1 Foreword

This documents gives insructions to use new or modified Python scripts made by me, Matleena Myntti, during summer 2011, and also tells differences between my scripts and Simo Seppälä's original scripts. It also functions as some kind of user manual for new users of the scripts, no matter if they knew already Seppälä's scripts or not. More accurate information is provided in example scripts.

I used the scripts for my researches and calculations concerning the effect of the positions of annual rings on the compression effects of a wood piece. Due to that, I needed a bit more and different information than Seppälä.

Many of the scripts mentioned in this document are heavily based on Seppälä's scripts, and some of them are even Seppälä's original, unmodified scripts. Rest of them are some original scripts made by me (and, partially, Juha Koivisto). Most of the scripts in the last group are mainly small calculation, renaming or helping scripts.

2 List of files mentioned in the document

Here is lists of the scripts and files relating to them. The lists don't contain the files that the scripts create.

2.1 Meanings of the symbols

x Scripts

o Non-scripts

– Seppälä's files that I did't modify

\+ Seppälä's files that I modified (often heavily)

* My original files

2.2 File list

2.2.1 Preparation for calculations

x* renameImages.sh

o+ dict.csv

o– CroppingData.tex

o* xDim.tex

2.2.2 Instron data analysis

x+ plotTTMData-2.0.py

2.2.3 DIC

x– compareToFirst-1.0.py

o– default.dicconf

x* xScale.py

x* renameDffs.py

x* doDICVisualization.sh

x+ display_dff_6.0.py

x+ fitMeanStrain.py

x* shearingMap_1.0.py

x* uyData.py

x* aveUyData.py

3 Preparation for calculations

Since I didn't made any notable modifications to the experimental setup, let's assume straightly that we have already Instron data and camera pictures from the experiments.

In my experiments, the names of the experiments are in format *[sample set name, five characters] + [line] + [sample type, R (reference) or F (fatigued)] + [sample number, two-digit number]*. Examples: IB7-1-R14, IB152-F35, IB15a-F03.

Amount of characters in each part is important! If you alter it and you still want to use scripts similarly than I did, you have to modify the code!

3.1 Renaming pictures (if necessary)

When taking pictures, the camera computer saves the pictures in .tiff format with long names containing the name of the experiment, timestamp of the experiment, and very much miscellanous information about the conditions of the experiment. However, there is one difficult issue: sometimes the camera computer may name the files with using as decimal separators commas, not dots, and the other scripts assume that dots are used. Thus, that kind of files must be renamed.

For that, we have script **renameImages.sh** that renames the camera pictures with replacing the commas with dots. It assumes that the pictures are in .tiff format. Run it (with ./renameImages.sh) in the folder in which you have your pictures (they can be in subfolders). The current version of the script contains extra 'echos' in the starts of some commands, to avoid dangers of mistake runnings. Run the script in the current format, and if the output looks good, remove 'echos'.

3.2 Data files in my experiments

I used in my calculations this kind of data files:

1. **dict.csv**, a data file type used originally by Seppälä but I enlargened them notably. Originally, it contained just information of the fatigue times of each sample, but in my version, it contains also dimensions of each sample, data for calculating θ angles (see her candidate report), r_0 distance, comments and so on. dict.csv is put into the same folder which contains plotTTMData-2.0.py script and the Instron data files, and it must be named with name 'dist.csv', unless you change it in other codes. dict.csv contains information of all samples of which you want to create a common result table.

2. **CroppingData.tex**, a data file type that contains data about how to crop samples for DIC, if to rotate them, and how many pictures are used in DIC. It can be named however you want and be located wherever you want, just tell it in the command line when DIC:ing. However, in DIC:ing, all the samples mentioned in the file are DIC:ed with the same command and put into the same folder. The same file type were used by Seppälä, too, and I didn't modify it. CroppingData.tex contains information of all samples you want to put into the same DIC folder.

3. **xDim.tex**, a data file that contains information for scaling the camera picture to right size in graphs and color maps. Practically, there is widths of the samples as pixels and their slanting angles. xDim.tex can be named in any way, but it has to be in fixed location that is written into the other scripts. xDim.tex contains information of ALL samples.

Usually it's easiest to go all pictures through and gather all data at once to different data files. I tended to copy the first picture of each experiment into a separate folder for this task to be better known about which samples were already checked. Also, it helped me to browse the pictures again if I had to get more measurements. Read more from each data file example what data is needed in each file! Also, put in dict.csv the measurements of the samples and comments of them.

4 Instron data analysis

In my calculations, Instron data analysis is heavily based on Seppälä's methods. Main difference, however, is that the analysis also calculates θ angles (and its derivations), tags different samples and adds comments. Unlike Seppälä's original script, my script takes the dimensions of the samples from dict.csv instead of using constants.

For Instron data analysis, you need (all in the same folder)

1. Instron data files
2. dict.csv (of those data files)
3. **plotTTMData-2.0.py**

The analysis is simply run with command python plotTTMData-2.0.py

As result, you will get **interpolated data files** (stress-strain curve data!), **yield.csv** file containing a lot of data of yield points, Young moduli etc. and, in subfolder 'plots', and pdf images of the elastic part linear fit and the plateau area linear fits. Note that if you don't remove yield.csv before running the script again, no new data is saved to the interpolated data files or yield.csv. If you want to tag away those samples whose plateau area isn't neat, look at the latest plots, write 'n' as the 'isTidyCurve' value of bad curves, and run the script again.

It's recommended to save yield.csv as, say, an .odf file to be able to draw graphs etc. easily and thus visualize the results.

5 DIC

5.1 Making .dff:s

Making **deformation function files (.dff:s)** with DIC program doesn't differ from Seppälä's methods at all. Because making .dff:s takes very much time, it's best to put it run in the background at first.

For DIC:ing, you need

1. the camera picture files with correct names (can be in different folders)
2. CroppingData.tex (of those pictures)
3. **compareToFirst-1.0.py**
4. **default.dicconf**
5. **renameDffs.py** (if you want)

First, create a folder (let's call it DICFolder) in which you want to put your .dff:s. Put here default.dicconf file (and renameDffs.py file), and, if needed, change here values according to how you want to make .dff:s. For example, in the line 'pairiterator.name = ...', you can choose either 'First' or 'Previous' according to what kind of comparison you want to make. Also, changing the value in the line 'dic.xtol = ...' changes the accurarity of the DIC:ing: the smaller the number is, the more accurate and time-consuming it is. I have used values 0.05 and 0.001.

Then, run compareToFirst-1.0.py with command python compareToFirst-1.0.py /path/to/CroppingData.tex /another/path/to/DICfolder numberOfSkips

The last number tells how many images are skipped. I have used numbers 9 and 49: in my experiments, camera took 10 pictures per second, so I made .dff:s of every 10th picture (every second) or every 50th picture (every 5th second). How many pictures are analyzed is told in CroppingData.tex

As result, DICFolder gets subfolders with non-skipped copy images, their cropped versions, and .dff files. If you want to rename the .dff files (and, later, pictures made of it) in the way that they contain information of what sample they were taken from etc, run renameDffs.py with command python renameDffs.py */crop_dff/*.dff.

5.2 Getting scales

To make axes of graphs and color maps of the DIC data right-sized, we have to calculate scales. The other parts of the program don't fail if you forget this part, though: it just keeps measurements as pixels. Seppälä used no separate scales but ticked the axes in his pictures with constant ticks.

For scales, you need

1. xDim.tex
2. dict.tex (of all samples)
3. **xScale.py**

Write on the xScale.py the correct path to the wanted dict.csv file and run the script with command `python xScale.py`

As result, you get in the same folder a file named **xResults.csv** which contains data of the scales and errors with measuring width of the sample.

5.3 Visualizing deformation, local strain etc.

.dff files are quite useless if we can't get any other information of them but just lagre matrixes of numbers. So, we needs graphs and color maps. In my scripts, all of them can be created with one command in terminal. Check that you have enough room in your computer, first, pictures take quite a lot room: a bit more than 200 kb per .dff file is probably enough. Seppälä made also average strain graphs and local strain maps (both in the same script), and I enlargened it a bit and divided it into several scripts. This operation takes quite a lot time, though not as much than with .dff creations.

For visualization, you need

1. .dff files you created earlier (in folder named, say, DICFolder)
2. xResults.csv
3. **doDICVisualization.sh**
4. **display_dff_6.0.py** – Creates local strain maps.
5. **fitMeanStrain.py** – Crates average strain graphs with linear fits. Also gathers data about how the values (ie slope) of the linear fit change along time.
6. **shearingMap_1.0.py** – Creates shearing maps.
7. **uyData.py** – Creates data and graphs of average deformations.

Put all scripts to DICFolder and give them permissions to work. Correct paths to xResults.csv file. Then, run doDICVisualization.sh with command `./doDICVisualization.sh`

As result, the script creates **a strain map (.pdf), an average strain curve (.dat and .pdf), a shrearing map (.pdf), and an average deformation curve (.dat and .pdf)**.

5.4 Averages of average deformation curves and average strain of it

Because single samples had usually their own cracks and spikes in their deformation curves – not talking about even more sensitive average strain curves – it was easier to examine θ effect with taking averages of average deformation curves of each angle area (like $\theta_{mm} \in [80°, 90°]$ etc). For that, I created a script that uses earlier created average deformation data (ending ...-uy.dat) and plots average curves of average deformations and the average strain calculated of them. Plotting is made separately to reference and fatigued samples.

For this part, you need

1. -uy.dat files you created earlier in visualization.
2. **aveUyData.py**

Attention! Of which samples the average curves are made depends on in which folders the -uy.dat files are. For example, if you want to make average curves of samples whose θ_{mm} angle is between 80 and 90 degrees, first check which samples have that angle (from yield.csv, for example), move/copy the whole samples folders (ending _skip_number) from DICFolder to another folder (named, say, DICFolder_80-90). Put also aveUyData.py file here and run it with command `python aveUyData.py 'find */*/*-uy.dat'`

As result, the script creates **different average graphs** made of data in -uy.dat files and puts them into am_curves folder which is subfolder of the current folder.

6 Good programs etc. with these scripts

To benefit from these scripts and their outputs better, it could be handy to have/get some knowledge of these programs/languages/etc.:

- XMGrace. Very useful when you want to compare curves (like stress-strain curves, average strain curves, etc.)
- OpenOffice Spreadsheet, for making graps of yield.csv (at least it was easiest for me)
- Python, for modifying the .py scripts
- bash, for modifying the shell (.sh) scripts
- Some Unix commands

The other people in CSM group can help you! :)