ecalj Install

1. Some software tools including fortran compilars

ecalj mainly ade from fortran source codes located at
`ecalj/SRC/main/*.f90` and at `ecalj/SRC/subroutines/*.f90`.
In addition, python are used. Bash scripts remain (but gradually replacing bash with python).

We may need to install following tools and libraries.We need

> git, gfortran, openmpi, bash, cmake intel-mel

- git (To download the ecalj. It is convenient to upgrade your code)
- Fortran compiler (we can choose gfortran, ifort, or nvfortran)
- Math library (blas, lapack, fft). We can usually use intel-mkl.
- MPI library (open mpi works for ubuntu24)
- cmake, make, bash, gnuplot

We can use apt to install them when ubuntu. Similar in other systems, or your system already have.

I use following versions for thinkpad T14: ubuntu 24.04
openmpi-bin/noble,now 4.1.6-7ubuntu2 amd64
openmpi-common/noble,noble,now 4.1.6-7ubuntu2 all
cmake/noble,now 3.28.3-1build7 amd64
make/noble,now 4.3-4.1build2 amd64
gfortran/noble,now 4:13.2.0-7ubuntu1 amd64
intel-mkl/noble,now 2020.4.304-4 amd64

PROF

- I use mpirun (Open MPI) 4.1.6 for ubuntu24.
- git makes things easier. Especiall for version up. >git diff at ecalj/ shows orginal and your modification. gitk --all show all the history of ecalj.

2. Python and python modules.

> [!TIP]
> We need python >3.9. Usually we will prepare the latest Python in your ./local.

We need to install following modules (see step4 with pip.). Usually we can use venv, anaconda or something.
Here we show a case using mise.

> The mise is a package management software (We can use anaconda instead). Or you can install tools at the following step 4.
> I think you can install all python tootls just by venv, which is a default in python.

1. Add the following settings to `~/.bashrc` for the automatic installation and activation of `mise`:

```
export PATH="$HOME/.local/bin:$PATH"
type mise > /dev/null 2>&1 || curl https://mise.run | sh
eval "$(~/.local/bin/mise activate bash)"
```

2. Update `~/.bashrc` to install `mise`:

```
source ~/.bashrc
```

3. Install `python` using `mise`:

```
mise use python@latest -g
```

4. Install the required `python` libraries:

```
pip install numpy pandas seekpath spglib pymatgen mp-api scipy plotly
```

3. Install and InstallTest

For ohtaka and kugui in ISSP, skip here and see here

Get ecalj package

```
git clone https://github.com/tkotani/ecalj.git # Get source code
```

PROF

After you did the above git clone command, a directory ecalj/ appears.
We can check history of ecalj by ">gitk --all" at ecalj/ directory after you got git clone.

Run InstallAll.py

To install use

```
InstallAll.py [Options]
```

InstallAll.py --h shows a help as

```
```

```
takao@t14:~/ecalj$ ./InstallAll.py -h
usage: InstallAll [-h] [-np NP] [--clean] [--gpu] [--bindir BINDIR] --fc
FC [--notest] [--verbose]

Install ecalj and run tests.

options:
  -h, --help       show this help message and exit
  -np NP           number of mpi cores for install test
   default: 8
   specify the number of MPI parallelization in test calculation
  --clean          Clean CMakeCache CMakeFiles before make
   default: none
   delete the cache files before compiling
  --gpu            nvfortran for GPU
   default: none
   compile the GPU and GPU-MP version
  --bindir BINDIR  ecalj binaries and scripts
  --fc FC          fortran compilar gfortran/ifort/nvfortran
  --notest         no test. only compile
  --verbose        verbose on for debug
```

InstallAll.py writes binaries and scripts to a directory foobar given by --bindir foobar/.
(Defaults is $HOME/bin/.) Add the directory foobar to your path. --fc is required.
--fc nvfortran together with --gpu is needed for GPU.

It performs compile and link followed by the install test
at ecalj/SRC/TestInstall/. (testecalj.py is the script for test)
If succeeded, we see 'OK! All PASSED!' at the end of tests.
The compile and install test may take 5~10 minutes.

Set command path BINDIR. For example, write

PROF

```
PATH="~/bin/:$PATH"
```

in your .bashrc when you move all ecalj binaries to your ~/bin.

Install VEST and getsyml

It is convenient to see structures with VESTA.
(I installed VESTA-gtk3.tar.bz2 (ver. 3.5.8, built on Aug 11 2022, 23.8MB) on ubuntu 24)
At ecalj/StructureTool/, we have 'viewvesta' command. Try

```
viewvesta ctrl.si
```

to check the structure in viewer.At /StructureTool, we have exchange converters,
`vasp2ctrl` and `ctrl2vasp`. These allows convert structures with POSCAR.

In addition, we need to install getsyml.py to obtain symmetry line for band plot.
Generated syml.* is used for the band plot in ecalj. (syml is a little strange... we will fix).
As long as you have spglib and seekpath, we don't need extra things to do.
But here is a memo for install ./GetSyml/README.org.

Additional memo

- When InstallAll have finished, we have all required binaries and shell scripts in a directory specified by --bindir (Default is your ~/bin/).

- Clean up by CleanAll:
  If something wrong, run InstallAll.py with --clean.
  You may need to do 'rm -f CMakefiles CMakeCache.txt' at ecalj/SRC/exec/ (and all *.mod files under SRC/).--- you do not need to do this usually.

- Compile fortran only.
  To compile fortran source only, move to ecalj/SRC/exec/ and run

> FC=fortran cmake . -D CMAKE_BUILD_TYPE=Debug
> , for example. You may look into CMakeLists.txt.
> Remove CMakeCache.txt and CMakeFiles/ if you want to recompile.

- Compilar bug: In cases, we have troubles due to the compilar.
  Usually we use -O2 in CMakeList.txt.
  But we may need to use -O1 or -O0 for some files to avoid compilar bugs.
  We may set some conditional compilation settings. May need to examine CMakelists.text

- Souce codes, Test, make system are under SRC/
  SRC/
  ├── TestInstall : Root of Install test
  ├── exec : CMakeLists.txt and scripts
  ├── main : All main *.f90
  └── subroutines : All subrouitnes. *.f90.
  All fortran codes are in main/ and subrouitnes/
  We have a CMakeLists.txt which generates Makefile. Look into it.

- Install test system at ecalj/SRC/TestInstall.
  We have a test system with make at ecalj/SRC/TestInstall. Look into test.py and testecalj.py.
  These controls all the test.

I think it is not so difficult to add your own test to testecalj.py.
You have to compute something at first. Then inputs and minimum results are stored in a directory.
Then you describe the test in testecalj.py.

To test all of binaries, just do

```
./test.py
./test.py gwall !tests only GW part.
./test.py si_gwsc nio_gwsc !test si_gwsc and nio_gwsc only.
```

- openmpi failed on ubuntu22. I obseved that gfortran+openmpi failed for ubuntu22. Use mpich. But I don't know cucrrent status.

補足

- Qiitaでの解説(https://qiita.com/takaokotani/items/9bdf5f1551000771dc48)