Compile

target： 编译 ./libs/ 中**py_fock.xxx.so**, **py_integral.xxx.so**, **hij_tensor.xxx.so** 三个动态链接库

py_fock.xxx.so： 生成所有的onstate

py_integral.xxx.so: 读取 pyscf 中积分信息

hij_tensor.xxx.so pytorch 接口 计算矩阵元$\langle i|H|j\rangle$，获取所有单双激发等

py_fork 和 py_integral

- 安装**gcc** (< 12.00, 最好11), **boost**, intel **oneapi** (2023 或 2022) , **pybind11**, python >= 3.8 (pytorch 2.0)
- 修改Makefile 中 **CXX** 和 **MATHLIB** 位置

```
CXX = g++
MATHLIB = /opt/intel/oneapi/mkl/2023.0.0/lib/intel64
MATH = -L$(MATHLIB) -Wl,-rpath,$(MATHLIB) -lmkl_intel_lp64 -lmkl_core -lmkl_sequential -lpthread -lm -ldl
CXXFLAGS =  $(MATH)
suffix = $(shell python3-config --extension-suffix )
pybind = $(shell python3 -m pybind11 --includes)
# include_dirs= ./include

# cuda
```

- **make py_fock** ; **make py_integral** (-mv 这一换行可以注释)
- MacOS 可能需要添加**-undefined dynamic_lookup**

```
# $(CXX) $(SRC) $(MATH)  -o $(TARGET)
main: $(OBJECTS) hamiltonian onstate integral onspace
py_fock:  py_fock.cpp
    $(CXX)  -O3 -Wall -shared -std=c++17 $(MATH) -fPIC $(pybind) py_fock.cpp  $(SRC) -o py_fock$(suffix)
    -mv py_fock$(suffix) ../interface/
Hn_test: Hn_hamiltonian_test.cpp
    $(CXX) -O3 $(CXXFLAGS) Hn_hamiltonian_test.cpp $(OBJECTS) -o Hn_test
py_integral: py_integral.cpp
    $(CXX) -O3 -Wall -shared -std=c++17 -fPIC $(pybind) tools.cpp  py_integral.cpp -o py_integral$(suffix)
    -mv py_integral$(suffix) ../interface/
hamiltonian: test_hamiltonian.cpp
```

hij_tensor

- sh compile.sh -s **GPU** or sh complie.sh -s **CPU** . 使用-s 控制编译 CPU 和 GPU 代码
- 用**os.environ["CC"]** 和 **os.environ["CXX"]** 设置 gcc和g++ 命令

```
cat > setup.py  <<EOF
import glob
import os
import platform
import os.path as osp

sys_name = platform.node()
print(f"sys_name: {sys_name}")
if sys_name == "myarch":
    os.environ["CC"] = "gcc-11"
    os.environ["CXX"] = "g++-11"
    os.environ["CUDA_HOME"] = '/home/zbwu/soft/anaconda3'
    os.environ["MAX_JOBS"] = '2' # ninja

from setuptools import setup
from torch.utils.cpp_extension import CUDAExtension, BuildExtension, CppExtension

ROOT_DIR = osp.dirname(osp.abspath(__file__))
include_dirs = [osp.join(ROOT_DIR, "include")]
sources = glob.glob('*.cpp') + glob.glob('*.cu')

s = "hij_tensor"

```