ABACUS-DeePKS

Release 0.1

unknown

Sep 05, 2021

CONTENTS:

CHAPTER

ONE

DEEPKS

1.1 Class Hierarchy

1.2 File Hierarchy

1.3 Full API

1.3.1 Namespaces

Namespace GlobalC

Contents

Variables

- *Variable GlobalC::ld*

1.3.2 Classes and Structs

Class LCAO_Descriptor

- Defined in file__home_docs_checkouts_readthedocs.org_user_builds_abacus-deepks_checkouts_latest_source_src_lcao_LCAO_descriptor.h

Class Documentation

class **LCAO_Descriptor**

This class computes the descriptors for each atom from LCAO basis set, interfaces with pytorch to obtain the correction potential in LCAO basis, and computes the forces according to the correction potential.

For details of DeePKS method, you can refer to DeePKS paper.

Public Functions

explicit **LCAO_Descriptor**()

~LCAO_Descriptor()

void **init**(const int lm, const int nm, const int tot_inl)

only for descriptor part, not including scf

Parameters

- **lm** – max angular momentum quantum number: 'L'
- **nm** – max orbital number with the same 'L', for each 'L'
- **tot_inl** – total number of radial orbitals (sum of atoms 'I', angular number 'L' and orbital number 'N')

void **build_S_descriptor**(const bool &calc_deri)

calculate $S_{\alpha,\mu} = \langle\alpha|\phi_\mu\rangle$ overlap between lcao basis Phi and descriptor basis Alpha

Parameters **calc_deri** – 0 for $\langle\phi|\alpha\rangle$, 1 for $\langle\frac{d\phi}{dR}|\alpha\rangle$

void **deepks_pre_scf**(const std::string &model_file)

i. Load DeePKS model

ii. Initialize the deltaV Hamiltonian matrix

iii. If FORCE, initialize the matrces for force

Parameters **model_file** – path of a traced model file, provided by deepks-kit

void **cal_projected_DM**(const ModuleBase::matrix &dm)

calculate projected density matrix:

$$D^I_{nlmm'} = \sum_i \sum_{\mu,\nu} \langle\alpha^I_{nlm}|\phi_\mu\rangle c_{i,\mu} c_{i,\nu} \langle\phi_\nu|\alpha^I_{nlm'}\rangle$$

Parameters **dm** – density matrix

void **cal_descriptor**(void)

EIGENVALUE of pdm in block of I_n_l.

void **cal_dm_as_descriptor**(const ModuleBase::matrix &dm)

compute the descriptor for each atom

Parameters **dm** – density matrix

void **cal_gedm**(const ModuleBase::matrix &dm)
calculate $\frac{dE_\delta}{dD^I_{nlmm'}}$

Parameters **dm** – density matrix

void **build_v_delta_alpha**(const bool &cal_deri)
calculate $\sum_I \sum_{nlmm'} \langle\phi_\mu|\alpha^I_{nlm}\rangle \frac{dE}{dD^I_{nlmm'}} \langle\alpha^I_{nlm'}|\phi_\nu\rangle$ (for gamma_only)

Parameters **cal_deri** – 0 for 3-center intergration, 1 for its derivation

void **build_v_delta_mu**(const bool &cal_deri)
calculate $\sum_I \sum_{nlmm'} \langle\phi_\mu|\alpha^I_{nlm}\rangle \frac{dE}{dD^I_{nlmm'}} \langle\alpha^I_{nlm'}|\phi_\nu\rangle$ (for multi-k)

Parameters **cal_deri** – 0 for 3-center intergration, 1 for its derivation

void **cal_v_delta**(const ModuleBase::matrix &dm)
compute $H_{\delta,\mu\nu} = \langle\phi_\mu|V_\delta|\phi_\nu\rangle$

Parameters **dm** – density matrix

void **add_v_delta**(void)
add $H_{\delta,\mu\nu}$ to the Hamiltonian matrix

void **cal_f_delta_hf**(const ModuleBase::matrix &dm)
compute Hellmann-Feynman term of the force contribution of E_δ

Parameters **dm** – density matrix

void **cal_f_delta_pulay**(const ModuleBase::matrix &dm)
compute Pulay term of the force contribution of E_δ

Parameters **dm** – density matrix

void **cal_f_delta**(const ModuleBase::matrix &dm)
compute the force contribution of E_δ

Parameters **dm** – density matrix

void **print_descriptor**(void)
print descriptors based on LCAO basis

void **print_H_V_delta**(void)
print the H_δ matrix in LCAO basis

void **print_F_delta**(const std::string &fname)
print the force related to V_δ for each atom

Parameters **fname** – the name of output file

void **save_npy_d**(void)

The following 3 functions save the `[dm_eig], [e_base], [f_base]` of current configuration as `.npy` file, when `deepks_scf = 1`. After a full group of consfigurations are calculated, we need a python script to `load` and `torch.cat` these `.npy` files, and get `l_e_delta,npy` and `l_f_delta.npy` corresponding to the exact E, F data.

Unit of energy: Ry

Unit of force: Ry/Bohr

void **save_npy_e**(const double &ebase)

Parameters **ebase** – E_{base}, 'en.etot', in Ry

void **save_npy_f**(const ModuleBase::matrix &fbase)

Parameters **fbase** – F_{base}, in Ry/Bohr

void **cal_e_delta_band**(const std::vector<ModuleBase::matrix> &dm)
calculate $tr(\rho H_\delta), \rho = \sum_i c_{i,\mu} c_{i,\nu}$ (for gamma_only)

Parameters **dm** – density matrix

Public Members

double **E_delta** = 0.0
(Unit: Ry) Correction energy provided by NN

double **e_delta_band** = 0.0
(Unit: Ry) $tr(\rho H_\delta), \rho = \sum_i c_{i,\mu} c_{i,\nu}$ (for gamma_only)

double ***H_V_delta**
Correction term to the Hamiltonian matrix: $\langle\psi|V_\delta|\psi\rangle$.

ModuleBase::matrix **F_delta**
(Unit: Ry/Bohr) Total Force due to the DeePKS correction term E_δ

1.3.3 Variables

Variable GlobalC::ld

- Defined in file__home_docs_checkouts_readthedocs.org_user_builds_abacus-deepks_checkouts_latest_source_src_lcao_LCAO_descriptor.cpp

Variable Documentation

LCAO_Descriptor `GlobalC::ld`

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

G

L