Embedding with eDFTpy

Team Embedding

Department of Chemistry & Department of Physics
Rutgers, the State University of New Jersey, Newark, NJ

Wednesday October 13th 2021

Install eDFTpy

Requirements

- Python 3.6 or newer
- NumPy 1.18 or newer
- SciPy 0.18 or newer
- DFTpy latest
- ASE 3.21.1 or newer
- pylibxc
- mpi4py 3.0.2 or newer
- mpi4py-fft 2.0.4 or newer
- xmltodict
- upf_to_json

Install

You can get the source from gitlab like this:

```
$ git clone https://gitlab.com/pavanello-research-group/edftpy.git
$ python -m pip install ./edftpy
```

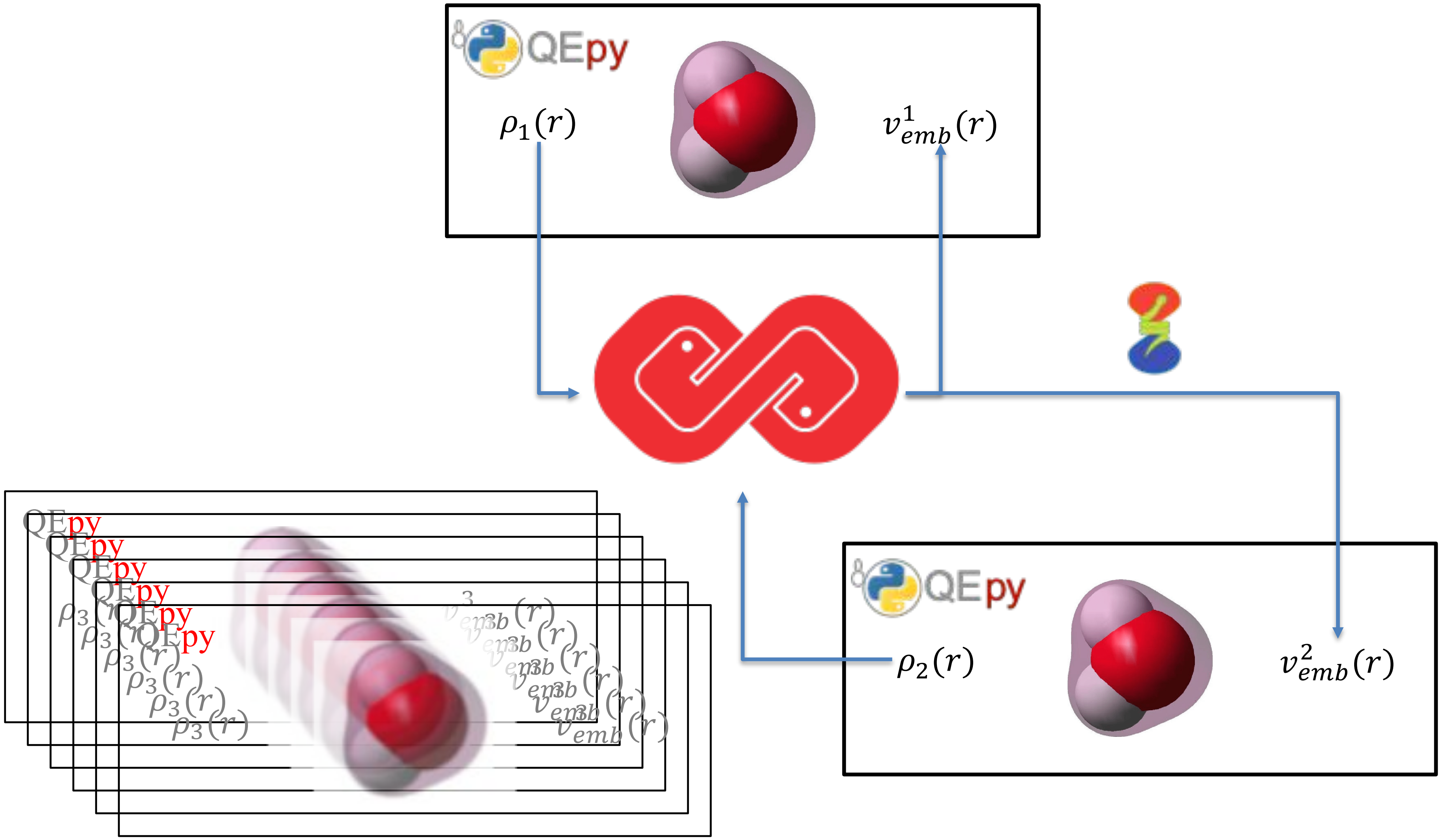
or simpler:

```
$ python -m pip install git+https://gitlab.com/pavanello-research-group/edftpy.git
```

http://dftpy.rutgers.edu/

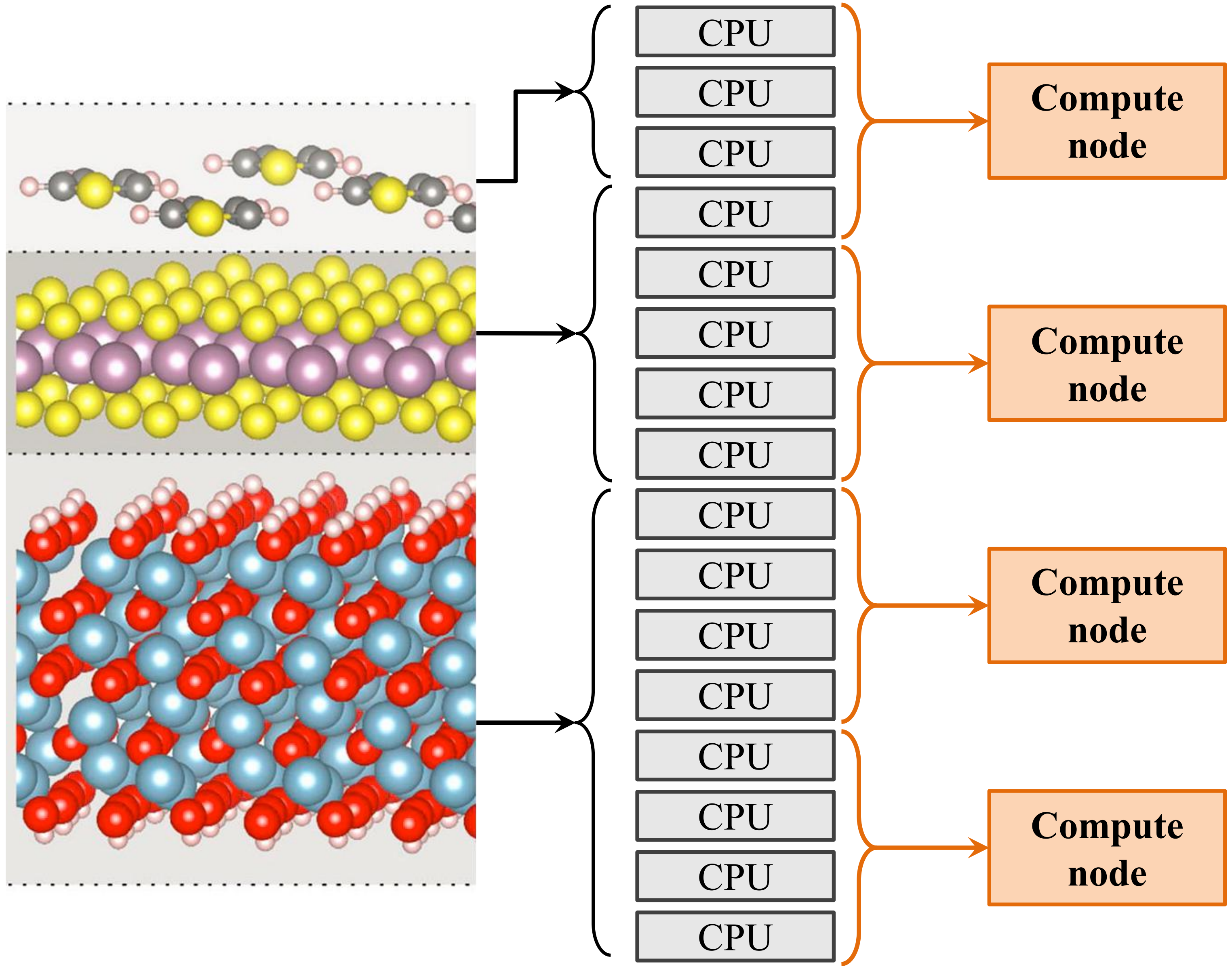
http://edftpy.rutgers.edu/

What does eDFTpy do? And why is it called eDFTpy?

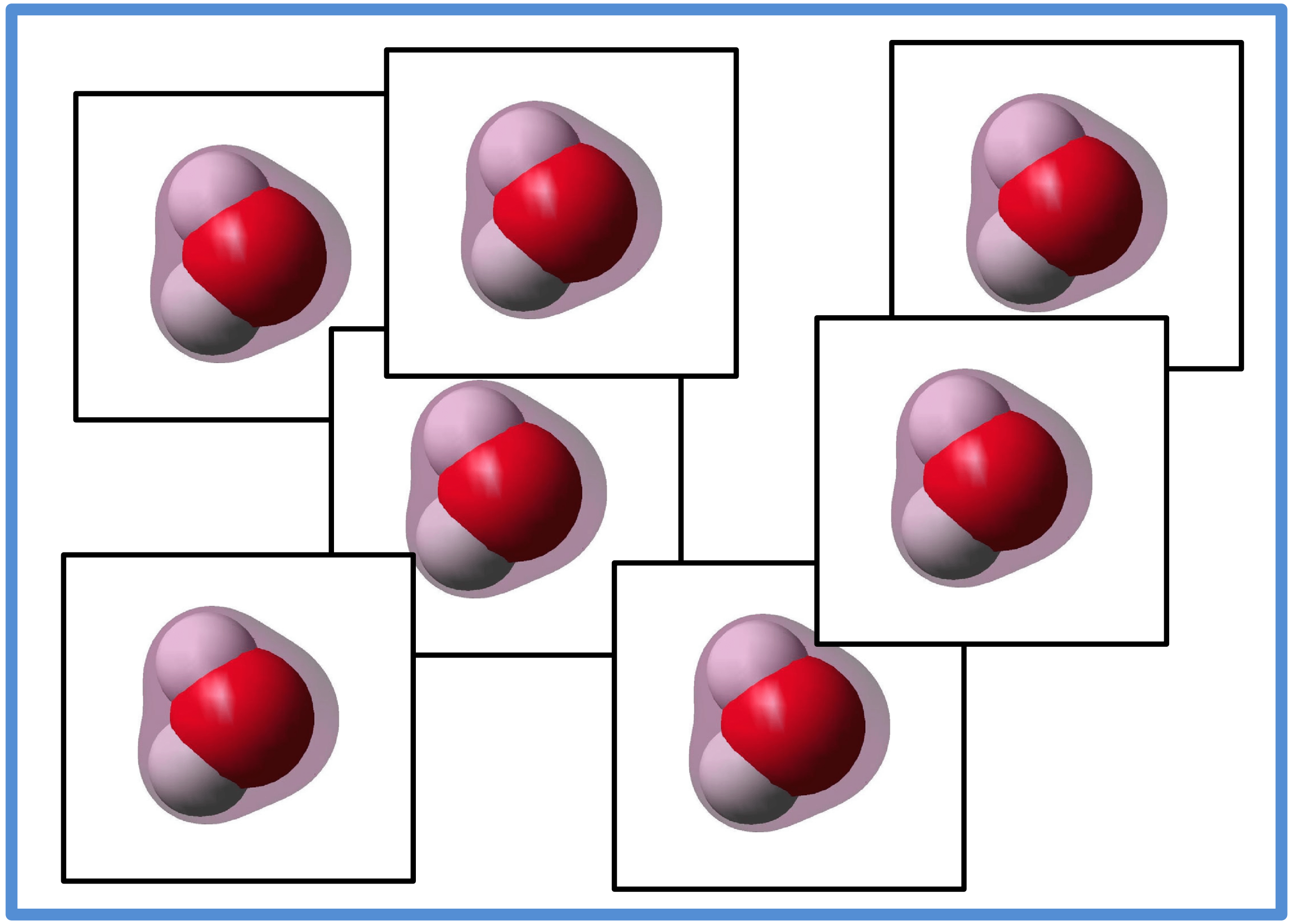


eDFTpy must handle parallel lead balancing

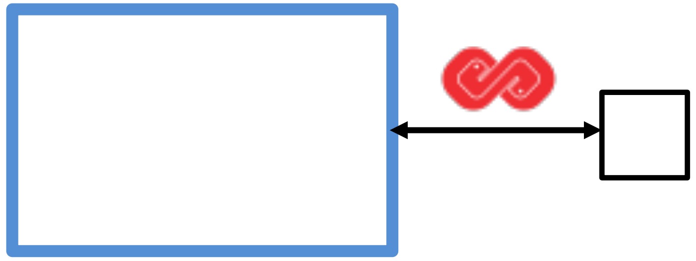
- Subsystems are assigned their own set of CPUs and simulation cell
- Global potentials (such as v_H) need to be computed on the global grid



Simulation cells and FFT grids in eDFTpy

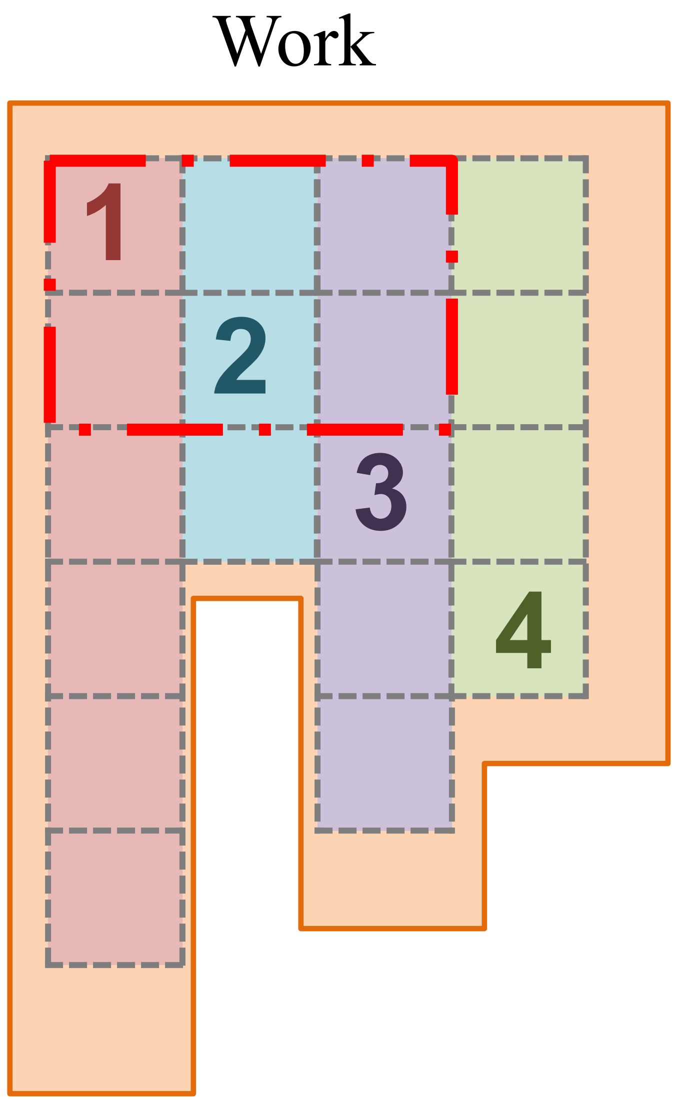


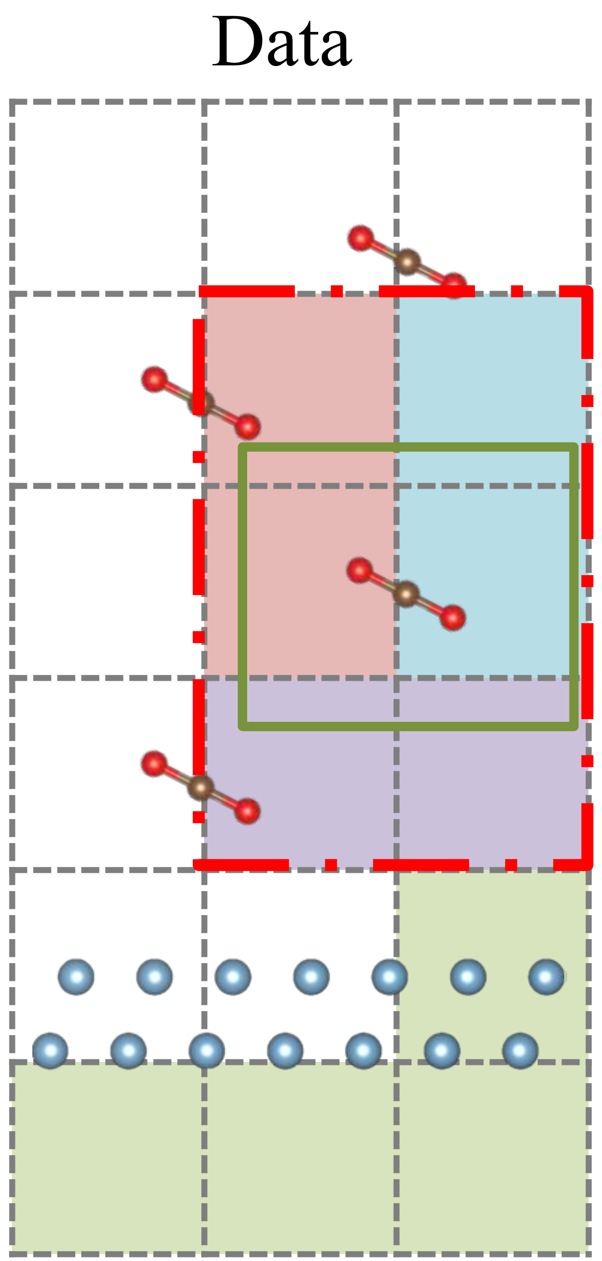
- Large simulation cell (physical)
- Subsystem cells (not physical)
- eDFTpy handles maps between them



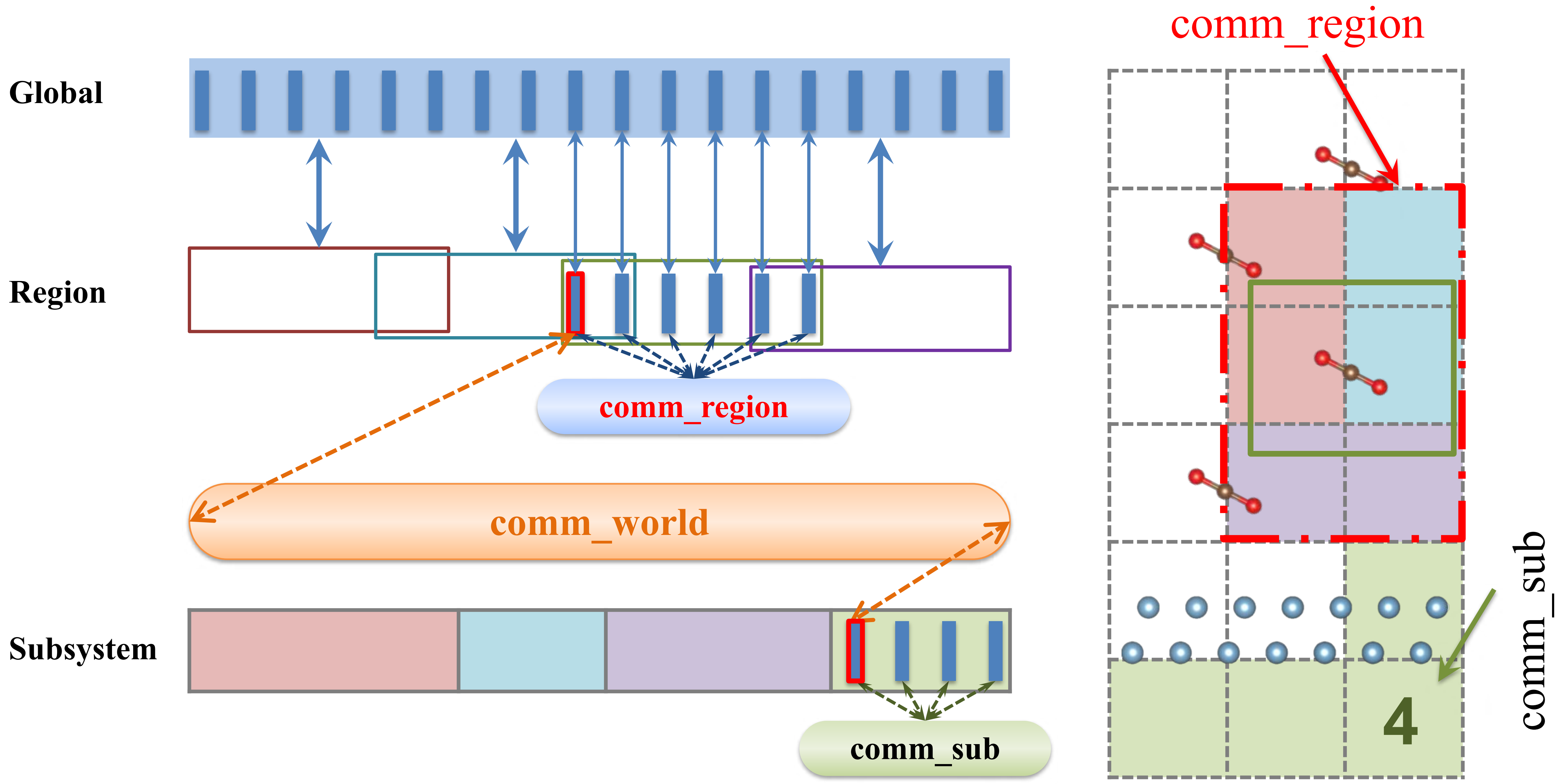
Parallel communication: brain fry

- A system composed of 4 subsystems
- Work and data are handled differently

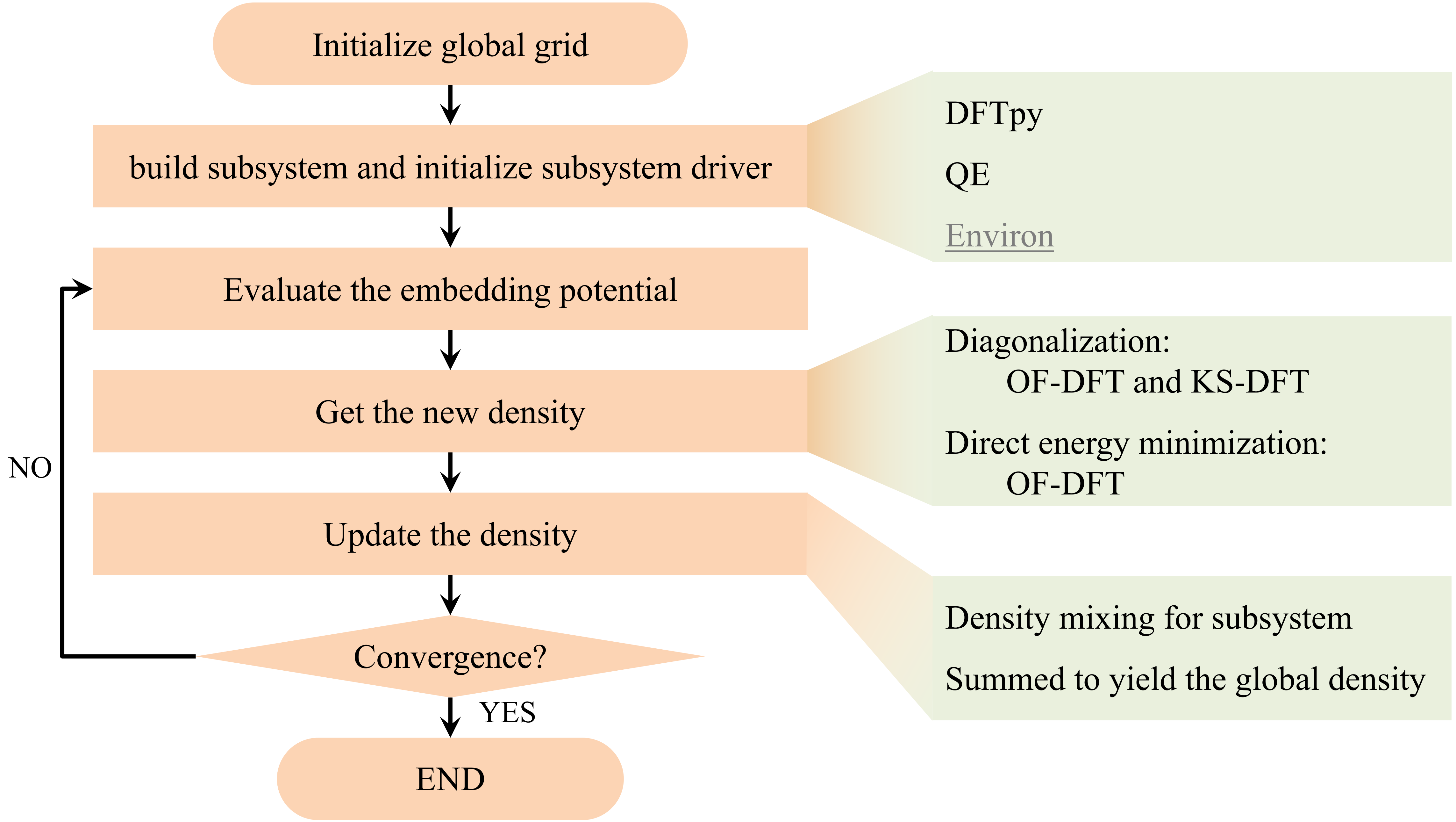




Parallel communication: brain fry^2



Flowchart of eDFTpy


Initialize global grid
build subsystem and initialize subsystem driver
Evaluate the embedding potential
Get the new density
Update the density
Convergence?
NO
YES
END
DFTpy
QE
Environ
Diagonalization:
OF-DFT and KS-DFT
Direct energy minimization:
OF-DFT
Density mixing for subsystem
Summed to yield the global density

Let’s try eDFTpy with a simple input file run!

```
[JOB]
task            =    Optdensity

[PATH]
pp              =    ./
cell            =    ./

[PP]
O               =    O_ONCV_PBE-1.2.upf
H               =    H_ONCV_PBE-1.2.upf

[OPT]
maxiter         =    200
econv           =    1e-6

[GSYSTEM]
cell-file       =    h2o_2.xyz
grid-ecut       =    1200
exc-x_str       =    gga_x_pbe
exc-c_str       =    gga_c_pbe
kedf-kedf       =    GGA
kedf-k_str      =    revAPBEK
density-output  =    total.xsf
```

./Materials/examples-scf

```
[SUB_KS_0]
calculator          =  qe
embed               =  KE
cell-split          =  0.5 0.5 0.5
cell-index          =  0:3
grid-ecut           =  2400
density-output      =  .xsf
;basefile           =  qe_in.in

[SUB_KS_1]
calculator          =  qe
embed               =  KE XC
cell-split          =  0.5 0.5 0.5
cell-index          =  3:6
grid-ecut           =  2400
```

input.ini

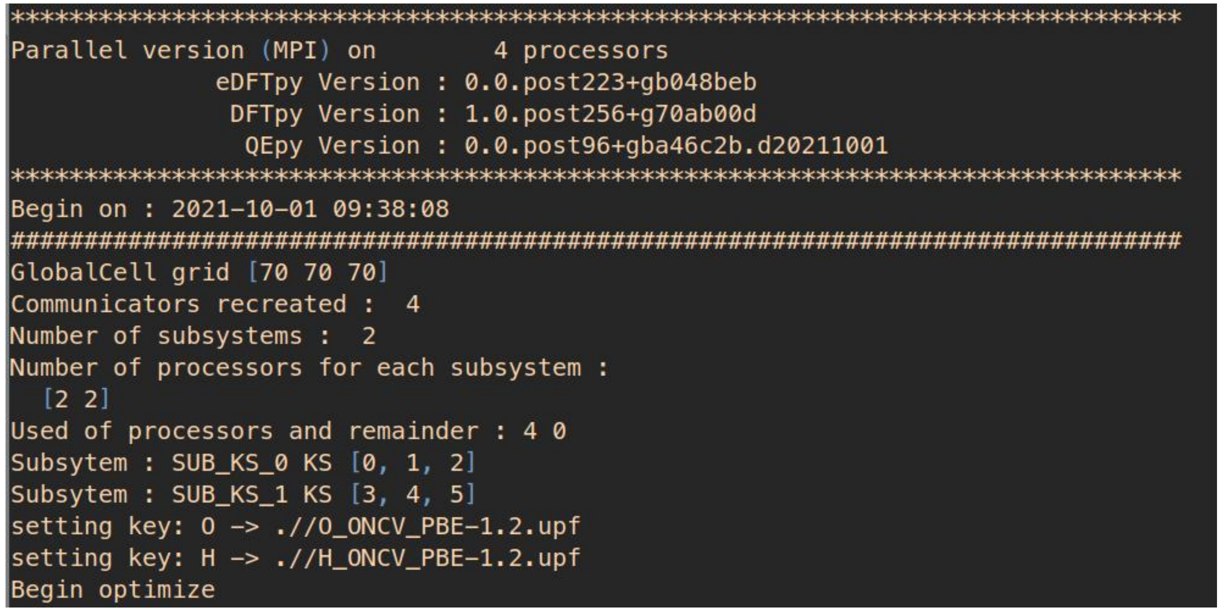
```
[SUB_KS]
calculator          =  qe
embed               =  KE XC
cell-split          =  0.5 0.5 0.5
cell-index          =  0:6
decompose-method    =  distance
decompose-radius-O  =  0.90
decompose-radius-H  =  0.60
grid-ecut           =  2400
```

input_auto.ini

Example of eDFTpy

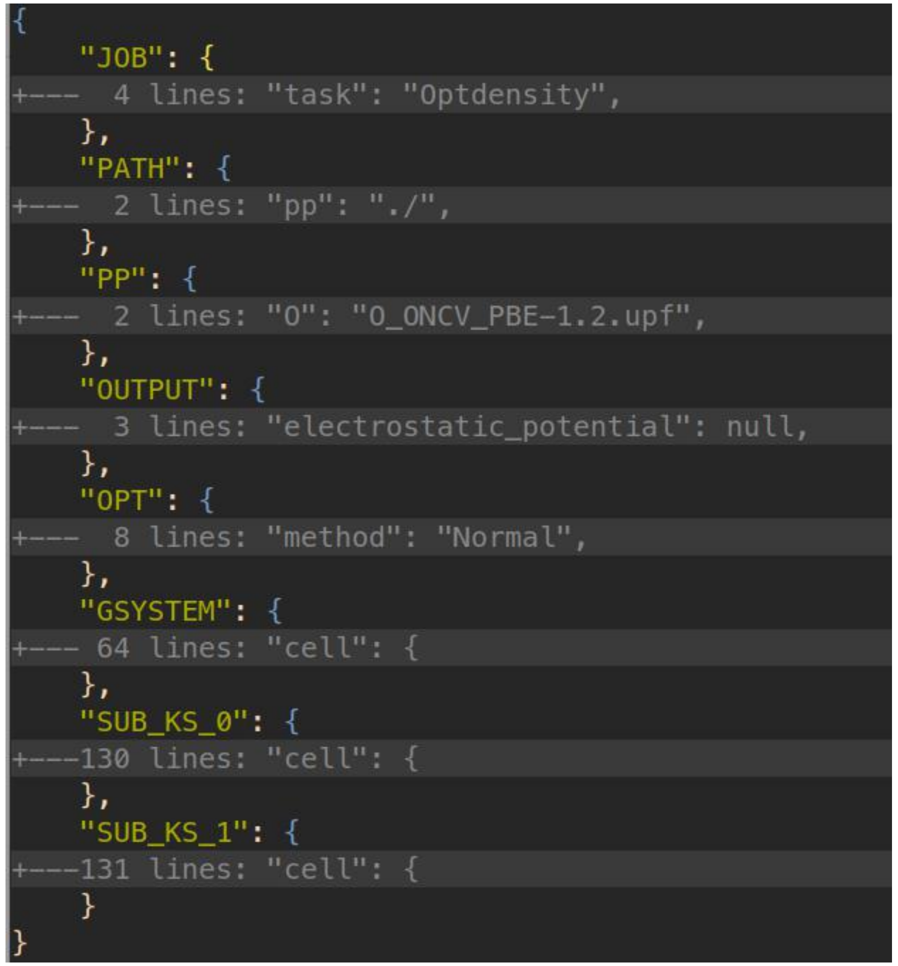
mpirun -n 4 python -m edftpy **input.ini** --mpi

mpirun -n 4 python -m edftpy **input.json** --mpi



```
********************************************************************************
Parallel version (MPI) on        4 processors
              eDFTpy Version : 0.0.post223+gb048beb
               DFTpy Version : 1.0.post256+g70ab00d
                QEpy Version : 0.0.post96+gba46c2b.d20211001
********************************************************************************
Begin on : 2021-10-01 09:38:08
################################################################################
GlobalCell grid [70 70 70]
Communicators recreated :  4
Number of subsystems :  2
Number of processors for each subsystem :
  [2 2]
Used of processors and remainder : 4 0
Subsytem : SUB_KS_0 KS [0, 1, 2]
Subsytem : SUB_KS_1 KS [3, 4, 5]
setting key: O -> .//O_ONCV_PBE-1.2.upf
setting key: H -> .//H_ONCV_PBE-1.2.upf
Begin optimize
```

input.json



```
{
    "JOB": {
+---  4 lines: "task": "Optdensity",
    },
    "PATH": {
+---  2 lines: "pp": "./",
    },
    "PP": {
+---  2 lines: "O": "O_ONCV_PBE-1.2.upf",
    },
    "OUTPUT": {
+---  3 lines: "electrostatic_potential": null,
    },
    "OPT": {
+---  8 lines: "method": "Normal",
    },
    "GSYSTEM": {
+--- 64 lines: "cell": {
    },
    "SUB_KS_0": {
+---130 lines: "cell": {
    },
    "SUB_KS_1": {
+---131 lines: "cell": {
    }
}
```

...now go ahead and run it!

./Materials/examples-scf

Structure of eDFTpy

```
h2o_2.xyz
O_ONCV_PBE-1.2.upf
H_ONCV_PBE-1.2.upf
input.ini
sub_ks_1.in
sub_ks_0.in
edftpy_running.json
edftpy_gsystem.xyz
sub_ks_1.xyz
sub_ks_0.xyz
sub_ks_0.out
sub_ks_1.out
total.xsf
sub_ks_1.tmp
sub_ks_0.xsf
sub_ks_0.tmp
```

- Structure file
- Pseudopotential files
- Input file of eDFTpy
  - input.ini or input.json

- Input files of subsystem driver
  - QE: prefix.in
- Outputs for check
  - edftpy_running.json
  - edftpy_gsystem.xyz
- Outputs of subsystem driver
  - prefix.xyz
  - prefix.out
- Temporary files of subsystem driver
  - prefix.tmp
- Some properties files defined in input file
  - density of global system and subsystem

Advanced features:

CPU Load Balancing

```
[SUB_KS_0]
calculator        =   qe
embed             =   KE
cell-split        =   0.5 0.5 0.5
cell-index        =   0:3
grid-ecut         =   2400
nprocs            =   3

[SUB_KS_1]
calculator        =   qe
embed             =   KE XC
cell-split        =   0.5 0.5 0.5
cell-index        =   3:6
grid-ecut         =   2400
nprocs            =   2
```

input_pro.ini

nprocs

- The number of processors for the subsystem
- If the minimum of all subsystems is 1, it will be used as a multiplier of # processors.

Use gaussians

```
[SUB_KS_0]
calculator              =   qe
embed                   =   KE
cell-split              =   0.5 0.5 0.5
cell-index              =   0:3
grid-ecut               =   2400
density-use_gaussians   =   True

[SUB_KS_1]
calculator              =   qe
embed                   =   KE XC
cell-split              =   0.5 0.5 0.5
cell-index              =   3:6
grid-ecut               =   2400
```

input_pro.ini

density-use_gaussians

This is to avoid problems of electrons leaking in the core region of surrounding fragments when hard pseudopotentials are employed.

Let’s do some coding!

Check out the jupyter notebooks on *./Materials/jupyter-scf*

Challenge!

./Materials/examples-scf-challenge

1) Jupyter challenge: Add you own external potential to a specific subsystem
2) Input file challenge: make an input file for **challenge.xyz** with optimal load balance

The End…

Questions?

@MikPavanello

@XuechengShao

@alesgeno