CheasePy 2.0 User Guide

Ehab Hassan

October 3, 2019

CheasePy is code written in Python to run the CHEASE (*Cubic Hermite Element Axisymmetric Static Equilibrium*) code, which solves the Grad-Shafranov equation for toroidal MHD equilibria using pressure and current profiles and fixed plasma boundaries that is defined by a set of experimental data points (R,Z). The CheasePy code allows an iterative running of the CHEASE code either to check the preservation of MHD equilibria or converging to an experimentally defined total toroidal plasma current by modifying any input quantity.

1 Toroidal MHD Equilibrium

The MHD equilibrium equations are given by:

$$\boldsymbol{J} \times \boldsymbol{B} = \boldsymbol{\nabla} p \tag{1}$$

$$\boldsymbol{\nabla} \times \boldsymbol{B} = \boldsymbol{J} \tag{2}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0 \tag{3}$$

The magnetic field can be represented as:

$$\mathbf{B} = T\boldsymbol{\nabla} + \boldsymbol{\nabla}\phi \times \boldsymbol{\nabla}\Psi \tag{4}$$

where ϕ is the ignorable toroidal angle and Ψ is the poloidal magnetic flux function.

For static MHD equilibria, the Grad-Shafranov equation is given by:

$$\boldsymbol{\nabla} \cdot \frac{1}{R^2}\boldsymbol{\nabla}\Psi = \frac{j_\phi}{R} = -p'(\Psi) - \frac{1}{R^2}TT'(\Psi) \tag{5}$$

where j_ϕ denotes the toroidal plasma current density, R the major radius of the torus. The nature of the equilibria is determined by the two free functions $p'(\Psi)$ and $TT'(\Psi)$, where the pressure p and the poloidal current flux function T are functions of Ψ only.

The total current everywhere inside the plasma ($\Psi < 0$) is positive and is given by:

$$I = \int j_\phi dS = \int j_\phi \left(J/R\right) d\Psi d\chi \tag{6}$$

where J is the Jacobian.

To achieve MHD equilibrium CHEASE can treat different current profiles such as:

- The poloidal current flux:

$$TT'(\Psi) \tag{7}$$

- The surface averaged current density:

$$I^*(\Psi) = \frac{\oint j_\phi (J/R) d\chi}{\oint (J/R) d\chi} = -R_0^2 \frac{C_1}{C_0} p'(\Psi) - R_0^2 \frac{C_2}{C_0} \frac{TT'(\Psi)}{\mu_0} \tag{8}$$

- The averaged parallel current:

$$I_\parallel = \frac{\oint \boldsymbol{J} \cdot \boldsymbol{B} J d\chi}{\oint \boldsymbol{B} \cdot \boldsymbol{\nabla} \phi J d\chi} = R_0 \frac{\langle \boldsymbol{J} \cdot \boldsymbol{B} \rangle}{\langle T/R^2 \rangle} \tag{9}$$

$$I_\parallel = \frac{\oint \boldsymbol{J} \cdot \boldsymbol{B} J d\chi}{\oint \boldsymbol{B} \cdot \boldsymbol{\nabla} \phi J d\chi} = -R_0 \frac{C_1}{C_2} p'(\Psi) - R_0 \frac{TT'(\Psi)}{\mu_0}(\Psi) \left(1 + \frac{1}{T^2(\Psi)} \frac{C_3}{C_2}\right) \tag{10}$$

- The averaged parallel current density:

$$J_\parallel = \frac{\langle \boldsymbol{J} \cdot \boldsymbol{B} \rangle}{B_0} \tag{11}$$

where,

$$\{C_0, C_1, C_2, C_3\} = \oint \left\{ \frac{1}{R}, 1, \frac{1}{R^2}, \frac{|\nabla \Psi|^2}{R^2} \right\} J d\chi \tag{12}$$

$$\langle \boldsymbol{J} \cdot \boldsymbol{B} \rangle = -T(\Psi) p'(\Psi) - T'(\Psi) \frac{\langle B^2 \rangle}{\mu_0} \tag{13}$$

$$\langle B^2 \rangle = \frac{\oint B^2 J d\chi}{\oint J d\chi} \tag{14}$$

$$\langle T/R^2 \rangle = \frac{\oint T/R^2 J d\chi}{\oint J d\chi} \tag{15}$$

The toroidal current density can be expressed as:

$$j_\phi = \frac{1}{R} \left(\frac{C_0}{C_2} I^*(\Psi) + \left(\frac{C_1}{C_2} - R^2 \right) p'(\Psi) \right) \tag{16}$$

$$j_\phi = \frac{1}{yR} \left(I_\parallel(\Psi) + \left(\frac{C_1}{C_2} - yR^2 \right) p'(\Psi) \right) \tag{17}$$

where

$$y = 1 + \frac{1}{T^2(\Psi)} \frac{C_3}{C_2} \tag{18}$$

The normalization used here has the following format:

$$\begin{aligned}
&\Psi = B_0 R_0^2 \\
&I = \frac{B_0 R_0}{\mu_0} \qquad \textit{and} \qquad J = \frac{B_0}{\mu_0 R_0} \\
&p = \frac{B_0^2}{\mu_0} \qquad \textit{and} \qquad p' = \frac{B_0}{\mu_0 R_0^2} \\
&T = B_0 R_0 \qquad \textit{and} \qquad T' = \frac{1}{R_0} \qquad \textit{and} \qquad TT' = B_0
\end{aligned}$$

2 CHEASE Code

When CHEASE code runs based on experimental data (which is what CheasePy designed for) it takes as an input three files. The *EXPEQ* file (which is a reduced version of the EFIT file, also known as EQDSK file) which has the experimental equilibrium such as the current profile, in addition to the magnetic configuration. The second file is the *EXPTNZ* file which contains profiles for the ion and electron temperatures (T_i & T_e) and densities (n_i & n_e), in addition to the effective atomic number (Z_{eff}) as a function of the normalized poloidal axis (ψ_N). The CHEASE code also takes the *chease_namelist* file which has the initialization parameters that depend on the plasma fusion machine shot, the type of the input physical quantities, and the run-mode of the CHEASE code. For more information about CHEASE code and the different options for the input parameters in the *chease_namelist* file you may go to their website which has all the required resources and references.

After a successful run, the CHEASE code produces several output files, but CheasePy code uses only *four* of them. The *EXPEQ.OUT(IN)* output files have specific experimental quantities from the EFIT file depending on the selected values of the input parameters in the *chease_namelist* file, such as pressure (P), poloidal or toroidal coordinate grid points ($\rho(\psi)$ or $\rho(\phi)$), parallel current density profile ($J_{\parallel}$), etc. *EXPTNZ.OUT(IN)* output file should have exactly the same profiles in the *EXPTNZ* input file but with a resolution matches what is given in *chease_namelist*. The CHEASE code also pack several output quantities related to the coordinate systems and physical quantities into an HDF5 file called *ogyropsi_iterxxx.h5*, and it should be noticed here that all the quantities in the HDF5 file have SI units in contrary to the normalized quantities in the *EXPEQ.XXX* and *EXPTNZ.XXX* files. CHEASE code also gives a text file as an output that contains all the outcomes similar to the HDF5 file named based on the iteration number to be *iterxxx.OUT*. The CHEASE code produces also several other output files but they aren't needed by CHEASEPY.

For calculating the MHD equilibrium in the zeroth iteration the CHEASE code uses the EXPEQ file that may have the EFIT or EQDSK format which contains an extensive amount of information about the machine geometry, magnetic configuration, and plasma equlibirum. However CHEASE also accepts a reduced version of the EQDSK file which contains only the required data for the next iteration to run properly. Then, the format of the EXPEQ file doesn't change in the consecutive iterations. In contrary, the format of the EXPTNZ doesn't change from the zeroth iteration to the end of the simulation run, it has the same profiles but with a different resolution as mentioned above. Some parameters in the *chease_namelist* file need to be changed from one iteration to another, but most of them have their values unchanged. For example, the *NEQDSK* parameter takes 1 when using the EFIT file in EQDSK format, or it takes 0 to create the EXPEQ from input/given parameters.

We will shed more lights on the description and function of the parameters needed to be modified to properly run the CheasePy code in other sections, but it worth mentioning here that the provided *chease_namelist* file with the CheasePy code has only the parameters that you may need to change based on the case you run such as pressure type (nppfun), current type (nsttp), coordinates resolution (NPSI,NCHI,NS,NT,NISO). Other parameters which are not provided in the basic *chease_namelist* file are either set by the CheasePy code in the *namelistcreate()* function as they barely change for most of well-known cases, or they got set by the CHEASE code itself. You may also add these parameters to the *chease_namelist* file in case you will change them in a regular basis, or you need to see the effect of changing these parameters on the results of the simulation.

3 CheasePy Code

To run the CheasePy code you need to have the CHEASE code compiled first in your local machine as instructed in the CHEASE developer website and have the executable file named *chease_hdf5* in the same directory of the *cheasepy.py, runcchease.py* and *parameter_namelist* files (you may have the $PATH is set/export to the *cheasepy.py* script in case you do not have it in the same directory along with the *chease_hdf5* executable.) Also, you need to have Anaconda for Python 2.7/3.7 version installed in your local machine, where the *cheasepy.py* script works with both versions of Python (2.7 or 3.7).

3.1 Directory Structure

In the main directory there are three files: *runchease.py*, *cheasepy.py*, and *chease_hdf5* with the first one runs CheasePy, the second one has a modular structure that contains all the methods and variables used to create files, run the simulation, manipulate data, and plot outputs. The third file is the executable of the CHEASE code as described above. In addition, there is the ***shots*** folder which contains the available experimental shot records each in its own folder. Each shot folder is named after the shot name and contains *at least* three files: *machine.name_shot.Number_EQDSK* which has the EFIT (or EQDSK) file format, *machine.name_shot.number_EXPTNZ* which contains the electron and ions temperature and density profiles, and *chease_parameters.csv* which contains the parameters required by the CHEASE code to do the run for the chosen shot and get the proper MHD equilibria. You may also include the profiles in either (p-????) EFIT equilibrium profiles, or iterdb file, where CheasePy will convert them to the proper EXPTNZ format. Also, you may have EXPEQ file that contains ONLY the required physical quantities for CHEASE to run a specific case instead of the EQDSK extended file. But you need to keep the name of these files has the same format as we explained above, i.e. *machine.name_shot.Number_file.type*, e.g. DIIID_162940_EXPEQ, DIIID_162940_ITERDB, or DIIID_162940_PROFILES, with the folder that contains these files is named DIIID_162940.

You may add more shots to the shot folder by following the same method of naming the shot folder and the three files. Having the wrong name for the shot folder and/or its content might cause *CheasePy* code doesn't work. Although it sounds uncommon to put a restriction on the naming of the folders and files, this reduces lots of effort the user of CheasePy might to put in the future.

3.2 Creating chease_namelist File

The CHEASE code runs iteratively based on the parameters in the CSV file which contains at least two columns. The first column has the required parameters for the zero iteration which uses an EXPEQ file in the reduced or the extended EFIT format, and the next iterations use the reduced format of the EXPEQ file. The CHEASE code keeps using the given parameters in the last column for all iterations left in case the number of columns are less than the number of iterations. For example, if you have five columns in the CSV file and you need to do 20 iterations in addition to the zero iteration, the first column parameters are used with the zero iteration, the second column for the first iteration, the third column for the second iteration, the fourth column for the third iteration, and the fifth column for the other 17 iterations.

The *CheasePy* code starts with copying the required shot files from the shot folder - as we explain below, to the main directory, and then creates the *chease_namelist* file (which is required for CHEASE code to run) using the *namelistcreate(csvfn,rec,setParam={})* function that takes three arguments. The first argument is the path to the CSV file, the second argument is the iteration number - starting with zero for the zero iteration, and the third argument has a Python dictionary structure and it is used to force values for a list of selected parameters those are given or not given in the CSV file (i.e. passing a value for a specific parameter in the *setParam* argument of the *namelistcreat()* function overrides any other value either provided in the CSV file of set as a default by CHEASE code itself). *CheasePy* repeats these steps every iteration by reading the parameters in the column corresponds to the current iteration and creates the *chease_namelist* file using the *namelistcreate()* function. It worth notice here that if the *namelistcreate* function doesn't receive certain parameters from the *chease_parameters.csv* or from *setParam* argument it sets the most common value used for those parameters.

3.3 Creating EXPEQ and EXPTNZ Files

After each iteration CHEASEPY renames the output files to reflect the iteration number by adding a suffix *_iterxxx*, e.g. *ogyropsi_iter000.h5*, *EXPEQ_iter000.OUT*, and *EXPTNZ_iter000.OUT* for the zeroth iteration. Hence, you have several options to prepare the input files for next iterations by creating the *EXPEQ* and *EXPTNZ* files either from the *EXPEQ_iterxxx.OUT* and *EXPTNZ_iterxxx.OUT* from the previous run, respectively, or from the *ogyropsi_iterxxx.h5* file after employing the proper normalization discussed above. You may also use several sources to create the *EXPEQ* file by setting the right data source for each set of parameters. For example you may use the current flux profiles (e.g. $I_{\parallel}, J_{\parallel}$) from the *EXPEQ_iterxxx.OUT* of the previous iteration while using the pressure (P) or pressure-gradient (P') from the *EXPTNZ_iterxxx.OUT* file, and using the poloidal coordinate ($\rho(\Psi)$) from the *ogyropsi_iterxxx.h5* file of the previous iteration by setting the data sources as current_src=2 or 'expeq', pressure_src=3 or 'exptnz', and rhopsi_src=0 or 'chease', respectively. This allows the user to change the input temperature and density profiles for the ions and electrons and starts the next iteration with a new set of the available parameters and profiles to check the effect of these changes on the MHD equilibria.

Before creating the *EXPEQ* and *EXPTNZ*, the *CheasePy* code uses *read_chease(cheasefpath)* function which takes the path to the *ogyropsi_iterxxx.h5* file to extract the profiles and parameters from this HDF5 file, calculating the required physical quantities and parameters, and employing the required normalization. Similarly, the functions *read_exptnz(exptnzfpath)* and *read_expeq(expeqfpath)* takes the path of the EXPTNZ and EXPEQ files, respectively, and return the normalized physical profiles, physical quantities, and coordinate geometries. All three functions return a Python dictionary structures that contains their contents. In addition to the path, these read functions may take two other arguments *setParam* and ***kwargs* which are used to control the outputs of these functions. The *setParam* argument passes some *control parameters* to the *read* function to override the way the function works in its default case. For instance, if the *setParam* has the control parameter *nrhomesh* it tells the function to have all the return physical quantities as functions of the poloidal coordinate (ρ_{ψ_N}) if *nrhomesh* equals 0 or *rhopsi*, or the toroidal coordinate (ρ_{ϕ_N}) if *nrhomesh* equals 1 or *rhotor*. The third argument ***awkeys* is used to pass the path of the EQDSK or CHEASE files to the read function to provide a way to interpolate all the physical quantities over a unified rhopsi grid or rhotor grid that is provided by EQDSK or CHEASE which are passed as input to the function.

CHEASEPY uses the function *write_exptnz(setParam={},**kwargs)* to create the EXPTNZ file based on the *control parameters* passed to the *setParam* argument to specify the sources of the physical quantities and the grid type. The arguments passed to the ***kwargs* are the path to the chease source file, *ogyropsi_iterxxxx.h5*, and/or the exptnz source file, *EXPTNZ_iterxxxx.OUT*. For instance, *setParam{eprofile_src}='exptnz'* tells the *write_exptnz* to take the electron density and temperature profiles from the *EXPTNZ_iterxxxx.OUT* file, however *setParam{iprofile_src}='chease'* tells the *write_exptnz* to take the ion density and temperature profiles from the *ogyropsi_iterxxxx.h5* file. Similarly, CHEASEPY uses the function *write_expeq(setParam,**kwargs)* to create the EXPEQ file using the parameters in *ogyropsi_iterxxxx.h5*, *EXPEQ_iterxxxx.OUT*, and *EXPTNZ_iterxxxx.OUT* files which their paths are given in the ***kwargs* argument and the list of sources are provided in the *setParam* argument. The *write_exptnz()* function can also take ITERDB and PROFILES as sources for the profiles, where *CheasePy* converts them to the right format and interpolate the physical quantities to the used grid in the EXPTNZ file.

On the other hand, the *write_expeq()* function can calculate the pressure from all the sources those have the density and temperature for ions, electrons, and impurities, such as *"chease" = 0, "eqdsk" = 1, "expeq" = 2, "exptnz" = 3, "profiles" = 4, and/or "iterdb" = 5*, where the user can use either the text of numbers alternatively in the *setParam* argument for any read/write function in the CheasePy code. In a similar way, the *control parameters* in the *setParam* dictionary-like input to the *write_exptnz()* and *write_expeq()* functions are used to specify the type (ρ_ψ or ρ_{phi}) and source (CHEASE or EQDSK) of the grid based on the provided files in the ***kwargs* argument.[1]

3.4 Plotting the Outputs

When *CheasePy* code is done with all the required iterations it plots some fields such as the density and temperature profiles, the current flux profile, the magnetic field, the toroidal current, etc, with all the plotted figures packed together in one PDF file. You can also plot the fields in the outputs of a previous run of *CheasePy* directly without rerun the code by picking the right selection as we explained below. It is easy to add plots for other fields those are not plotted by copying the chunk of code used to plot another field and replace the name of the existing field with the new field name. You need also to rename the figure object and add it to the PDF file. This all can be done in the *cheaseplotter(OSPATH,skipfigs=1)* function which takes two argument; the *OSPATH* which specifies the local path for the HDF5 files and the *skipfigs* which specifies the number of skipped figures with a default value 1. The latter argument is used when you have a large number of outputs and you need to skip some of them not to make the plot very crowded. The *cheaseplotter()* function searches for all files of name format like *ogyropsi_iterxxx.h5* and parse it to extract the all fields and coordinates and plot the selected fields.

3.5 Running CheasePy

The user can run *CheasePy* code by typing any of the following commands in the main directory:

python runchease.py -m (manual)
python runchease.py -s (submit)

[1] If the source of the grid *rhomesh_src* is set to None no interpolation takes place, and the given grid is used for all physical quantities in either EXPTNZ or EXPEQ.

The (*-m*) allows the user to choose the operation mode parameters *step-by-step*, however the (*-s*) allows the user to setting the operation mode parameters in the *runchease.py* script and then *submit* the run directly to the command console without expecting any further actions. Below, we describe the steps for setting up the operation mode parameters manually and this applies to the submission case with all control parameters are given in the *runchease.py* script file.

Once you execute the command *python runchease.py -m* you will get a list to select from what you need to do with *CheasePy* code which has the following selections:

```
Select on of the following options:
(1) Run the Code and Plot the Outputs.
(2) Plot the Available Outputs.
(3) Remove Input/Output Files.
(4) Exit.
Selected Option:
```

Select 1 if you need to run *CheasePy* and plot the outputs, select 2 if you want only to plot the outputs of a previous run (you have to have the output files in the main directory), select 3 if you want to remove all the files from a previous run, and select 4 if you want to exit the run without doing anything[2].

All selections except the the selection 1 will return without any further question, however if you picked selection 1 you get another list of selections to choose between:

```
Select CHEASE running mode:
(1) Check Equilibrium Preservation Over Multiple Iterations.
(2) Converge to Total Current by correcting Current.
(3) Converge to Total Current by correcting Pressure.
Selected Option:
```

In this list of selections you can select between running the *CheasePy* code to check the preservation of the MHD equilibria (select 1), to converge the total toroidal current calculated inside the code to the total plasma current given in the EQDSK file by correcting the parallel current (select 2), or to converge the total toroidal current calculated inside the code to the total plasma current given in the EQDSK file by correcting the plasma pressure (select 3). By choosing the first selection, the *CheasePy* code will use the outputs of every iteration as an input to the next iteration without any manipulation/modification of these outputs. However, choosing the second selection will force the code to calculate the relative error between the calculated total toroidal plasma current and experimentally given total plasma current and use it to modify the Ohmic current part of any of the parallel currents ($I_{\|}$ or $J_{\|}$). Then create the new EXPEQ file using the modified profile of the parallel current to find the new total toroidal current and compare it again to the given total plasma current. It repeats this process up to a specific given tolerance of the relative error (default: 10^{-7}). Similar operation will be done when choosing the third selection but by modifying the plasma pressure and reconstruct the equilibrium.

The function *current_correction(chease,expeq={},setParam={})* is used to implement the correction in the parallel current the way defined by the user, and returns the correction

[2]If you select (3) to remove all the files from a previous run the same menu will appear again to select how you like to proceed after removing the files.

to the *write_expeq()* function to use it in constructing the expeq file that will be used as an input in the next iteration. The *current_correction()* can be found at top of the cheasepy.py module for the user to modify the way he likes to define the implementation of the correction. It takes the path to the *chease data*, the path to the *expeq data*, and a list of operational parameters in the *setParam* including the total current from the EQDSK file to use it to estimate the corrected parallel current.[3]

Picking any of these selections, a questions popup in the screen asking you if you want to remove the files of the previous run:

```
Remove output (h5, pdf, dat, OUT) files of previous runs (yes/no)?
```

Answering this question with (**y or yes**) makes the code removes all the files from the previous run and show you a list of the files in the main directory.

```
List of Available CHEASE Files:
runchease.py    chease_hdf5    shots
```

Then it shows you a list of the available shots to choose one of them to be run by CHEASE code:

```
List of the available shots:
(01) CASE1_20100624
(02) DIIID_EFITD_163470
(03) DIIID_EFITD_163525
(04) TCV_49541_15
Select Shot Number:
```

Once you select any of the shots the *CheasePy* code copies the files from the shot folder that matches the selected shot to the main directory and return with a list of the available files in the current (main) directory:[4]

```
Chease runs the CASE1_20100624 shot:
CASE1_20100624_EXPEQ  CASE1_20100624_EXPTNZ  chease_namelist
chease_parameters.csv runchease.py shots
```

Checking the files in the current folder shows the *chease_namelist* file which was created right after copying the files from the shot folder.

Finally, the *CheasePy* asks you if you want to start the run:

```
Do you want to continue? (yes/no)?
```

With a (**y or yes**) answer the *CheasePy* code starts the iterative calling of the *CHEASE* code to run with the given parameters in the *chease_parameters.csv* file and source of data that are used to create the EXPEQ and EXPTNZ files.

[3]The default method for correcting the parallel current is suggested in the *current_correction()* and the user can use it as guidance for calculating the correction.

[4]When you run the code manually the code searches the target directory of the shot for a profile file in the following order, PROFILES, EXPTNZ, then ITERDB. If it doesn't find any of them it returns an error. For the equilibrium the code searches for the EQDSK file first either with NEQDSK = 0 or 1 in the chease_parameter input file. However, if it doesn't find EQDSK file it searches for the EXPEQ file and it returns an error if it doesn't find either of them.

When you run the code using the submission option (*python runchease.py -s*) you need to setup the sources in the *runchease.py* file as input values to the **srcVals** Python 'dictionary' defined in that script file. For example, to set the source for the electron density and temperature profiles as iterdb, you should use: srcVals['eprofiles_src'] = ['iterdb'].[5] If the source of the electron density profile will change after the zero iteration to be 'chease,' the **srcVals** should be set as follow: srcVals['eprofiles_src'] = ['iterdb','chease']. Same implementation should be applied on all sources of profiles; 'iprofiles_src', 'pressure_src', 'current_src', and 'rhomesh_src', where the source of the rhomesh will be unified over all profiles in both the EXPTNZ and EXPEQ files.

In case the submitted running code halt for any reason before finishing the required number of iterations, you may rerun the code with setting the remove_inputs and remove_outputs to False or *'no'* to keep the outputs from the previous run, hence the code will test if there is any files from a previous run and create the EXPEQ and EXPTNZ from the files of higher iteration number and continue from there.

4 Future Plans

CHEASEPY code is still under development to add new features to the code. But this version of the code is stable and supposed to give no critical issues. In case a user has any concern or need to add more features to the code you may contact Ehab Hassan at ehab@utexas.edu. Otherwise, users can add these features themselves but they supposed to inform the original developers with these added features.

You can find the latest version of CHEASEPY code on github, and we recommend that you pull directly from the Master branch and keep your version up to date with any changes we implement to the code.

5 References

The CHEASE code for toroidal MHD equilibria, H. LUTJENS, A. BONDESON, O. SAUTER, Computer Physics Communications (1995).

[5]The definition of the source MUST be in a Python 'list' format.