EXAM

Software Tools for Computational Materials Science (ST4CMS)

WS2020/21 + SS2021

Date: 26.07.2021

Dr.-Ing. A. Prakash

General Instructions:

- The tasks provided in this document are for the purpose of the exam and for you as a person only. DO NOT distribute them to anybody else!!!
- You may use any resource on the internet to obtain information on the commands to be used for a particular task/subtask.
- You can also use the lecture slides and exercises as a reference.
- You may NOT, however, copy code from any source on the internet!! If any such code is found, it will be treated as malpractice and may lead to exmatriculation from the university.
- Any attempts to copy code of another student will also be treated as malpractice and may lead to exmatriculation from the university.
- Each student has been provided with a different set of data in the form of a zipped file. You have to use the data assigned to you to program the tasks and to test/validate your programs!

Task 1: Bash and Awk - *Combine datasets (simulation snapshots) from multiple cpus to a single large dataset*

Scenario: You have performed a computation on a high-performance computing (HPC) machine. To increase efficiency, the problem domain was decomposed into sub-domains, the response of individual sub-domains being dealt with on separate cpus. Unfortunately, this came with the downside that the master node had to now collect the data from all the cpus in order to write out a snapshot of the simulation for further analysis. Such a collect and output effort reduces efficiency significantly. As a workaround, the data was written out as a snapshot file for each subdomain. The filename is of the following pattern: `Simulation_final.XX.snapshot`, where `XX` denotes the cpu in question.

The format of such a subdomain snapshot file is shown below:

```
#F A 1 1 1 3 3 2
#C number type mass x y z vx vy vz Epot eam_rho
#X      9.9204335000000000e+01 0.0000000000000000e+00 0.0000000000000000e+00
#Y      0.0000000000000000e+00 7.0148058000000006e+01 -3.8546428019485585e-01
#Z      0.0000000000000000e+00 0.0000000000000000e+00 5.7275649000000001e+01
## Generated on Fri Oct 12 11:45:46 2018
## by imd_mpi_eam4point_nve_fnorm_homdef_stress_fbc_nbl_mono_imfd7059 (version of )
#E
1254 1 0.002797 5.036381 6.935170 5.621548 0.505722 0.251326 0.038875 -3.353904 0.994635
6731 1 0.002797 2.475905 6.928626 6.971787 0.137739 -0.799834 -0.131892 -3.355538 0.994800
3477 1 0.002797 1.662030 2.193398 2.763960 -0.104811 -0.857215 -1.765790 -3.432987 0.981619
...
```

The first eight lines are comment lines that provide information on the simulation performed and the data contained in the file. Lines 9 and onwards provide details on the individual material points used in the simulation by the corresponding cpu.

Each dataline contains 11 columns and refer to quantities provided in line 2, i.e., number of the material point, material type, mass of the material point, position of the material point (`x`, `y` and `z` coordinate), velocity of the material point (in `x`, `y` and `z` directions), energy (`Epot`) and a contribution to the constitutive function (`eam_rho`). The size of the quantities is given in the first line: the first three 1's refer to the scalar valued first three columns, the next two 3's refer to vector valued quantities (position and velocity). All further columns are summed up to a single number (here, 2).

For further analysis, it is necessary to have a single file containing data from all the subdomains (or in this case, cpus). Furthermore, data in certain columns are redundant and/or unnecessary are to be removed; data is certain other columns need to be modified.

Your task is to write a bash script that will combine the data from all the files and put it into a single file. Whilst combining, redundant header information must be removed. Furthermore, you will write an AWK script to reduce and modify the data in certain columns of the individual files.

Following are the tasks to be programmed:

Task 1a: Write an AWK script >>>`ReduceColsSimSnapshot.awk`<<< that does the following to the data lines in each file:

- Remove columns 2, 3 and 11 (`type`, `mass` and `eam_rho`)
- The positions are provided currently in Angstroms. Convert the position vector to nanometers. (Use a conversion factor of 0.1)
- Use a conversion factor of 1.6 for the column containing energy (`Epot`)

Task 1b: Write a bash script >>>`ReduceAndCombineSnapshots.bsh`<<< to combine the data from all the files into a single file, whilst ensuring the following:

- The combined data file must contain the header information only once at the top/head of the file. In other words, the header must be copied from a single file. Header information from all other files is to be ignored. (NOTE: All files contain the same header information!)
- The combined data file must include only the columns of interest, i.e., the 2nd, 3rd and 11th columns from the original data file must be removed.
- Since the number of columns has now changed, the header information must be changed appropriately (Hint: Use `AWK` or `sed` for this!). The first and second lines must now read:
```
#F A 1 0 0 3 3 1
#C number x y z vx vy vz Epot
```
- The simulation box size (lines 3, 4 and 5 must also be converted to nm using a conversion factor of 0.1)

The bash script must be general enough to work for any dataset and not just the dataset provided to you as an example. You may, however, assume that any other dataset will also conform to the format provided to you. The prefix of the filename and the number of files is to be received as input on the command line. Executing the following command must accomplish all the tasks listed above:

```
$> ./ReduceAndCombineSnapshots.bsh Simulation_final snapshot 24
```

where `Simulation_final` is the prefix of the datafiles, `snapshot` is the extension of the datafiles and `24` is the number of snapshots. The name of each snapshot file can be now be generated via a loop whose counter goes from 1 to number of snapshots as:
`<prefix>.<counter>.<extension>`

IMPORTANT: *The sanctity of the original data must be maintained. Do not overwrite or edit any of the files. Even the timestamp of the original datafiles must not be changed when the script* `ReduceAndCombineSnapshots.bsh` *is executed.*

The following approach is suggested to solve the problem.

In the bash script:

- Gather the three input variables: `Fileprefix`, `Fileext` and `nsnapshots` denoting the filename prefix, filename extension and number of snapshots, respectively.
- Copy the header information of any one of the snapshot files to a new file: `<Fileprefix>.header`
- Make the expected changes to the header information.
  Hint: Use `AWK` or `sed` to change the first two lines.
  Hint: Use `AWK` to scale the simulation box size to the desired values. Note that `bc` will not be a good choice for this since the data is provided in scientific notation
- Loop over the number of snapshots and perform the following:
  - Obtain the filename of the snapshot under consideration
    Hint: `<Fileprefix>.<counter>.<Fileext>`
  - Remove header information from the file
  - Call the AWK script >>>`ReduceColsSimSnapshot.awk`<<< with the header-less data and modify the dataset as required and append it to a different file. Continue to append to the same file for further values of the loop counter
- Once the above loop has finished, add the modified header information to the top of the file containing the combined (and modified) data. Hint: use `sed` or `cat` for this purpose

NOTE: The `AWK` script >>>`ReduceColsSimSnapshot.awk`<<< is optional. You may execute the corresponding `AWK` command inline inside your bash script!

Your script will be validated later by using a completely different but similar dataset. It is hence important that your script is generic and able to handle any dataset originating from, e.g., a higher number of cpus, or with additional attributes (as individual columns) in the snapshot. The data structure will, however, remain similar to the provided to you, i.e., have similar number of header lines and header information.

Task2: Python – Program to plot the Mohr's circle of stresses

Write a Python program to compute and plot the Mohr's circle of stresses. Plot the Mohr's circle for the stress states of individual material points and the average of all material points.

Theory: Mohr's circle of stresses is a two-dimensional graphical representation of the stress tensor and is named after the German engineer *Christian Otto Mohr*. Any point on the circle denotes the stress state on a rotated coordinate system. The Mohr's circle is plotted with the normal stresses along the abscissa and the shear stresses along the ordinate (see fig. 1 below).

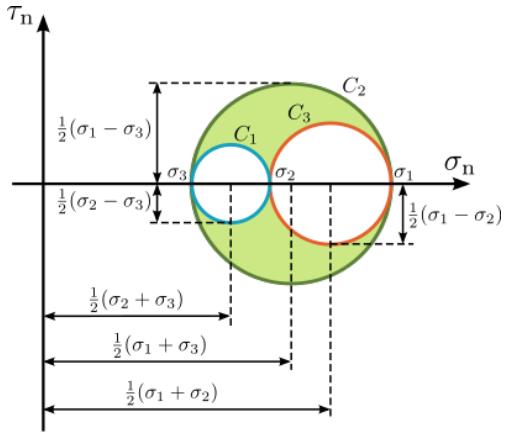


Fig. 1: Schematic representation of the Mohr's circle. Source: Wikipedia.

The centers and radii of the three circles are given by the equations:

$$C1 = \frac{1}{2}(\sigma_2 + \sigma_3); \quad R1 = \frac{1}{2}(\sigma_2 - \sigma_3)$$

$$C2 = \frac{1}{2}(\sigma_1 + \sigma_3); \quad R2 = \frac{1}{2}(\sigma_1 - \sigma_3)$$

$$C3 = \frac{1}{2}(\sigma_1 + \sigma_2); \quad R3 = \frac{1}{2}(\sigma_1 - \sigma_2)$$

where σ_1, σ_2, σ_3 are the three eigenvalues of the stress tensor.

Details of the dataset:

The dataset contains xx number of files. Each file provides the stress state (as the complete tensor, in MPa) and the volume of a material point. The data in the file is organized as follows:

```
# Comment line containing the volume of the material point
Stress_xx   Stress_xy   Stress_xz
Stress_xy   Stress_yy   Stress_yz
Stress_xz   Stress_yz   Stress_zz
```

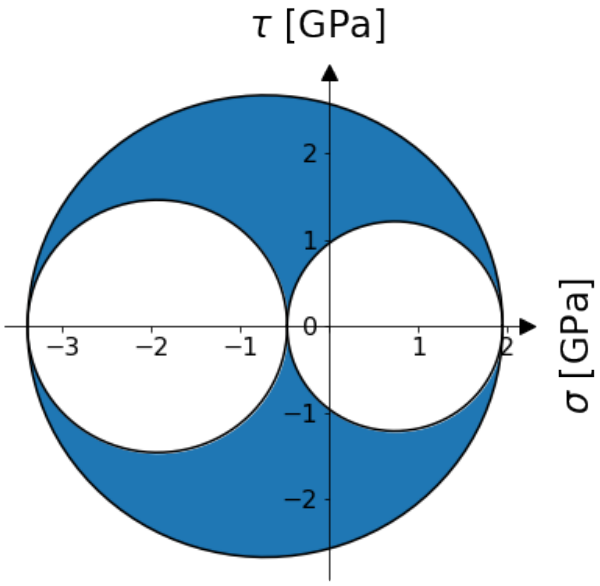
Your task is to write a Python program that plots the Mohr's circle for any given stress state. To demonstrate the working of your program, you will then generate the Mohr's circle for all the ten stress states provided to you. Furthermore, you will generate the Mohr's circle for the stress tensor obtained as the average of all the ten material points; the average being performed both with and without accounting for the volume of the material point

The Python program must include the following functions:

- Function `readData(fname)`: To read in data from a single file
  `fname` → Name of the file from which data is to be read
  This function should return the stress tensor as well as the volume of the material point (stored in the first line)
- Function `AvgStress(n,SigArr,VolumeArr)`: To average stresses
  `n` → Number of points to average. NOTE: Although `n` is 10 in the current scenario (`n` is the number of material points), the function must be general. Hence use the value provided to the function and do not hard code `n`!
  `SigArr` → An array of stress tensors that are to be averaged
  `VolumeArr` → An array of material point volumes of the corresponding stress tensors
  This function should perform a volumetric average of *n* stress tensors as follows:

$$\bar{\sigma} = \frac{\sum_{i=1}^{n} \sigma_i V_i}{\sum_{i_1}^{n} V_i}$$

  where σ_i and V_i denote the stress and volume, respectively, of the i^{th} material point.
- Function `EigVal(Sigma)`: To compute eigenvalues of a given stress tensor, `Sigma`. The function must return the sorted eigenvalues. You may use available functionality in numpy to calculate the eigenvalues.
- Func `plotMohrsCircle(SigmaEigval, foutNamePNG)`: To plot the Mohr's circle(s) of stresses for the given eigenValues.
  `SigmaEigval` → Eigenvalues of the stress tensor
  `foutNamePNG` → filename for the output PNG file
  In this function, you should calculate the center and radius of all three circles. Plot the Mohr's circle(s) and shade the different regions as shown in the figure below. Note that the circumference of the circle must be clearly demarcated.



- Annotate the plot appropriately, i.e., ensure proper axes labels (with units) and ticks with the right fontsize.
- Save the plot as a PNG file. An example pattern of the PNG filename can be: `MohrCirc_<PrefixStringOfDataset>.PNG`, where `<PrefixStringOfDataset>` is the file name of the corresponding dataset without the extension.
  Example: `Filename - Gr122_MatPt10.dat`
  `<PrefixStringOfDataset> - Gr122_MatPt10;`
  `PNG filename – MohrCirc_Gr122_MatPt10.PNG`
- For the average stress tensor, use only the grain number appended with `_VolAvg`.
  Example: `Filename - Gr122_MatPt10.dat;`
  `PNG filename – MohrCirc_Gr122_VolAvg.PNG`

NOTE: You may choose the name of the PNG file. It is important that all required information is present in the filename, i.e., Grain number, material point number or the averaging procedure.

The main program must deal with the following:

- Obtain the prefix string of the filename(s) containing the dataset, and the number of material points as user input.
  Hint: You may use `sys.argv` for this purpose.
- Read the stress tensors and volumes of all material points using the function `readData`. How you choose to store this dataset is left to your discretion. You may store it as an array (i.e. array of arrays) or as a dictionary or simply as a list.
- Obtain the average stress tensor. Here, perform averaging with and without the material point volume, i.e. in the latter case choose an identical volume for all material points. Ensure that your function `AvgStress` works not only for the entire array of stress tensors, but also for a subset of it (choose a value of n which is lesser than 10!).
- Obtain the sorted eigenvalues of individual stress tensors and the average stress tensor(s) by calling the function `EigVal`.
- Plot the Mohr's circle for individual stress tensors and the average stress tensor(s) by calling the function `plotMohrsCircle`. Make sure that you are passing the right output filenames for each case!

Some hints:

- To plot the circle, there are two possible methods:
  - `plt.Circle(center, radius)`
  - `plt.plot (xx,yy)`

  The former requires you to add a patch via `ax.add_patch`. The latter is a simple plot that we have also done in the lecture/exercise, with `xx`, `yy` the arrays of `x` and `y` coordinates of points on the circumference on the circle (Divide the circle into segments and calculate the points on the circumference).
- Scale the stress values from MPa to GPa. This will make your axis ticks easier to read!

Execution:

The Python program should be executable as follows:

```
$> python MohrCirc.py <FilePrefixStr> <MatPtStr> <FileExtension> <NumMaterialPoints>
```

An example of a specific command could be:

```
$> python MohrCirc.py Gr1 MatPt dat 10
```

Further points to keep in mind:

- Ensure you are using the right environment by activating your virtual environment.
- You may choose to program your tasks in a Jupyter notebook or in the ipython interpreter. However, your final submission must by a python script that is directly executable from the command prompt.
- Whilst programming, make sure that your code is well commented. Put in appropriate comments wherever necessary, particularly where you need to explain the logic of your program. Avoid trivial comments like: “Here we add two numbers”.
- Remember the basic principle of programming – You program a certain logic so as to use it repeatedly. To this end, your program must be generic and usable with other datasets also. Adhering to the given design will help you in this regard.

Final submission:

Once you are done with your programming tasks:

- Copy all your programmed scripts into the main folder of the exam (ST4CMS_Exam_SS2021_DatasetXX), where XX is the number of the dataset
- Prepare a `tar`ball archive and `gzip` it
- Rename the file as follows: <Matr.Nr.>_<Name>_ST4CMS_Exam_SS2021_DatasetXX
- Upload it to the task >>>Exam tasks for SS2021<<<. You should receive an email confirmation upon successful submission.
- **NOTE**: Each person has a limit of 100MB on the file to be submitted.