer —

INSTALLATION GUIDE

CONTENTS

ABOUT

This document provides general information about using the CADO software. CADO is a fully CAD-integrated topology optimization tool under the open source BSD license. This project is a part of the Bavarian Graduate School of Engineering at TU München and was developed by Saumitra Joshi, Juan Carlos Medina, Friedrich Menhorn, Severin Reiz, Benjamin Rüth, Erik Wannerberg and Anna Yurova in 2015-2016. The figures in this document are provided as reference output for the user.

1 TOPY

In our tool we use ToPy (https://github.com/williamhunter/topy) for topology optimization.

1.1 Prerequisites

In order to install ToPy, make sure that the following software is installed on your computer:

- Python (version 2.7)
- NumPy (Usually provided by Python distribution)
- PyVTK tool (https://pypi.python.org/pypi/PyVTK)
- Pysparse library (http://pysparse.sourceforge.net/)

Here are some recommendations for the installation of the tools/libraries mentioned above.

To install PyVTK tool, please run the following commands in your terminal:

```
> sudo apt-get install python-pip

> pip install pyvtk
```

The installation of the Pysparse library is a bit more cumbersome, since the pip-installation (like in the previous case) fails most of the times. So, here we provide an alternative way of installing Pysparse from the *.git* repository.

To install Pysparse (assuming the pip installation fails), make sure that *git* (https://git-scm.com/) is installed on your computer and then run the following commands in your terminal:

```
> git clone git://pysparse.git.sourceforge.net/gitroot/
pysparse/pysparse/

> cd pysparse

> sudo python setup.py install
```

1.2 Install ToPy

If all the tools specified in the section 1.1 are installed, we can now proceed to the installation of ToPy itself. For that download ToPy from https://github.com/williamhunter/topy.

```
159
160 def _write_legacy_vtu(x, fname):
161     """
162     Write a legacy VTK unstructured grid file.
163
164     """
165     # Voxel local points relative to its centre of geometry:
166     voxel_local_points = asarray([[-1,-1,-1],[ 1,-1,-1],[-1, 1,-1],[ 1, 1,-1],
167                                   [-1,-1, 1],[ 1,-1, 1],[-1, 1, 1],[ 1, 1, 1]])\
168                                     * 0.5 # scaling
169     # Voxel world points:
170     points = []
171     # Culled input array -- as list:
172     xculled = []
173
174     try:
175         depth, rows, columns = x.shape
176     except ValueError:
177         sys.exit('Array dimensions not equal to 3, possibly 2-dimensional.\n')
178
179     for i in xrange(depth):
180         for j in xrange(rows):
181             for k in xrange(columns):
182                 if x[i,j,k] > THRESHOLD:
183                     xculled.append(x[i,j,k])
184                     points += (voxel_local_points + [i,j,k]).tolist()
185
186     voxels = arange(len(points)).reshape(len(xculled), 8).tolist()
187     topology = UnstructuredGrid(points, voxel = voxels)
188     file_header = \
189     'ToPy data, created '\
190     + str(datetime.now()).rsplit('.')[0]
191     scalars = CellData(Scalars(xculled, name='Densities', lookup_table =\
192     'default'))
193     vtk = VtkData(topology, file_header, scalars)
194     vtk.tofile(fname, 'ascii')
195
196 def  timestamp():
```

Figure 1: Changing of the output type of ToPy to `'ascii'`

For CADO it is necessary to have an output in the *ascii* format. By default the output *.vtk* files from ToPy are binary, so we need to change them to *ascii*. In order to do that, please perform the following actions:

- Open the ToPy source file *core/visualization.py*
- Go to the method `_write_legacy_vtu(x, fname)` in line 160
- Change `'binary'` to `'ascii'` in line 194 (see fig. 1)

After making the following edit, run the following command from the root directory of ToPy:

```
> sudo python setup.py install
```

1.3 Test ToPy

In order to test whether the installation of ToPy was completed successfully it is possible to run some test cases provided in *examples* folder. For that, do the following:

- Enter one of the folders in examples (e.g. examples/cantilever)

```
                        python optimise.py cantilvr_3d_etaopt_gsf.tpd

======================================================================
ToPy problem definition (TPD) file successfully parsed.
TPD file name: cantilvr_3d_etaopt_gsf.tpd (v2007)
----------------------------------------------------------------------
Domain discretisation (NUM_ELEM_X x NUM_ELEM_Y x NUM_ELEM_Z) = 28 x 37 x 111
Element type (ELEM_K) = H8
Filter radius (FILT_RAD) = 1.5
Number of iterations (NUM_ITER) = 50
Problem type (PROB_TYPE) = comp
Problem name (PROB_NAME) = cantilvr_3d_etaopt_gsf
GSF active
Damping factor (ETA) = 0.40
No passive elements (PASV_ELEM) specified
Active elements (ACTV_ELEM) specified
======================================================================
 Iter | Obj. func.   | Vol.  | Change     | P_FAC | Q_FAC | Ave ETA | S-V fra
----------------------------------------------------------------------
ToPy: Solution for FEA converged after 451 iterations.
   1  | 5.231335e+01 | 0.150 | 8.5000e-01 | 1.000 | 1.000 |  0.400  |  0.010
ToPy: Solution for FEA converged after 452 iterations.
   2  | 3.023124e+01 | 0.150 | 2.0000e-01 | 1.000 | 1.000 |  0.400  |  0.010
```

Figure 2: Output of the ToPy test

```
-- Processing ToolKit: TKVRML (VrmlConverter;VrmlAPI;Vrml;VrmlData)
-- Processing ToolKit: TKXCAF (XCAFApp;XCAFDoc;XCAFPrs)
-- Processing ToolKit: TKXCAFSchema (MXCAFDoc;PXCAFDoc;XCAFDrivers;XCAFSchema)
-- Processing ToolKit: TKXmlXCAF (XmlXCAFDrivers;XmlMXCAFDoc)
-- Processing ToolKit: TKBinXCAF (BinXCAFDrivers;BinMXCAFDoc)
-- Processing ToolKit: TKXDEIGES (IGESCAFControl)
-- Processing ToolKit: TKXDESTEP (STEPCAFControl)
-- Processing ToolKit: TKDraw (Draw;DBRep;DrawTrSurf)
-- Processing ToolKit: TKTopTest (TestTopOpeDraw;TestTopOpeTools;TestTopOpe;BRepTest;GeometryTest;HLRTest;MeshTest;GeomliteTest;DrawFairCu
rve;BOPTest;SWDRAW)
-- Processing ToolKit: TKViewerTest (ViewerTest)
-- Processing ToolKit: TKXSDRAW (SWDRAW;XSDRAW;XSDRAWIGES;XSDRAWSTEP;XSDRAWSTLVRML)
-- Processing ToolKit: TKDCAF (DDF;DDocStd;DNaming;DDataStd;DPrsStd;DrawDim)
-- Processing ToolKit: TKXDEDRAW (XDEDRAW)
-- Processing ToolKit: TKTObjDRAW (TObjDRAW)
-- Processing application: DRAWEXE (DRAWEXE)
-- Configuring done
-- Generating done
-- Build files have been written to:                    git/oce-master/build
```

Figure 3: OpenCascade installation: cmake

- Execute a ToPy test run by running the following command in your terminal:

  > `python optimize.py <example.tpd-file>`

The output should look as showed in picture 2.

2 OPENCASCADE

OpenCascade (http://www.opencascade.com/) is an open-source CAD kernel. It is widely used in engineering and design for geometry construction and editing.

2.1 Install OpenCascade

For technical reasons, we do not use OpenCascade from the official webpage, but from the *.git* repository.

To install OpenCascade this way, make sure that git (https://git-scm.com/) is installed on your computer and then run the following commands in your terminal:

- Clone the repository:

Figure 4: Sample output of the OpenCascade building process

```
> git clone git://github.com/tpaviot/oce.git
> cd oce
> mkdir build
> cd build
```

- Execute `cmake`:

  ```
  > cmake ..
  ```

  Sample output: see fig. 3

- Build OpenCascade:

  ```
  > make ..
  ```

  To speed up the process, build can be done in parrallel:

  ```
  > make -j<number_of_processors>
  ```

  Sample output: see fig. 4

- Install OpenCascade:

  ```
  > sudo make install ..
  ```

These steps are in accord with the installation guide on the Github page of OpenCascade itself. One can also use the CMake-GUI to change some of the build configuration if need be (e.g. include OpenMP support).

2.2 Test OpenCascade

In order to test whether the installation of OpenCascade was completed successfully it is possible to run a test provided by OpenCascade.

For that, run the following command from your terminal:

```
> make test
```

All performed tests should be successful (See fig. 5)

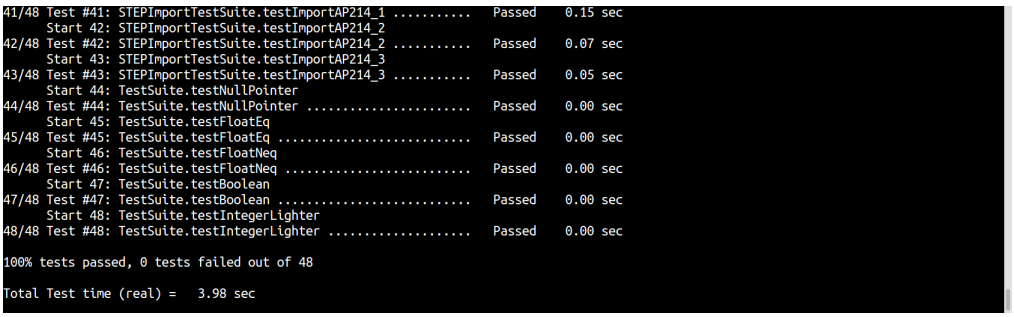

Figure 5: Sample output of the OpenCascade test

3 MISCELLANEOUS

3.1 Qt & QtCreator

To install the the newest version of **Qt**, visit the page http://ftp.fau.de/qtproject/archive/qt/5.4/5.4.2/ and download the *.run* file suitable for your computer. After that, change the rights for the installer file and install **Qt** by following instructions of the installation manager:

```
> chmod +x qt-opensource-linux-x64-5.5.0-2.run
```

```
> sudo ./qt-opensource-linux-x64-5.5.0-2.run
```

3.2 FreeCAD

Download and install FreeCAD following the instructions from the official FreeCAD webcite:
http://www.freecadweb.org/wiki/?title=Download.

It can also be installed directly from the command line as follows:

```
> sudo apt-get install freecad
```

4 CADO

4.1 Prerequisites

In order to install CADO the following tools should be installed on your computer:

- Topy (see Sec. 1)

- OpenCascade (see Sec. 2)
- QtCreator (see Sec. 3.1)
- FreeCAD

After having installed all the prerequisites, CADO is ready to install. To do that, perform the following command from the repository main folder:

```
> make
```

After the installation process has completed run the program from the command line:

```
> ./cado
```