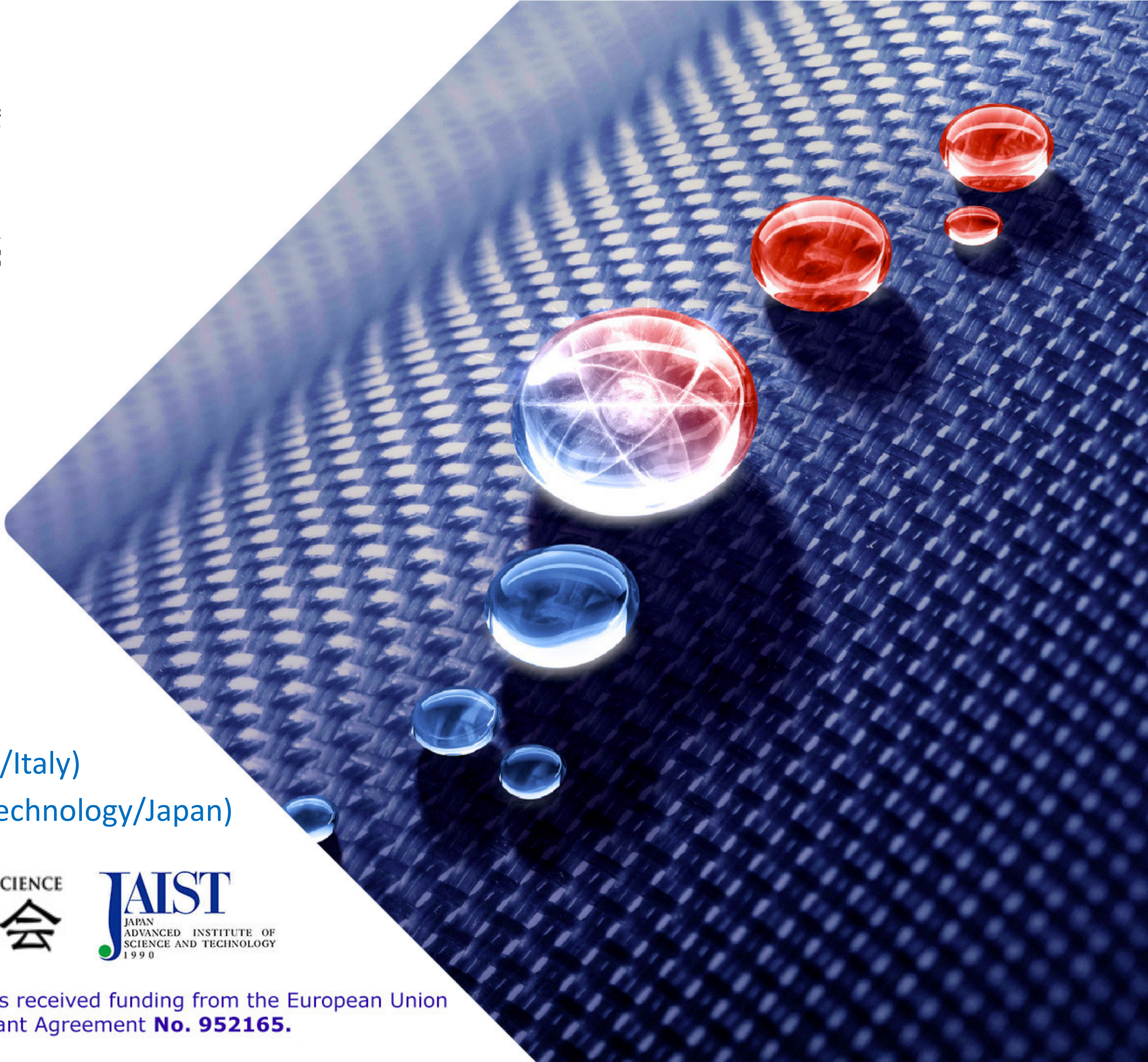
TREX-IO: hands-on session

Kosuke Nakano

- SISSA (International School for Advanced Studies/Italy)
- JAIST (Japan Advanced Institute of Science and Technology/Japan)



Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation programme under Grant Agreement **No. 952165.**

Targeting Real chemical accuracy at the EXascale

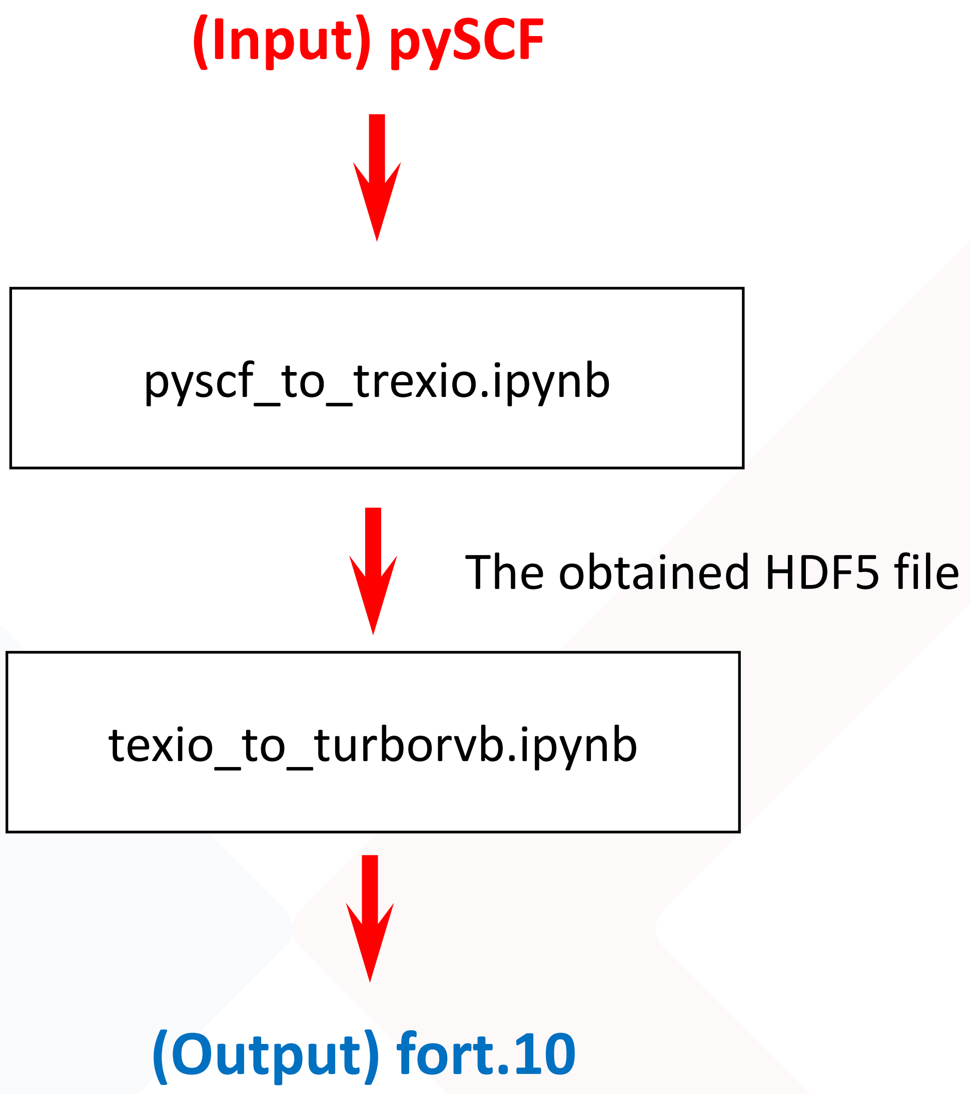
Today's topics:

- PySCF -> TREX-IO HDF5

- TREX-IO HDF5 -> TurboRVB WF

Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation programme under Grant Agreement **No. 952165.**

A workflow for pySCF -> TurboRVB



1) pySCF -> TREX-IO :

2) TREX-IO -> TurboRVB WF

PySCF -> TREX-IO

See pyscf_to_trexio.ipynb.

TODO list:

- Pseudo potential Information (No ECP info. is stored).
- shell_prim_factor implementation (for the time being, it is set 1 for all.)

Would someone help me during this hackathon??

TREX-IO -> TurboRVB

See. texio_to_turborvb.ipynb.

TODO list:

- Implementation as a “tool”, i.e., it is a jupyter notebook now.

  How trexio.py is packaged?? Could we talk about this topic during the hackathon?

- Contracted basis sets with same exponents.

- Pseudo potential Information (No ECP info. is stored)

TurboRVB and Turbo-Genius

TurboRVB
Quantum Monte Carlo Package SISSA

QMC engines (DFT, VMC-optimization, VMC, LRDMC)

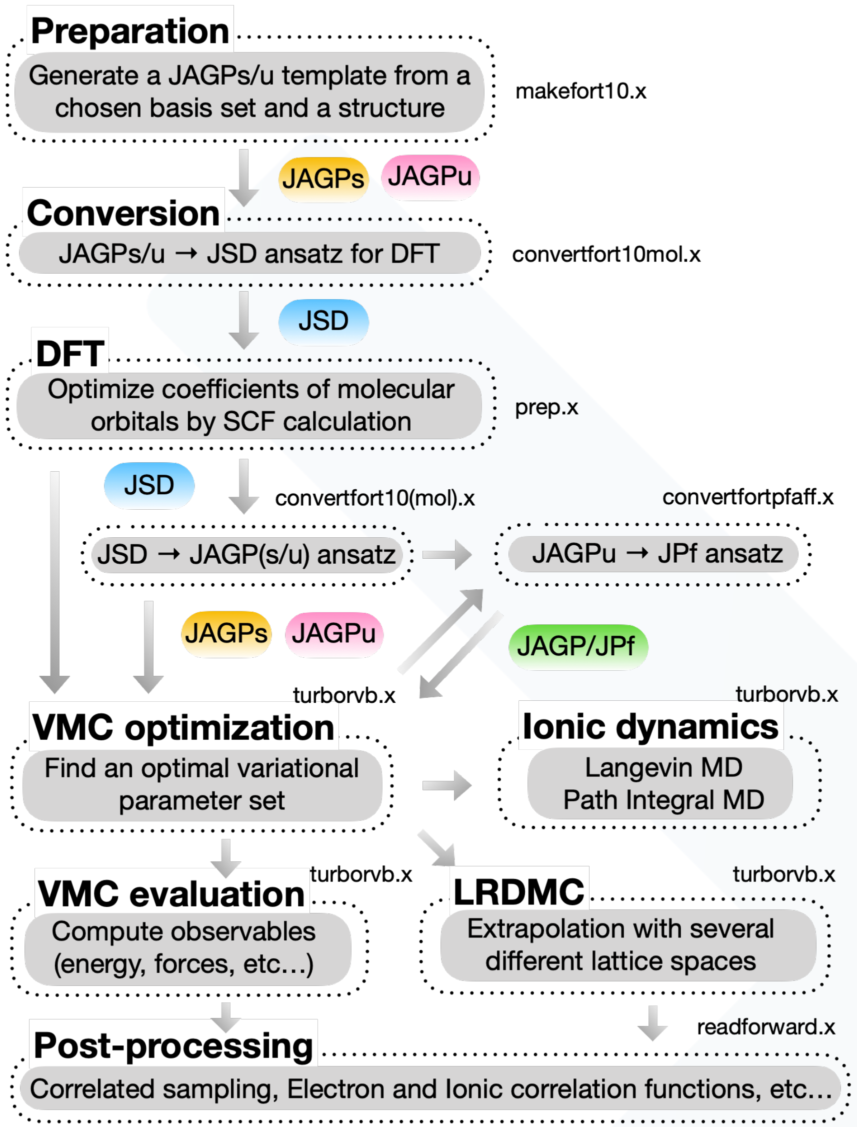
K. Nakano, C. Attaccalite, M. Barborini, L. Capriotti, M. Casula, E. Coccia, M. Dagrada, Y. Luo, G. Mazzola, A. Zen, and S. Sorella, ***J. Chem. Phys.*** 152, 204121 (2020)

TurboGenius

Python wrappers.

K. Nakano and collaborators, ***in preparation*** (2022)

A typical workflow in TurboRVB


Preparation
Generate a JAGPs/u template from a chosen basis set and a structure
makefort10.x
JAGPs
JAGPu
Conversion
JAGPs/u → JSD ansatz for DFT
convertfort10mol.x
JSD
DFT
Optimize coefficients of molecular orbitals by SCF calculation
prep.x
JSD
convertfort10(mol).x
convertfortpfaff.x
JSD → JAGP(s/u) ansatz
JAGPu → JPf ansatz
JAGPs
JAGPu
JAGP/JPf
turborvb.x
VMC optimization
Find an optimal variational parameter set
turborvb.x
Ionic dynamics
Langevin MD
Path Integral MD
VMC evaluation
Compute observables (energy, forces, etc…)
turborvb.x
LRDMC
Extrapolation with several different lattice spaces
turborvb.x
readforward.x
Post-processing
Correlated sampling, Electron and Ionic correlation functions, etc…

= Workflow =

1. Prepare a structure and basis set **makefort10.x**

↓

2. DFT Construct a reasonable initial WF! **prep.x**

↓

3. VMC-opt Optimize the wavefunction **turborvb.x**

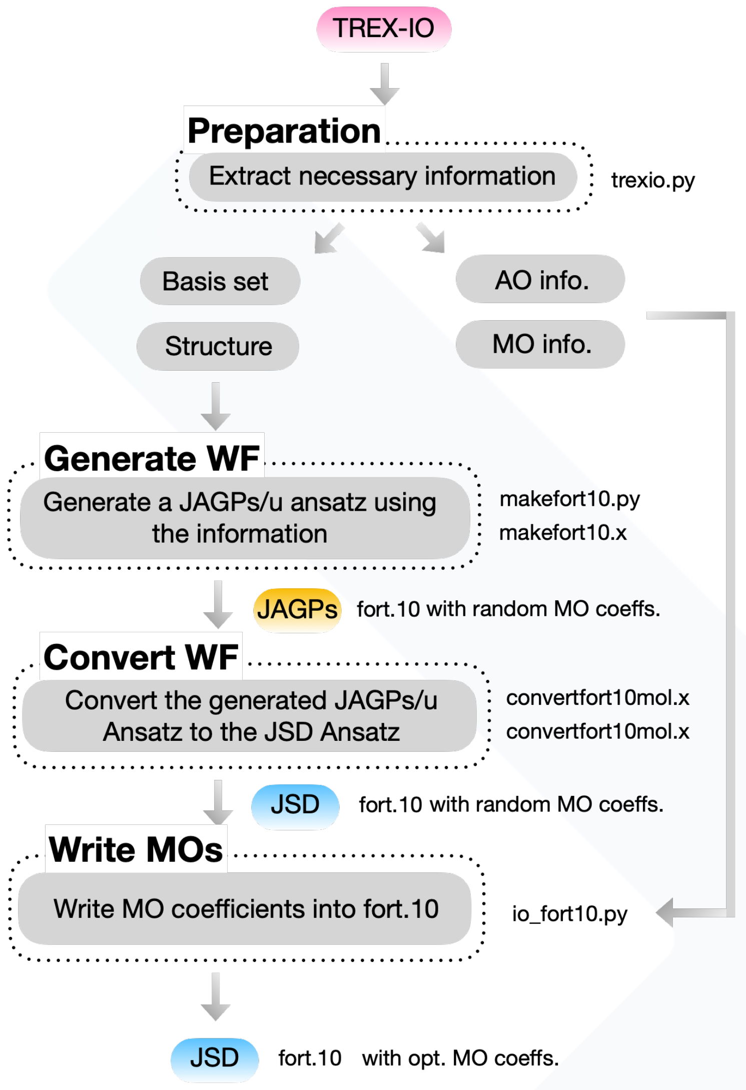
↓

4. VMC Do a VMC run. **turborvb.x**

↓

5. LRDMC LRDMC with the optimized WF. **turborvb.x**

Workflow of the TREXIO-TurboRVB converter


TREX-IO
Preparation
Extract necessary information
trexio.py
Basis set
Structure
AO info.
MO info.
Generate WF
Generate a JAGPs/u ansatz using the information
makefort10.py
makefort10.x
JAGPs
fort.10 with random MO coeffs.
Convert WF
Convert the generated JAGPs/u Ansatz to the JSD Ansatz
convertfort10mol.x
convertfort10mol.x
JSD
fort.10 with random MO coeffs.
Write MOs
Write MO coefficients into fort.10
io_fort10.py
JSD
fort.10 with opt. MO coeffs.

First, the converter generates a TurboRVB WF file using only basis set and structure information stored in a TREX-IO file.

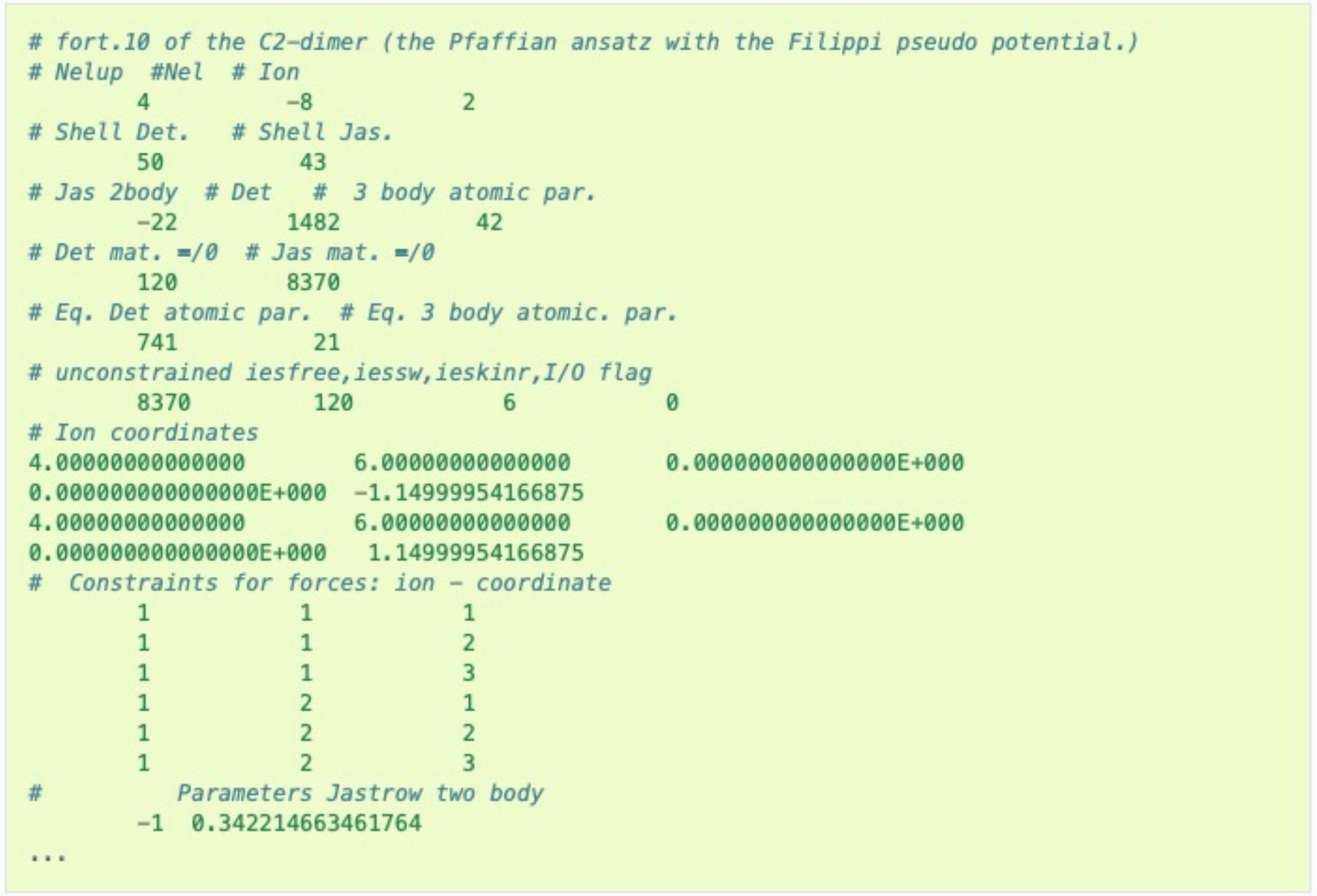
$\downarrow$

Then, the converter writes the MO information stored in a TREX-IO file into the generated WF file.

This is because of the complication of the TurboRVB WF format.

Wavefunction file (fort.10)

All the information (except for pseudo potential info.) is included in a single file, named “fort.10”

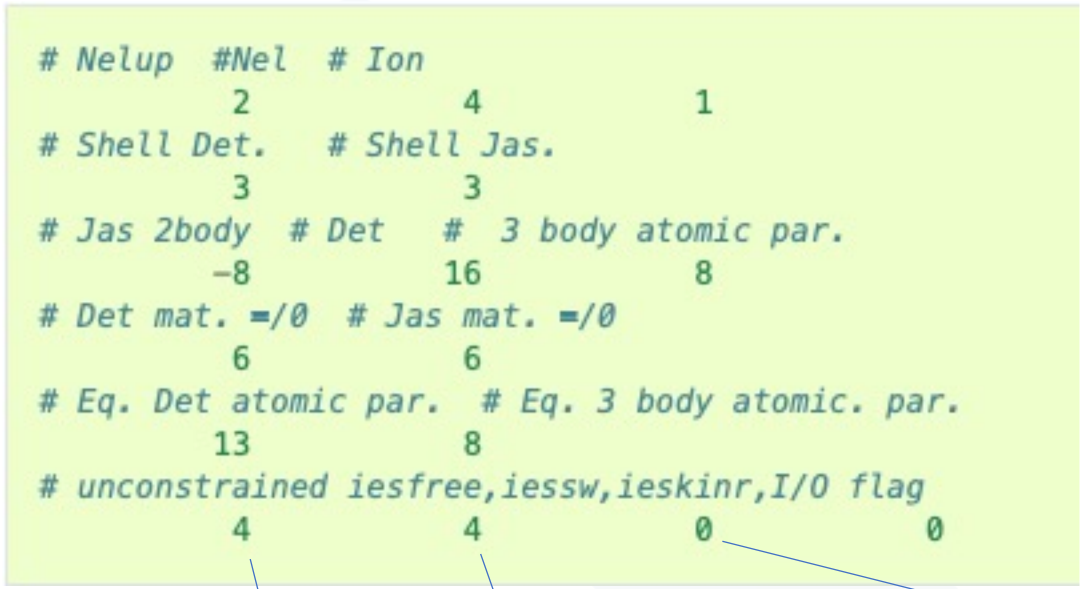


```
# fort.10 of the C2-dimer (the Pfaffian ansatz with the Filippi pseudo potential.)
# Nelup  #Nel  # Ion
        4          -8           2
# Shell Det.   # Shell Jas.
        50          43
# Jas 2body  # Det   #  3 body atomic par.
        -22        1482          42
# Det mat. =/0  # Jas mat. =/0
        120        8370
# Eq. Det atomic par.  # Eq. 3 body atomic. par.
        741          21
# unconstrained iesfree,iessw,ieskinr,I/O flag
        8370         120          6           0
# Ion coordinates
4.00000000000000        6.00000000000000       0.000000000000000E+000
0.000000000000000E+000  -1.14999954166875
4.00000000000000        6.00000000000000       0.000000000000000E+000
0.000000000000000E+000   1.14999954166875
#  Constraints for forces: ion - coordinate
        1           1           1
        1           1           2
        1           1           3
        1           2           1
        1           2           2
        1           2           3
#          Parameters Jastrow two body
        -1  0.342214663461764
...
```

“fort.10” can be generated by “makefort10.x” (see later).

fort.10 Header

Header:



```
# Nelup  #Nel  # Ion
         2           4           1
# Shell Det.   # Shell Jas.
         3           3
# Jas 2body  # Det   #  3 body atomic par.
        -8          16           8
# Det mat. =/0  # Jas mat. =/0
         6           6
# Eq. Det atomic par.  # Eq. 3 body atomic. par.
        13           8
# unconstrained iesfree,iessw,ieskinr,I/O flag
         4           4           0           0
```

Nelup: The number of spin up electrons in the system.

Nel: The total number of electrons in the system.

Ion: The number of nuclei in the system.

Jas 2body: Onebody and Twobody Jastrow types

The number of atomic forces.

The total number of determinant variational param.

The total number of Jastrow variational param.

Coordinates:

```
# Ion coordinates
 N1 Z1                  x1      y1      z1
 N2 Z2                  x2      y2      z2
   ..                  ..      ..      ..
 Nn Zn                  xn      yn      zn
```

- N: Atomic number

- Z: The number of valence electrons

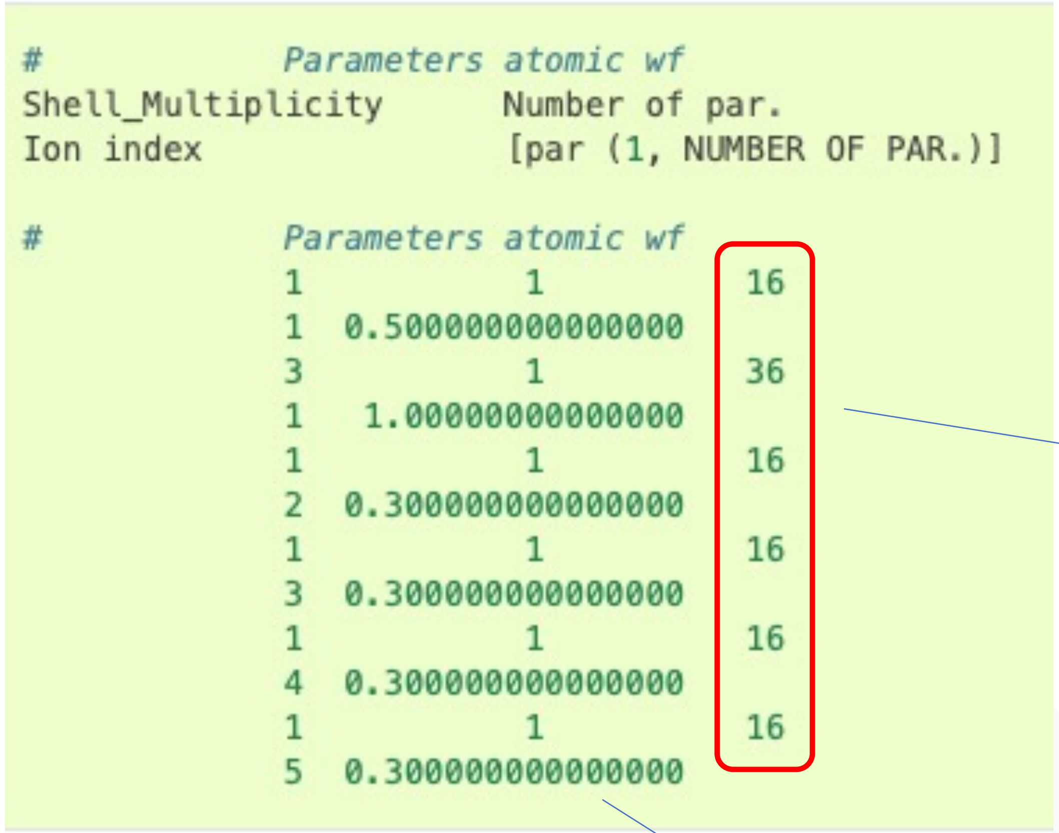
- xn, yn, zn : atomic positions (Bohr)

Pseudo potential case   N != Z

If you want to use a H-pseudo potential, please put N=1.0, Z=1.00001 (dummy).

Basis set in makefort10.input/fort.10

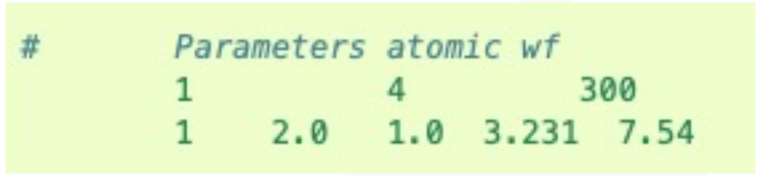
Basis set for the determinant part:



```
#             Parameters atomic wf
Shell_Multiplicity       Number of par.
Ion index                [par (1, NUMBER OF PAR.)]

#             Parameters atomic wf
              1          1          16
              1  0.500000000000000
              3          1          36
              1   1.00000000000000
              1          1          16
              2  0.300000000000000
              1          1          16
              3  0.300000000000000
              1          1          16
              4  0.300000000000000
              1          1          16
              5  0.300000000000000
```

$$\phi(r) \sim \exp(-Zr^2)$$



```
#          Parameters atomic wf
           1          4         300
           1     2.0   1.0   3.231   7.54
```

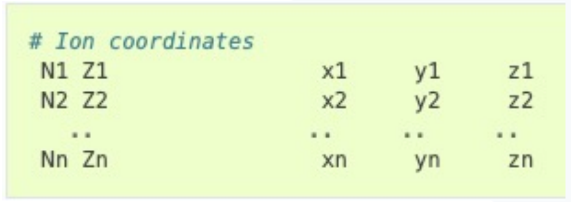
$$\phi(r) = 3.231 \cdot \exp(-2.0 \cdot r^2) + 7.54 \cdot \exp(-1.0 \cdot r^2)$$

Shell codes:	
16 -> s orbital	
36 -> p orbital	
68 -> d orbital	
48 -> f orbital	
51 -> g orbital	
72 -> h orbital	
73 -> i orbital	

Wavefunction (makefort.10.x)

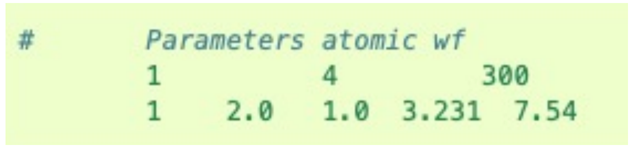
Input: makefort10.input → Binary: makefort10.x → Output:fort10_new

makefort10.x is a tool for generating JAGP WF(fort.10) from makefort10.input.



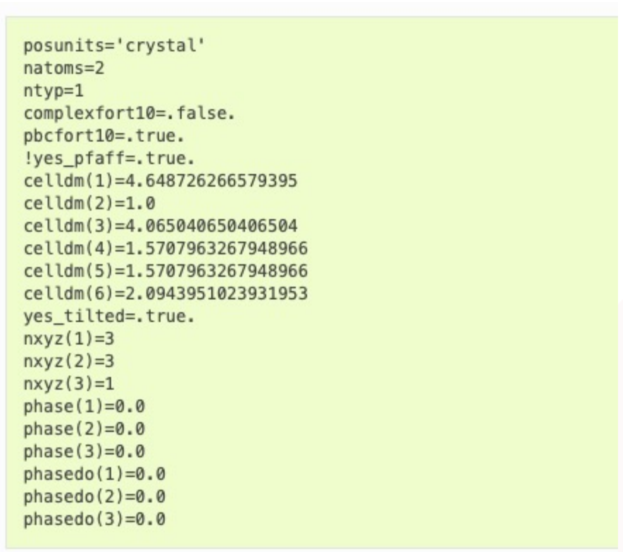
```
# Ion coordinates
 N1 Z1                   x1      y1      z1
 N2 Z2                   x2      y2      z2
  ..                     ..      ..      ..
 Nn Zn                   xn      yn      zn
```

Structural information.



```
#       Parameters atomic wf
        1         4           300
        1    2.0  1.0  3.231  7.54
```

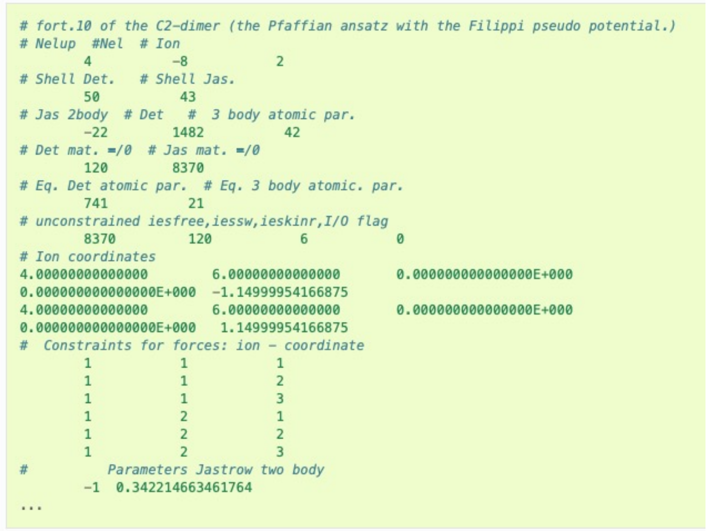
Basis-set information.



```
posunits='crystal'
natoms=2
ntyp=1
complexfort10=.false.
pbcfort10=.true.
!yes_pfaff=.true.
celldm(1)=4.648726266579395
celldm(2)=1.0
celldm(3)=4.065040650406504
celldm(4)=1.5707963267948966
celldm(5)=1.5707963267948966
celldm(6)=2.0943951023931953
yes_tilted=.true.
nxyz(1)=3
nxyz(2)=3
nxyz(3)=1
phase(1)=0.0
phase(2)=0.0
phase(3)=0.0
phasedo(1)=0.0
phasedo(2)=0.0
phasedo(3)=0.0
```

makefort10.input file

makefort10.x



```
# fort.10 of the C2-dimer (the Pfaffian ansatz with the Filippi pseudo potential.)
# Nelup  #Nel  # Ion
        4          -8          2
# Shell Det.   # Shell Jas.
        50         43
# Jas 2body  # Det   #  3 body atomic par.
        -22        1482         42
# Det mat. =/0  # Jas mat. =/0
        120        8370
# Eq. Det atomic par.  # Eq. 3 body atomic. par.
        741          21
# unconstrained iesfree,iessw,ieskinr,I/O flag
        8370         120           6           0
# Ion coordinates
4.00000000000000        6.00000000000000       0.000000000000000E+000
0.000000000000000E+000  -1.14999954166875
4.00000000000000        6.00000000000000       0.000000000000000E+000
0.000000000000000E+000   1.14999954166875
#  Constraints for forces: ion - coordinate
        1            1            1
        1            1            2
        1            1            3
        1            2            1
        1            2            2
        1            2            3
#          Parameters Jastrow two body
        -1  0.342214663461764
...
```

Wavefunction file (fort.10)

JAGPs

Single Slater-determinant (convertfort10mol.x)

$$\begin{tikzcd} \boxed{\text{Input: convertfort10mol.input, fort.10\_in}} \arrow[r] & \boxed{\text{Binary: convertfort10mol.x}} \arrow[r] & \boxed{\text{Output:fort10\_new}} \end{tikzcd}$$

convertfort10mol.x is a tool for adding molecular orbitals to fort.10_in.

This is used for converting a JAGP WF to a JSD WF. JAGPs → JSD

JSD Slater Determinant

$$f\left(\mathbf{r}_i,\mathbf{r}_j\right)=\sum_{k=1}^{M=N_{\mathrm{el}}/2}\lambda_k\tilde{\Phi}_k\left(\mathbf{r}_i\right)\tilde{\Phi}_k\left(\mathbf{r}_j\right)$$

with $\tilde{\Phi}_k=\sum_{i=1}^{N_{\mathrm{basis}}}c_{i,k}\cdot\phi_i\left(\boldsymbol{r}\right)$

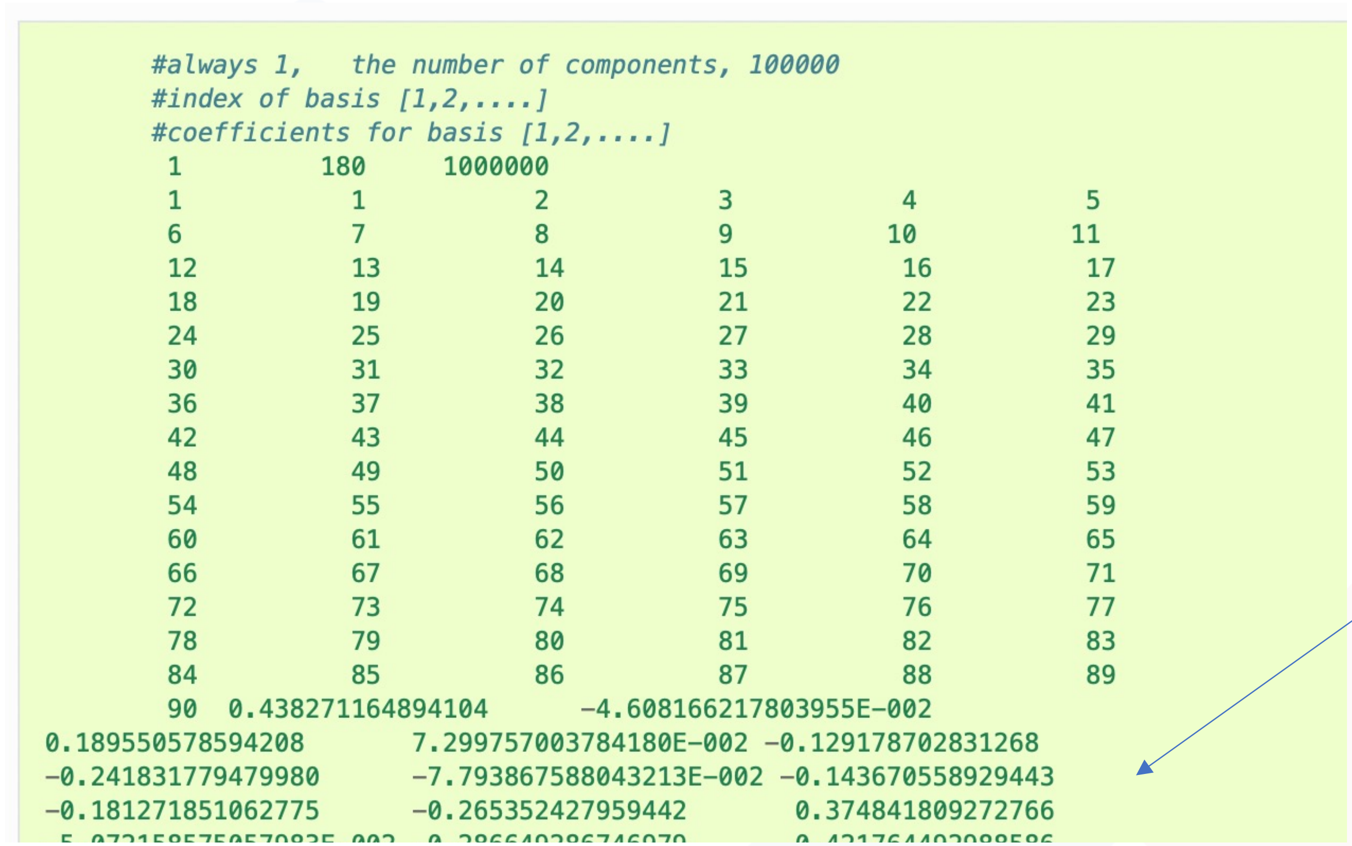
JAGPs

$$f\left(\mathbf{r}_i,\mathbf{r}_j\right)=\sum_{l,m,a,b}A_{\{a,l\},\{b,m\}}\psi_{a,l}\left(\mathbf{r}_i\right)\psi_{b,m}\left(\mathbf{r}_j\right) \rightarrow$$

DFT (prep.x) works only with molecular orbitals!! So, one should convert a WF from the JsAGPs to JSD.

Molecular orbital coeff. in fort.10

Molecular orbitals (100000): In fort.10, 1000000 indicates a molecular orbital.



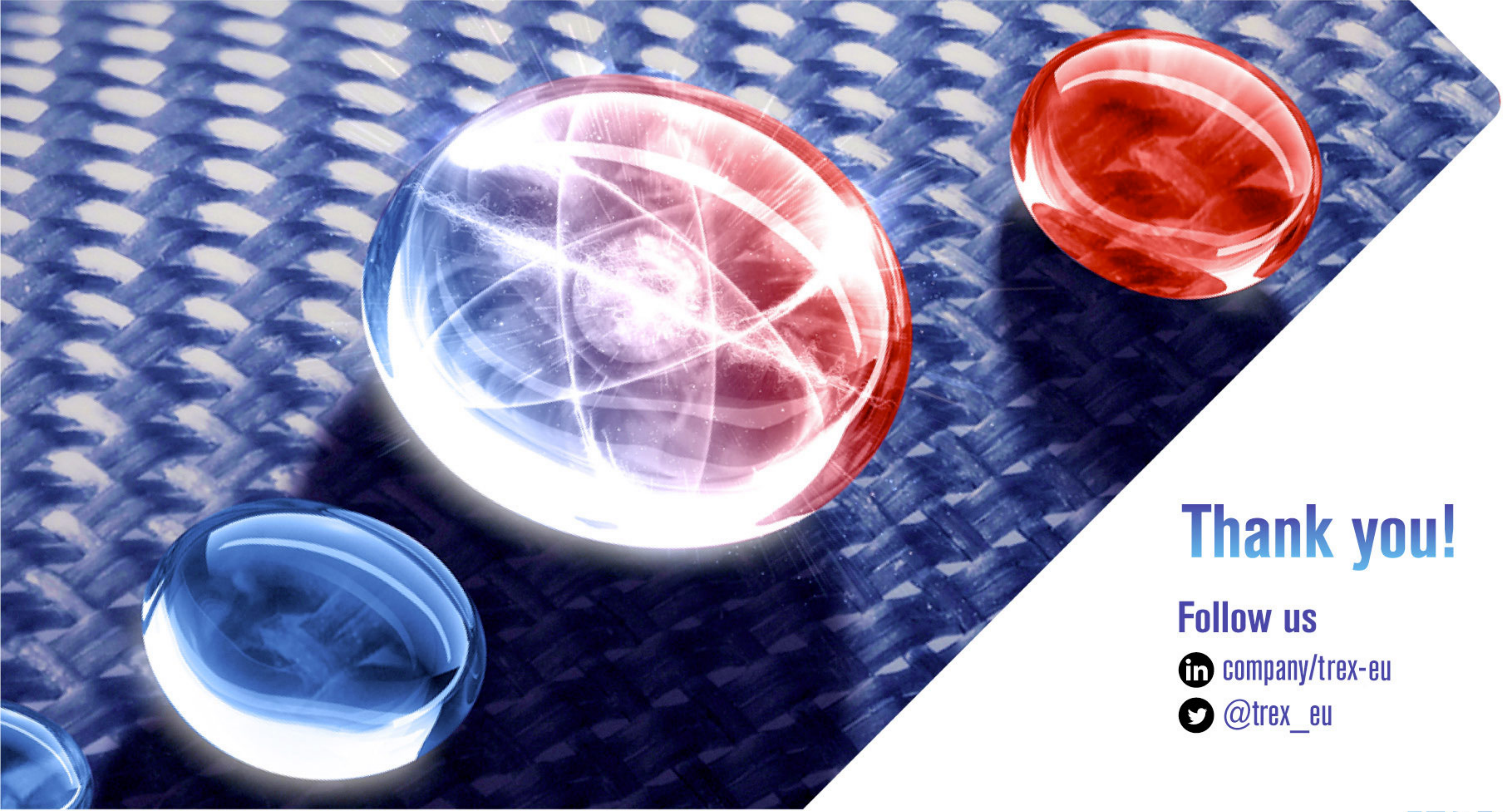
```
        #always 1,    the number of components, 100000
        #index of basis [1,2,....]
        #coefficients for basis [1,2,....]
         1         180      1000000
         1           1           2           3           4           5
         6           7           8           9          10          11
        12          13          14          15          16          17
        18          19          20          21          22          23
        24          25          26          27          28          29
        30          31          32          33          34          35
        36          37          38          39          40          41
        42          43          44          45          46          47
        48          49          50          51          52          53
        54          55          56          57          58          59
        60          61          62          63          64          65
        66          67          68          69          70          71
        72          73          74          75          76          77
        78          79          80          81          82          83
        84          85          86          87          88          89
        90  0.438271164894104      -4.608166217803955E-002
 0.189550578594208       7.299757003784180E-002 -0.129178702831268
-0.241831779479980      -7.793867588043213E-002 -0.143670558929443
-0.181271851062775      -0.265352427959442       0.374841809272766
-5.072158575057983E-002 -0.286649286746979       0.421764492988586
```

$$\Phi_k = \sum_{i=1}^{N_{\mathrm{basis}}} c_{i,k} \cdot \phi_i(\boldsymbol{r})$$

These coefficients are replaced with the values read from a TREX-IO file using a method implemented in fort10.py!

io_fort10.turborvb_basis_set_list.update_molecular_orbitals(mo_coefficient_turbo)

Molecular orbitals can be added by "convertfort10mol.x". DFT works only with molecular orbitals.



Thank you!

Follow us

company/trex-eu

@trex_eu

Targeting Real Chemical Accuracy at the Exascale project has received funding from the European Union Horizon 2020 research and innovation programme under Grant Agreement **No. 952165.**

Targeting Real chemical accuracy at the EXascale