ecalj Install

To install ecalj package, we have to install softwares and python modules, which are used in ecalj.
In addition, we make some softlinks for convenience. Most of all are almost automatic. Follow steps below.

1. Some software tools including fortran compilars

Main part of ecalj is written in fortran90. Source codes are located at `ecalj/SRC/main/*.f90` and at `ecalj/SRC/subroutines/*.f90`.
GPU implementation with OpenACC/CuBlas is embedded in the source codes switched by cpp.
In addition, python are used. But we still use bash scripts in cases (but gradually replacing bash with python).

We may need to install following tools and libraries.

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

We can use apt to install them for ubuntu. Similar in other systems. Recently, we are asked to install things to our own .local in cases.

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 gnuplot cif2cell
```

PROF

For latest OS such as linux, we are asked to install these in your `.local/` directory after generating your environment with venv. (Ask chatGPT or something).

3. Install and InstallTest

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

Get ecalj package

The ecalj package is at https://github.com/tkotani/ecalj/
Check out master branch (but in cases we may put an experimental version.
So, it might be better to ask us before pull it).

(For machine at ISSP, see https://github.com/msobt/ecaljdoc)

The command to clone ecalj is

```
git clone https://github.com/tkotani/ecalj.git
```

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.

InstallAll.py

To install, we 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
```

PROF

InstallAll.py copies all binaries and scripts for ecalj to your directory foobar specified by --bindir foobar/.
(Defaults is $HOME/bin/.) Set foobar/ as you like. --fc is required.
For GPU, --fc nvfortran together with --gpu is needed.

Install ecalj

InstallAll.py performs compile and make softlink, 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.

To install ecalj, do `./InstallAll.py --help'. As it shows,

```
./InstallAll.py --fc gfortran
```

or something. For GPU, we do `./InstallAll.py --fc nvfortran --gpu`. `--clean` is for starting from scratch (no cache).

Then `./InstallAll.py --fc gfortran` compile Fortran source codes, link, and copy all programs as well as scripts to your bin. Then it runs the minimum install tests at ecalj/SRC/TestInstall. It may take ~ 10 minutes or so.

- It is helpful to check ./InstallAll.py. It describes where we install binaries (BINDIR=~/bin as defaults). Tools getsyml, viewvesta, vasp2ctrl are softlinked.

Finally, you will have to finish things as

```
PASSED! TEST 1 out.lmf.copt
PASSED! TEST 1 out.lmf.te
...
PASSED! nio_gwsc/QPU
PASSED! nio_gwsc/log.nio
PASSED! fe_gwsc/QPU
PASSED! fe_gwsc/QPD
PASSED! fe_gwsc/log.fe
PASSED! ni_crpa/Screening_W-v.h
PASSED! ni_crpa/Screening_W-v_crpa.h
PASSED! srvo3_crpa/Screening_W-v.h
PASSED! srvo3_crpa/Screening_W-v_crpa.h
OK! ALL PASSED ===
   See work/summary.txt

real    4m49.712s
user    19m38.952s
sys 3m38.583s
```

PROF

Then things fine! (in this case we take less than five minutes.)

You can run test one by one at ecalj/SRC/TestInstall/ with the command ./testecalj.py si_gwsc.

- Each tests have each directory such as si_gwsc.
  Each test generates ecalj/SRC/TestInstall/work/si_gwsc, run computation, and compare results with those already done at ecalj/SRC/TestInstall/si_gwsc. Tests are simple and described at testecalj.py. It is easy to add your test.

- When installation failed, we may need to restart from clean, you may need to delete CMakeFiles and CMakeCache at ecalj/SRC/exec/. And in cases I had trouble of old *.mod remains under SRC/. But usually, we only need to run ./InstallAll.py with the option `--clean. --clean should work to delete them.

- In some machines, parallel make(make -j) failed. In such a case, run make by hand as

```
cd ecalj/SRC/exec/
rm -rf CMakeFiles CMakeCache.txt
FC=nvfortran cmake .
make
```

After insallted set command path BINDIR. For example, write

```
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 the viewer. At /StructureTool, we have converters,
`vasp2ctrl` and `ctrl2vasp`. These allows us to 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.
As long as we have spglib and seekpath, we don't need extra things to do.
But here is a memo for install ./GetSyml/README.org.

Additional memo

PROF

- When we have InstallAll.py have finished, all binaries and shell scripts are copied to --bindir foobar (Default is your ~/bin/).
- Clean up:
  If something wrong, run InstallAll.py again with --clean.

  > You may need to do 'rm -f CMakefiles CMakeCache.txt' at ecalj/SRC/exec/ (and all *.mod files
  > under SRC/). ---> I think 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/ in advance if you want to make things clean.

- Compilar bug: In cases, we have troubles due to the compiler with the optimization -O2.
  We can often detour such bugs with less optimization options -O1 or -O0 dependent of source files, as described in CMakeList.txt (Usually people do not use -O3, I think). We may set such conditional compilation settings. See 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 with cmake.

- Test system: install test system at ecalj/SRC/TestInstall.
  We have a test system 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 comparison check in testecalj.py. To test all of binaries

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

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