

# Quickstart

Once installation is complete you can start running Sarkas. This quickstart guide will walk you through a simple example in order to check that everything is running smoothly.

The YAML input file can be found at [input\\_file](#) and this notebook at [notebook](#)

---

## Simulation

In Jupyter notebook you can run the following commands

```
In [1]: # Import the usual libraries
%pylab
%matplotlib inline
import os

plt.style.use('MSUstyle')

# Import sarkas
from sarkas.processes import Simulation, PostProcess

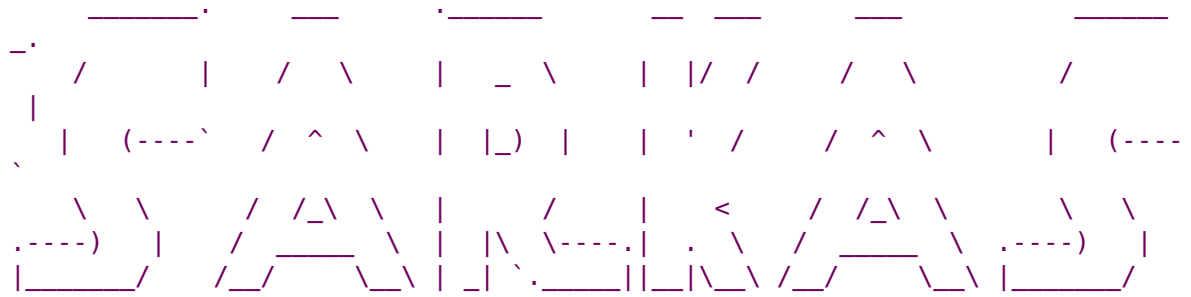
# Create the file path to the YAML input file
input_file_name = '/home/admin17/anaconda3/envs/sarkas/Testing/yocp_quicksta
```

```
Using matplotlib backend: <object object at 0x7f75ec1f1c00>
Populating the interactive namespace from numpy and matplotlib
```

The above commands imported the required libraries and define the file path to our input file.

Let's now run the simulation

```
In [2]: # Initialize the Simulation class
sim = Simulation(input_file_name)
# Setup the simulation's parameters
sim.setup(read_yaml=True)
# Run the simulation
sim.run()
```



An open-source pure-python molecular dynamics suite for non-ideal plasmas.



```

-----
AttributeError                                Traceback (most recent call last)
/tmp/ipykernel_781792/3566869978.py in <module>
      2 sim = Simulation(input_file_name)
      3 # Setup the simulation's parameters
----> 4 sim.setup(read_yaml=True)
      5 # Run the simulation
      6 sim.run()

~/anaconda3/envs/sarkas/lib/python3.7/site-packages/sarkas/processes.py in
setup(self, read_yaml, other_inputs)
    324
    325     else:
--> 326         self.initialization()
    327
    328     if self.parameters.plot_style:

~/anaconda3/envs/sarkas/lib/python3.7/site-packages/sarkas/processes.py in
initialization(self)
    207         self.parameters.cutoff_radius = self.potential.rc
    208
--> 209         self.thermostat.setup(self.parameters)
    210         self.integrator.setup(self.parameters, self.thermostat, sel
f.potential)
    211         self.particles.setup(self.parameters, self.species)

~/anaconda3/envs/sarkas/lib/python3.7/site-packages/sarkas/time_evolution/t
hermostats.py in setup(self, params)
    114
    115     # Check whether you input temperatures in eV or K
--> 116     self.type = self.type.lower()
    117
    118     if self.eV_temp_flag:

AttributeError: 'NoneType' object has no attribute 'lower'

```

## Postprocessing

Now that our simulation is complete we need to check if the simulation was physically sound. Run the following three lines will initialize the `PostProcess` class and calculate the observables defined in the input file.

It will also produce a plot of the Temperature and Total Energy of the Production phase.

```

In [ ]: # Initialize the Postprocessing class
postproc = PostProcess(input_file_name)
# Read the simulation's parameters and assign attributes
postproc.setup(read_yaml=True)
# Calculate observables
postproc.run()

```

You will notice that both the energy and temperature oscillates wildly. This is fine as long as the percentage deviations, in the top plots, are small. You should have a temperature

deviations between -2% to ~ 4-5% while energy deviations between -2% and 1%.

---

## Observables

The most common observable is the radial distribution function. This was calculated by `postproc.run()`, here we plot it by rescaling the x axis by the Wigner-Seitz radius  $a_{ws}$ .

```
In [ ]: # Initialize the Pair Distribution Function class
ax = postproc.rdf.plot(
    scaling=postproc.parameters.a_ws,
    y = ["C-C RDF", "Mean"],
    xlabel = r'$r/a_{\rm ws}$',
    ylabel = r'$g(r)$'
)
ax.legend(["C-C RDF"])
```

Things to check in here are:

- Does  $g(r)$  go to 1 for large  $r$  values ?
- Is there a peak at  $r \sim 1.5a$  ?
- Is the height of this peak about  $\sim 1.4$ ?

If the answer to all these question is yes than the simulation was successfull.